

Universidad Nacional Autónoma de Nicaragua – León
Facultad de Ciencias y Tecnología
Departamento de Computación.



MONOGRAFÍA PARA OPTAR AL TÍTULO DE INGENIERIA EN SISTEMAS DE INFORMACION

Tema: Miscelánea de Aplicaciones Web para apoyar la asignatura
Programación Orientada a la Web – Modalidad Sabatina.

Autores: Ángel Reniery Pérez Delgado.
Edilberto Ramírez Navas.

Tutor: MSc. Aldo René Martínez Delgadillo.

Fecha: León, 05 de Octubre del 2011.

Contenido

Síntesis	6
Introducción	2
Antecedentes.....	4
Justificación	5
Objetivo General	6
Objetivos Específicos.....	6
Marco Teórico.....	7
ASPECTOS FUNDAMENTALES DE HTML.....	11
HTML Básico.	11
Estructura de los documentos HTML.....	11
La cabecera en HTML	11
El cuerpo en HTML	12
Los encabezados	12
Estilos de párrafos: p, center, br, hr y blockquote	13
Formato de texto: B, I, U, SUB, SUP	13
Formato de texto: Font.....	13
La etiqueta <a>:.....	14
Los enlaces:	14
La inserción de imágenes con img.....	14
Imágenes y Enlaces	15
Las listas desordenadas	15
Las Listas ordenadas	16
Listas de definiciones	16
Las tablas	16
Formato de las celdas.....	17
Los caracteres especiales	18
Caracteres de Control:	20
Divisiones de texto en bloque	20
Marcos	21
Formularios	21
El elemento FORM	22
Elemento de entrada de datos de un formulario.....	22
Elemento input	22
Elemento select	23
Elemento textarea	23
CSS 3 (Hojas de Estilo).....	24
Formato de las hojas de estilo.....	24
Propiedades relativas o tipos de letras	25
Propiedad del texto	26

Propiedades de bloques.....	26
Otras propiedades	26
HTML5	34
Programa CGI.....	39
Publicación de páginas Web.....	39
JavaScript.....	42
Valores y Variables	42
Cajas de Diálogo	44
El método alert()	44
El método prompt().....	46
El método confirm ()	47
Operadores y Expresiones en JavaScript.....	48
Funciones	52
Pasando Argumentos a funciones	54
Retornando Valores	55
Sentencias condicionales	56
If / else.....	57
El operador condicional ?.....	57
Sentencia if/ else if	58
Switch	59
Bucles.....	60
Manejadores de Eventos	63
Introducción a los manejadores de eventos.....	63
HTML y el controlador de eventos.....	64
Manejo de un evento de ventana o marco	69
Gestión de eventos del mouse.....	73
Como utilizar los eventos del ratón.....	73
DOM (Document Object Model)	77
Nodos.....	80
Elementos de nodos	80
Nodos de texto	80
Atributos de nodos	81
Acceso directo a los nodos	82
getElementById.....	82
getElementsByTagName	83
Atributos HTML y propiedades CSS en DOM	85
getAttribute.....	85
setAttribute.....	86
Hermanos, cuerpo, primer y último hijo.....	88
Creando un blog	89
Tablas HTML en DOM	91
Clonando un nodo.....	93
Clonando una tabla con un único Id.....	95
PROGRAMACIÓN DEL LADO DEL SERVIDOR CON PHP.....	98
Secuencias de comandos del lado del Servidor Versus lado del cliente.	98

Sintaxis básica en PHP	99
Etiquetas de HTML como script	100
Tipos de datos y variables.	100
String, sintaxis con comillas simples:	100
String, sintaxis con comillas dobles:	101
Concatenación de cadenas.....	102
Integer (enteros)	103
Números de punto flotante	103
Valores Boolean.....	103
Arreglos o Matrices	103
Arreglos multidimensionales	105
Enviando datos a la salida	106
Sentencia print().....	106
Sentencia echo()	106
Sentencia printf().....	107
Sentencia sprintf()	109
Operadores	109
Operadores aritméticos.....	109
Operadores de asignación aritmética	110
Operadores de comparación.....	110
Operadores para controlar errores.....	111
Operadores de incremento / decremento.....	111
Operadores lógicos.....	111
Operadores de cadenas.....	112
Estructura de control	113
Continue.....	113
Include, include_once, require, y require_once.....	114
Goto	115
Funciones definidas por el usuario	115
Devolviendo valores desde una función.	117
Pasando información con PHP	118
Matrices Superglobales.....	118
Ámbito de aplicación de variables.....	118
\$GLOBALS	121
\$_SERVER	121
\$_GET.....	123
Codificación de URL	123
Acceso a variables de URL	123
\$_POST	125
\$_REQUEST.....	126
\$_FILES	127
\$_SESSION	130
Usando session_destroy()	131
\$_COOKIE	132
PHP Orientado a Objetos.....	134
Usando Clases y Objetos.....	135
Protección de la integridad de datos con la encapsulación.....	136
Polimorfismo	137

Herencia.....	137
Uso de PHP Zend Framework Codificación estándar	138
Creación de clases y objetos	139
Acceso a los métodos y propiedades de una clase padre	146
Uso de Filtros para validar la entrada de usuarios con PHP.....	152
Configuración de las opciones del filtro	154
Trabajando con Bases de Datos.....	158
Abriendo una conexión	158
Extensión MySQL de PHP	158
La extensión MySQLi.....	160
Uso de Sentencias preparadas.....	160
Objetos de datos PHP (PDO)	164
Ejemplo de reescritura de su DOP.....	164
Estructura de tabla y un curso acelerado en la planificación	165
Planeando tablas de bases de datos	166
XML	170
Todos los elementos XML deben tener una etiqueta de cierre.....	173
Ejemplo XMLHttpRequest:	178
XML está basado en texto	186
¿Por qué no el Bloc de notas?.....	186
¿Por qué un editor de XML?	186
XML válido.....	187
AJAX	195
Modelo Vista Controlador.....	240
Utilizando el Modelo	241
Cambiando el gestor de Base de Datos	241
Arquitectura	243
Hojas de estilo con YAML.....	244
Estructura del paquete:.....	245
¿Cómo empezar a utilizar nuestra plantilla?	245
Reglas de oro para usar YAML correctamente.....	245
El YAML Builder	246
Anexos.....	248
Conclusiones.....	299
Recomendaciones.....	300
Planificación Autoestudio.....	301
Planificación para el Docente.....	302
Bibliografía.....	303



Síntesis

En la actualidad, el Departamento de Computación no cuenta con un programa teórico-práctico de estudio adecuado, que trate sobre las tecnologías Web y que se ajuste a las necesidades de los estudiantes de la carrera de Ingeniería en Sistemas modalidad Sabatina, mediante la cual se puedan reforzar los conocimientos recibidos en las aulas de clases.

El presente trabajo monográfico consistirá en la realización de ejercicios prácticos utilizando las tecnologías Web actuales, así como una breve descripción de cada tecnología utilizando el Software respectivo con licencias privadas o libres (Open Source).

Se hará énfasis en la temporización requerida para el desarrollo de las prácticas contenidas en este documento, cada una de las cuales constará de un enunciado descriptivo de lo que se quiere alcanzar, las técnicas para lograrlo y la solución de cada práctica.

Introducción

Internet evoluciona rápidamente ofreciendo a los usuarios cada día aplicaciones, servicios y programas nuevos. Esto es producto de la necesidad de compartir recursos computacionales que a menudo son utilizados por las empresas. Al paso de la creciente demanda por interconectar empresas de diferentes ámbitos con tecnologías diferentes, está ligada al mejoramiento de las tecnologías web actuales que ofrecen mayores velocidades de conexión, entornos más amigables y mejor organización y acceso de la información.

En la actualidad, la asignatura de Programación Orientada a la Web, en la carrera de Ingeniería en Sistemas de Información en modalidad Sabatina, debido a la poca cantidad de horas impartidas (2 horas por sábado, teoría o laboratorio) no cuenta con un programa teórico-práctico que le facilite a los estudiantes profundizar sus conocimientos ni asociar las diferentes tecnologías web de acuerdo a las exigencias de las empresas modernas, lo que origina la graduación de los estudiantes con un bajo dominio del contenido impartido en la asignatura.

En este documento se presenta una miscelánea de prácticas y ejemplos que integran la implementación de lenguajes de programación orientados a la web, que les facilitaran a los estudiantes no solo mejorar su calidad y destreza en este aspecto, sino que preparará egresados con una mejor calidad profesional.

En este documento se especifica detalladamente la temporización del desarrollo de las prácticas, presentando para cada práctica un enunciado claro, objetivos que se pretenden alcanzar, el desarrollo y la solución de las prácticas, comentarios sobre los resultados, bibliografía, sugerencias y referencias al respecto.

El desarrollo de las prácticas está íntimamente relacionado con los siguientes temas:

La integración del protocolo **HTTP** y el lenguaje **HTML**, el primero nos permite enviar cualquier tipo de fichero fácilmente simplificando el funcionamiento del servidor, el otro nos proporciona un mecanismo de composición de páginas entrelazadas simples y fáciles altamente eficiente. Al crear una aplicación web con HTML (Lenguaje de Marcado de Hipertexto), se dará cuenta que no ofrece el control necesario para crear páginas web sofisticadas ni interactivas. Ahora, al combinar **JavaScript** podrá crear páginas interactivas con las características de muchos sitios web profesionales. Adquirirá los conocimientos necesarios para crear páginas web interactivas con estilos. Además se utilizará **PHP**, uno de los lenguajes de programación para desarrollo de aplicaciones web más populares en el mundo. Este respaldado por el soporte de una comunidad entusiasta de usuarios, utiliza cientos de funciones orientadas a desarrolladores, uso de captcha, bibliotecas estándar, graficación de estatus, etc. Se desarrollaran aplicaciones web con ASP.NET: estas aplicaciones crean contenidos web para los exploradores web clientes, en este contenido incluimos lenguaje de marcado de hipertexto extensible (XHTML), secuencias de comando del lado del cliente, imágenes y datos binarios, mediante el uso de formularios, controles web también llamados controles de SERVIDOR ASP.NET.

Tecnologías de programación web a utilizar:

El lenguaje de marcado HTML (HyperText Markup Language): se utiliza para crear documentos que presenten una estructura hipertexto, un documento de hipertexto es aquel que contiene información cruzada con otros documentos, la cual nos permite movernos de un documento a otro al referenciarlo desde la misma aplicación que estamos visualizando. HTML también nos permite crear documentos de tipo multimedia, es decir que contenga información más allá de la textual; por ejemplo: Imágenes, Videos, Sonidos, Sub-programas activos (plug-ins, applets).

CSS: El objetivo inicial de la web era el intercambio de información entre diversas máquinas conectadas a través de la red, por lo que se dio mayor importancia al contenido que a la forma. Sin embargo, cuando la web saltó del ámbito educativo al comercial, el diseño pasó a jugar un papel muy importante y aquí es donde los diseñadores se encontraron con las limitaciones de HTML para la composición de la página. Las hojas de estilo en cascada permiten especificar una serie de propiedades como la fuente y el tamaño del texto o la posición de los elementos en la página. Lo que es más interesante es que todas las propiedades se pueden modificar de forma dinámica.

JavaScript: Es tanto un lenguaje script como de programación, porque básicamente realizan lo mismo, es decir permite que los desarrolladores le indiquen a los navegadores que lleven a cabo una determinada acción como validar información que incorpora un usuario en un formulario. JavaScript es un lenguaje de programación orientado a objeto que permite crear aplicaciones utilizando objetos. Las aplicaciones realizadas con esta tecnología reaccionan ante eventos que ocurren mientras se ejecuta la aplicación definiendo manejadores de eventos.

PHP: Es un eficaz lenguaje de sentencias de comandos web que no ha dejado de crecer desde su lanzamiento en 1993. Admite tanta variedad de tecnología de Internet que se ha convertido en el lenguaje de secuencias de comandos preferido en Internet. Aunque el objetivo principal de la secuencia de comandos es generar contenido HTML, durante la ejecución de la secuencia se pueden ejecutar varios procesos, desde acceso a Bases de Datos hasta envíos de mensajes de correo electrónicos. Una diferencia entre este lenguaje y otras plataformas de desarrollo es que todo el código se ejecuta en el servidor, lo que implica que el cliente no necesita programas, complementos o bibliotecas especiales para ejecutar la secuencia de comandos.

DOM: Cuando se definió el lenguaje XML, surgió la necesidad de procesar y manipular el contenido de los archivos XML mediante los lenguajes de programación tradicionales. XML es un lenguaje sencillo de escribir pero complejo para procesar y manipular de forma eficiente. Por este motivo, surgieron algunas técnicas entre las que se encuentra DOM o Document Object Model es un conjunto de utilidades específicamente diseñadas para manipular documentos XML. Por extensión, DOM también se puede utilizar para manipular documentos XHTML y HTML. Técnicamente, DOM es una API de funciones que se pueden utilizar para manipular las páginas XHTML de forma rápida y eficiente.

XML: [XML significa](#) *eXtensible markup language*, o lenguaje de anotación extensible. Ya conocemos el lenguaje HTML (*hypertext markup language*), lenguaje de anotación para página webs que permite navegación tipo hipertexto; sin embargo, XML no es sólo un lenguaje, es una forma de especificar lenguajes, de ahí lo de extensible. Todo lenguaje que se exprese de una forma determinada puede ser XML. Por lo tanto, XML *no es un lenguaje para hacer mejores páginas web*, sino un lenguaje para información auto-descrita, o al menos, auto-descrita si las etiquetas están bien puestas.

AJAX: Si bien en sí mismo no es una tecnología, es una mezcla de buenos lenguajes de programación. Actualmente se recargan las páginas cuando se hace una petición al servidor, la mejoría al usar Ajax consiste en que en lugar de recargar toda la página, este trabaja en una capa extra entre los buscadores de los usuarios y los servidores web. Ajax maneja comunicación con los servidores, enviando peticiones a los servidores y procesando las respuestas. Los resultados serán integrados en el visor de la página sin tener que estar refrescando o recargando la página.

MVC y Frameworks: Modelo Vista Controlador es un patrón de arquitectura de software que se utiliza mucho en aplicaciones web, separándola en tres capas (*modelo, controlador y vista*). El fácil mantenimiento de código en un futuro, ya que al estar separadas los distintos procesos según su tipo. Si quisiéramos por ejemplo cambiar de tipo de base de datos, solo tendríamos que cambiar la capa modelo. Para una mejor percepción de esta arquitectura, muestro la transformación de una lista de artículos de un blog. Son diseñados con la intención de facilitar el desarrollo de *software*, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de *software* que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Por ejemplo, un equipo que usa [Apache Struts](#) para desarrollar un [sitio web](#) de un banco, puede enfocarse en cómo los retiros de ahorros van a funcionar en lugar de preocuparse de cómo se controla la navegación entre las páginas en una forma libre de errores. Sin embargo, hay quejas comunes acerca de que el uso de *frameworks* añade [código](#) innecesario y que la preponderancia de *frameworks* competitivos y complementarios significa que el tiempo que se pasaba programando y diseñando ahora se gasta en aprender a usar *frameworks*.



Antecedentes

En la Universidad Nacional Autónoma de Nicaragua se apertura la carrera de Ingeniería en Sistemas modalidad Sabatina en el año 2006, desde entonces, no se ha creado un plan académico para la asignatura de Programación Orientada a la WEB que le permita tanto a los estudiantes como a los docentes utilizarla como guía práctica para reforzar los conocimientos impartidos en las aulas de clases que se imparte en esta modalidad.

Este documento, consta con un desarrollo del temario, planteamiento de prácticas con sus soluciones, con el propósito de que los nuevos estudiantes tengan más herramientas disponibles con prácticas compiladas de uso real en las empresas y que le permita superar sus capacidades, al mismo tiempo enriquecer el fruto del auto estudio personal.

Al momento de impartirnos la asignatura, sentimos la necesidad de afianzar nuestros conocimientos, pero por no existir una guía de este tipo el aprendizaje se nos dificulto bastante porque debimos recurrir a investigar por diferentes medios y también resulto una limitante el corto tiempo(horas clase) que esta asignatura tiene asignada, de tal forma que para poder asimilar las lecciones de la clase y tener esta guía a la mano, representará una mejor asimilación del material impartido por los docentes.

A pesar de la buena voluntad del docente, se nos dificulto el desarrollo personal porque de la cantidad de sábados programados para la clase (40 horas aproximadamente), se redujo por problemas (choques de horarios, falta de laboratorios, falta de equipos en buen estado, problemas administrativos).

La problemática cultural de los alumnos del sábado, representa una barrera para el avance del desarrollo de las clases.

Por la falta de tiempo de los alumnos (en su mayoría trabajadores), es casi nula la posibilidad de venir a consultar al docente en días de semana, agregado a esto la ocupación del docente con los estudiantes del diario.

No se nos facilitó documentación adicional de la asignatura durante se nos impartió, esto representó una limitante al momento de estudiar y tratar de aclarar dudas con material de apoyo.



Justificación

Hasta hoy, en el Departamento de Computación no se cuenta con un trabajo teórico-práctico que trate sobre la programación con distintas tecnologías web. Es por esta razón que decidimos realizar este trabajo para apoyar al docente y a los estudiantes que cursen esta asignatura.

Es por ello que las practicas recopiladas en este documento, no son repeticiones de ejemplos que frecuentemente los estudiantes leen. A través de estas prácticas se pretende crear una asimilación consistente de los conocimientos que se les imparten a los alumnos. Las prácticas están referidas a situaciones y problemáticas reales que se viven en las empresas y que deben ser resueltas utilizando las tecnologías web actuales.

Se utilizaran programas de distribución libre y software propietarios, para que se vea la aplicación de la teoría impartida en clases. El conjunto de software a utilizar para la solución de las prácticas son: Xampp, Eclipse, DreamWeaver, Notepad++, Expression Web, Mozilla Firefox.

Al finalizar el estudio de este trabajo, tanto la parte teórica como la parte práctica, los estudiantes tendrán un nivel de dominio de las tecnologías contenidas en este documento que le permitirán abrirse paso en el mercado profesional, además de aumentar la calidad de los egresados de la carrera en esta modalidad.



Objetivo General

- ✓ Realizar un programa de prácticas resueltas usando diferentes tecnologías web actuales y documentar la realización de dichas prácticas para la asignatura Programación orientada a la Web.

Objetivos Específicos

- ✓ Documentar con ejemplos prácticos, las características de los lenguajes de marcado y de programación orientados a la web actuales y usados en este trabajo.
- ✓ Proponer un formato de prácticas para el desarrollo de las mismas.
- ✓ Desarrollar un programa de prácticas de laboratorio y resolver dichas prácticas documentando debidamente la solución.

Marco Teórico

Introducción a Internet

Hoy en día Internet –también conocida como “la red”- supone un excelente medio para obtener información de los más variados temas a cualquier hora del día y sin necesidad de moverse de casa. Así, se pueden conocer las noticias de última hora, ver el tráiler del próximo estreno del cine, visitar lugares lejanos, reservar los billetes de avión para vacaciones, contactar con personas de todo el mundo o comprar cualquier cosa que a uno se le pueda ocurrir.

Esto se debe a que la información disponible en Internet es casi ilimitada, y aumenta día a día. Las empresas vieron en la red primero un medio para anunciarse, actualmente, una vía para atender a sus clientes y ofrecer sus productos. Muchas instituciones públicas y privadas las utilizan para dar a conocer sus actividades y publicar datos de interés general o de un tema específico. Y los particulares disponen de un número creciente de servicios accesibles a través de Internet.

Para que toda esta montaña de información sea realmente útil es imprescindible que el acceso a ella sea simple e intuitivo, de forma que cualquier persona pueda conocer y utilizar la información que desea con tan solo unos conocimientos básicos.

Protocolos de Internet

Físicamente, Internet (Interconnected Networks) es una red compuesta por miles de redes independientes pertenecientes a instituciones públicas, centros de investigación, empresas privadas y otras organizaciones.

De estos proyectos nació un protocolo de comunicación de datos, IP o Internet Protocolo, que permitía a ordenadores diverso conectarse a través de una red, Internet, formada por la interconexión de diversas redes.

Estas redes comparten unas normas que aseguran la comunicación entre ellas: son los denominados **Protocolos de Internet**. Un protocolo es un conjunto de normas que permiten el intercambio de información entre máquinas de diversos tipos conectadas entre sí.

Existen diversos tipos de protocolos en función del tipo de comunicación que se establece entre las máquinas: **FTP** (para la transferencia de ficheros), **POP** y **SMTP** (para el envío y recepción de correo electrónico), **TELNET** (para la conexión con terminales remotos), **GOPHER**, **WAIS** y **HTTP** (para el acceso a servidores de información).

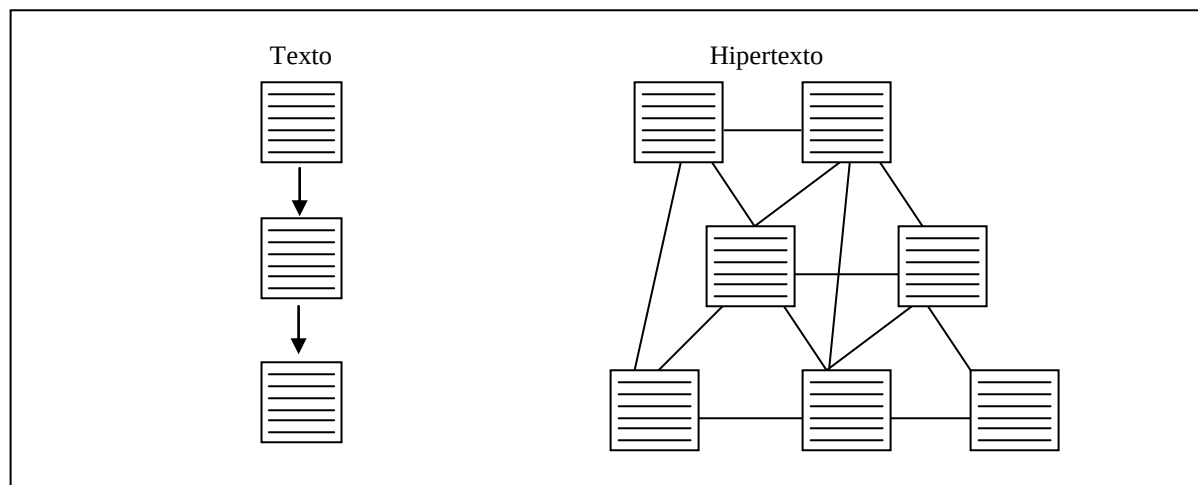
El protocolo **HTTP** (Hyper Text Transfer Protocol) es la base de la World Wide Web o telaraña mundial, abreviada **WWW** y más conocida como Web. La Web es un sistema que facilita el acceso a los distintos recursos que ofrece Internet a través de una interfaz común basada en el **hipertexto**. La Web se ha hecho enormemente popular con la aparición de unos programas dotados de interfaces gráficas amigables que permiten a cualquier persona acceder a los documentos de la red: son los conocidos navegadores.

A mediados de los ochentas la Fundación Nacional para la Ciencia norteamericana, la NFS, creó una red, la NSFNET, que se convirtió en el backbone (el troncal) de Internet junto con otras redes creadas por la NASA (NSINet) y el U.S.DoE (Departament of Energy con ESNET).

A partir de esta época, gracias entre otras cosas a la amplia disponibilidad de implementaciones de la suite de protocolos TCP/IP (formado por todos los protocolos de Internet y no solo por TCP e IP), alguna de las cuales eran ya de código libre, Internet empezó lo que posteriormente se convertiría en una de sus características fundamentales, un ritmo de crecimiento exponencial.

El hipertexto

El hipertexto es un texto con enlaces. Los documentos de hipertexto pueden contener enlaces, que son referencias a otras partes del documento o a otros documentos. De esta forma, el documento no tiene por qué ser leído secuencialmente, sino que en ciertos puntos se puede saltar a otra parte del documento y desde allí volver al punto original o bien saltar a otro punto distinto.



El hipertexto supone una aplicación del concepto habitual de texto al permitir que una serie de documentos enlazados unos con otros y posiblemente ubicados en lugares remotos aparezcan formando una unidad.

Las ideas básicas de la Web son tres:

- La desaparición de la idea de un servidor único de información mantenido por un equipo de profesionales dedicado. Cualquiera puede crear sus propios documentos y referenciar en ellos a otros documentos. No existe una autoridad central en la Web.
- La especificación de un mecanismo para localizar de forma unívoca documentos distribuidos geográficamente. Cada documento tiene una dirección llamada Uniform Resource Locator, URL. Esta dirección indica no solo la localización del documento, sino también el mecanismo para acceder a él (es decir, el protocolo, que en el caso de una página Web es el HTTP).
- La existencia de una interfaz de usuario uniforme que esconde los detalles de los formatos y protocolos utilizados para la transferencia de la información, simplificando por tanto el acceso a la misma. Los programas que utilizan estas interfaces son denominados examinadores (browsers), visualizadores o, más comúnmente, navegadores.

El elemento básico de la Web es la **página Web**, un documento que contiene enlaces de hipertexto a otros documentos relacionados que pueden estar localizados en la propia máquina o en máquinas remotas. De esta forma, siguiendo los enlaces se puede navegar por la red visitando páginas ubicadas en distintos lugares como si formaran una sola unidad.

El protocolo de la Web especifica un formato para escribir los documentos que forman parte de ella: es el Hypertext Markup Language (HTML).



Funcionamiento de la WEB

La web funciona siguiendo el denominado modelo cliente-servidor, habitual en las aplicaciones que funcionan en una red. Existe un servidor, que es quien presta los servicios, y un cliente, que es el que lo recibe.

Cliente WEB

El cliente web es un programa con el que el usuario interacciona para solicitar a un servidor web el envío de páginas de información. Las páginas que se reciben son documentos de textos codificados en HTML. El cliente web debe interpretar estos documentos para mostrárselos al usuario en el formato adecuado

Entre los clientes web (también conocidos como visualizadores y navegadores) más usuales están, el Internet Explorer, El Mozilla, El Opera. La mayoría de ellos soporta también otros protocolos como el **FTP**, y el **SMTP**.

Servidor WEB

El servidor web es un programa que está escuchando permanentemente las peticiones de conexión de los clientes mediante el protocolo **HTTP**. Funciona de la siguiente manera: Si encuentra en su sistema de su fichero el documento HTML solicitado por el cliente lo envía y cierra la conexión en caso contrario envía un código de error y cierra la conexión. El servidor web también se ocupa de controlar los aspectos de seguridad, comprobando si el usuario tiene acceso a los documentos.



Capítulo 1

ASPECTOS FUNDAMENTALES DE HTML y CSS.

1 ASPECTOS FUNDAMENTALES DE HTML

1.1 HTML Básico.

HTML: Es el lenguaje de marcado de hipertexto, es el lenguaje en el que se escriben las páginas Web. Este lenguaje se utiliza para decirle al navegador como debe ser presentado un documento (letra, tamaño, colores, etc.).

A lo largo de la historia, han existido diferentes versiones de HTML y actualmente nos encontramos en la versión HTML 4.0 aunque, se ha publicado una especificación llamada XHTML que define de manera restrictiva, como deben de ser escritos los documentos.

El lenguaje HTML se basa en la sintaxis SGML (toma siglas). Esto quiere decir que todo estará encerrado entre dos etiquetas de esta manera:

<etiqueta parámetros>... </etiqueta>

Hay ocasiones en las que no es necesario cerrar las etiquetas.

Todo código HTML está encerrado entre las etiquetas **<HTML>...</HTML>**

Un documento HTML tiene una estructura que lo separa en dos partes: cuerpo y cabecera. En la primera estará la página en sí, mientras que en la segunda incluiremos algunas cosas que no se ven al principio pero que llegan a ser muy importantes.

Los documentos HTML se conforman como documentos de texto plano (sin ningún tipo de formato especial), en los que todo el formato del texto se especifica mediante marcas de texto (llamados etiquetas, tags), que delimitan los contenidos a los que afecta la etiqueta (dispone de etiquetas de inicio y final de marcado).

1.2 Estructura de los documentos HTML

Todos los documentos HTML siguen aproximadamente la misma estructura. Todo el documento debe ir contenido en una etiqueta HTML, dividiéndose en dos partes: la cabecera, contenida en una etiqueta HEAD y el cuerpo del documento (donde está la información del documento), que está envuelto por una etiqueta BODY. La cabecera contiene algunas definiciones sobre el documento: su título, marcas extra de formato, palabras clave, etc.

Un primer ejemplo seria:

```
<html>
  <head>
    <title>Título del documento</title>
  </head>
  <body>
    Texto del documento
  </body>
</html>
```

1.3 La cabecera en HTML

```
<html>
  <head>
    <title> Mi primera página </title>
```

```
</head>  
<html>
```

En este caso, dentro de la cabecera solo hay otra etiqueta. Es la única imprescindible: el título de la página. Es lo que veremos como título de la ventana en los navegadores que lo permitan.

Otras etiquetas a usar dentro de la cabecera son:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8 />
```

1.4 El cuerpo en HTML

El cuerpo es todo lo que se encuentra encerrado dentro de las etiquetas:

```
<body> </body>
```

Luego pondremos el título algo recalcado:

```
<center><h1>...</h1></center>
```

Con esto colocaremos el texto centrado (<center>) y en formato <h1> (cabecera 1) que nos asegura que aumentara el tamaño del tipo de letra lo suficiente como para que se vea bastante resaltado. Luego separamos ese título que le hemos puesto a la página del texto por medio de una línea horizontal:

<hr>: La línea horizontal carece de etiqueta de cierre. Esto es normal en etiquetas que no varían los atributos de un texto, sino que insertan un elemento.

BACKGROUND	Permite definir un gráfico de fondo para la página.
BGCOLOR	Permite definir el color de fondo de la página.
BGPROPERTIES	Cuando es igual a FIXED el gráfico definido como fondo de la página permanecerá inmóvil aunque utilicemos las barras de desplazamiento.
TEXT	Cambia el color del texto.
LINK	Cambia el color de un enlace no visitado (por defecto azul).
VLINK	Cambia el color de un enlace ya visitado (por defecto púrpura).
ALINK	Cambia el color que toma un enlace mientras lo estamos pulsando (por defecto rojo).
LEFTMARGIN y TOPMARGIN	Especifican el número de píxeles que dejara de margen entre el borde de la ventana y el contenido de la página. Se suelen utilizar para dejarlos a cero.
MARGINWIDTH y MARGINHEIGHT	Más o menos equivalentes a los anteriores, pero estos funcionan en Netscape.

Tabla 1.1 Propiedades de la etiqueta body.

1.5 Los encabezados

HTML ofrece 6 etiquetas distintas para mostrar encabezados. Cada uno de ellos se diferencia en el tamaño de las cadenas de caracteres.

```
<html>  
<head>
```

```
<title>Documento sin título</title>
</head>
<body>
  <h1>Encabezado 1</h1>

  <h2>Encabezado 2</h2>

  <h3>Encabezado 3</h3>

  <h4>Encabezado 4</h4>

  <h5>Encabezado 5</h5>

  <h6>Encabezado 6</h6>

</body>
</html>
```

Encabezado 1

Encabezado 2

Encabezado 3

Encabezado 4

Encabezado 5

Encabezado 6

1.6 Estilos de párrafos: p, center, br, hr y blockquote

Etiqueta HTML	Ejemplo de uso
<code><p>....</p></code>	Se utiliza para declarar un párrafo con lo cual se da un espacio al inicio y al final del contenido.
<code><center>....</center></code>	Centra en pantalla lo que sea que se encuentre entre las dos etiquetas.
<code>
</code>	Se utiliza para insertar una línea en blanco. Como un enter.
<code><hr /></code>	Se utiliza para insertar una línea en blanco.
<code><blockquote>....</blockquote></code>	Se utilizan para tabular el contenido que se encuentre entre las dos etiquetas.

Tabla 1.2 Etiquetas que se utilizan para dar un formato a los párrafos.

1.7 Formato de texto: B, I, U, SUB, SUP

Etiqueta HTML	Ejemplo de uso
<code>....</code>	Pone el texto en negrita.
<code><i>....</i></code>	Pone el texto en cursiva.
<code><u>....</u></code>	Pone el texto subrayado.
<code><sub>....</sub></code>	Pone el contenido como sub-índice.
<code><sup>....</sup></code>	Pone el contenido como súper-índice.

Tabla 1.3 Etiquetas para el formato de texto.

1.8 Formato de texto: Font

Permite cambiar el tamaño color y tipo de letra del texto que va entre esta etiqueta.

Parámetro	Utilidad
Color	Permite cambiar el color: <code>Color por nombre </code> <code>Color RGB</code>

Size	Permite cambiar el tamaño: <code>Tamaño 2</code>
Face	Permite cambiar el face: <code>Con otro tipo de letra</code>

Tabla 1.4 Parámetros que se utilizan para cambiar color, tamaño y fuente.

También se pueden utilizar los tres parámetros de manera conjunta:

`<font color="#993366" size="6" face="Courier New, Courier, monospace">` Esto es un texto con una fuente diferente `</font>`

1.9 La etiqueta `<a>`:

Todo lo que encerramos entre `<a>` y `</a>`, ya sea texto o imágenes, será considerado como enlace y sufrirá dos modificaciones:

-Se visualizara de manera distinta en el navegador. El texto aparecerá subrayado y de un color distinto al habitual, y las imágenes estarán rodeadas por un borde del mismo color que el del texto del enlace.

-Al pulsar sobre el enlace, seremos enviados al documento que apuntaba el enlace. Para que el enlace sirva para algo debemos especificarle una dirección. Lo haremos de la siguiente manera:

`<a ref="dirección">Púlsame</a>` La dirección estará en formato URL.

1.10 Los enlaces:

Un enlace es una zona de texto o gráficos que si son seleccionados nos trasladan a otro documento de hipertexto o a otra posición del documento actual.

Una URL nos indica tanto una dirección de Internet como el servicio que esperamos nos ofrezca el servidor al que corresponde la dirección. Tiene el siguiente formato:

Servicio://maquina:puerto/ruta/fichero

Donde el servicio podrá ser uno de los siguientes: http, https, ftp, mailto, etc.

Ejemplo: `<a ref="http://localhost/primer.html">`

Existe otro formato de URL. Cuando queremos acceder a un fichero situado en la misma máquina que la página Web que estamos creando podemos utilizar este formato:

`<a href="primera.html">`

1.11 La inserción de imágenes con `img`

Se utiliza para incluir imágenes de la siguiente manera:

``

- src: Especifica el nombre del fichero que contiene el grafico. Los formatos estándar en la red son el GIF y el JPG.
- alt: Es el texto que se mostrara cuando el cursor este sobre la imagen.

1.12 Imágenes y Enlaces

Suele ser común incluir enlaces dentro de un gráfico. En ese caso, se le pondrá un borde al grafico que se puede quitar poniendo la propiedad border="0".

```
<a href="01-primer.html"></a>
```

Valor de ALIGN	Utilidad
Top	Coloca el punto más alto de la imagen coincidiendo con el más alto de la línea de texto.
Middle	Alinea el punto medio (en altura) de la imagen con la base del texto.
Bottom (defecto)	Alinea el punto más bajo de la imagen con la base del texto.
Left	Coloca la imagen a la izquierda del texto.
Right	Coloca la imagen a la derecha del texto.

Tabla 1.5 Propiedad align para alinear la imagen.

Por ejemplo:

```
<a href="http://www.w3c.org">Enlace</a>

```

Cabe destacar que HTML permite omitir en algunos casos la etiqueta de final si la etiqueta no debe envolver ningún texto al que afecte, como es el caso de IMG. Otro punto destacable es que si el cliente WWW que utilizemos (el navegador concreto que estemos usando) no entiende alguna etiqueta, la ignorará y hará lo mismo con todo el texto afectado por ésta.

1.13 Las listas desordenadas

La etiqueta nos permite presentar listas de elementos sin orden alguno.

```
<ul>
  <li>Primer elemento      * Primer elemento.
  <li>Segundo elemento     * Segundo elemento.
</ul>
```

Parámetro	Utilidad	Resultado
COMPACT	Representa la lista de la manera más compacta.	• Primer elemento. • Segundo elemento.
TYPE="disc","circle","square"	Indica al navegador el dibujo que precederá a cada elemento de la lista. Para mayor flexibilidad se admite también como parámetro de 	• Tipo disc o Tipo circle. • Tipo square.

Tabla 1.6 Parámetros que admite la etiqueta .

1.14 Las Listas ordenadas

La etiqueta **** nos permite presentar listas de elementos ordenados de menor a mayor anteponiéndole un número:

****Primer elemento.

1. Primer elemento.

****Segundo elemento.

2. Segundo elemento.

Parámetro	Utilidad	Resultado
TYPE="1","a","A","i","I"	Indica el tipo de numeración que precederá a cada elemento de la lista.	1. Tipo 1 a. Tipo a A. Tipo A i. Tipo i I. Tipo I
START="num"	Indica al navegador el numero por el que se empezara a contar os elementos de la lista.	3. Primer elemento. 4. Segundo elemento.
VALUE="num"	Atributo de , actúa como START pero a partir de un elemento.	1. Primer elemento. 4. Segundo elemento. 1. Tercer elemento.

Tabla 1.7 La etiqueta **** admite estos parámetros:

1.15 Listas de definiciones

Este es el único tipo de lista que no utiliza la etiqueta ****. Al tener como objetivo presentar una lista de definiciones, de modo que tiene que representar de manera distinta el objeto definido y la definición. Esto se hace por medio de las etiquetas:

<dt> y **<dd>**:

<dl>

<dl>Primer elemento**<dd>**Es un elemento muy bonito.

<dt>Segundo elemento**<dd>**Este, en cambio, es peor.

</dl>

Se verá como:

Primer elemento

Es un elemento muy bonito.

Segundo elemento

Este, en cambio, es peor.

1.16 Las tablas

Las tablas son posiblemente la manera más clara de organizar la información. También es el modo más adecuado de maquetar texto y gráficos de una manera algo más controlada que con los parámetros **align**.

<table>

<tr>

<td>1,1**</td>**

<td>1,2**</td>**

<td>1,3**</td>**

</tr>

```

<tr>
  <td>2,1</td>
  <td>2,2</td>
  <td>2,3</td>
</tr>
</table>

```

Se verá así:

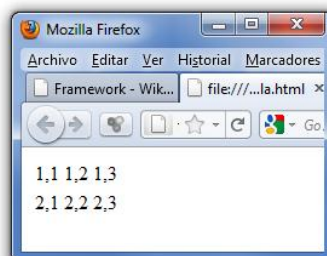


Ilustración 1.1 Tabla mostrada por el navegador.

Border	Especifica el grosor del borde que se dibujara alrededor de las celdas. Por defecto es cero, lo que significa que no dibujara borde alguno.
Cellspacing	Define el número de pixeles que separaran las celdas.
Cellpadding	Especifica el número de pixeles que habrá entre el borde de una celda y su contenido.
Width	Especifica la anchura de la tabla. Puede estar tanto en pixeles como en porcentaje de la anchura total disponible para el (pondremos "100%" si queremos que ocupe todo el ancho de la ventana del navegador).
Align	Alinea la tabla a izquierda (left), derecha (right) o centro (center).

Tabla 1.8 Propiedades de tablas:

1.16.1 Formato de las celdas

Las celdas por concreto, también poseen ciertas propiedades que permiten trabajar de mejor manera con ellas:

Align	Alinea el contenido de la celda horizontalmente a la izquierda (left), derecha (right) o centro (center).
Valign	Alinea el contenido de la celda verticalmente arriba (top), abajo (bottom) o centro (middle).
Width	Especifica la anchura de la celda. También se puede especificar tanto en pixeles como en porcentaje, teniendo en cuenta que, en este último caso, será un porcentaje respecto al ancho total de la tabla (no de la ventana del navegador).
Nowrap	Impide que, en el interior de la celda, se rompa la línea en un espacio.
Colspan	Especifica el número de celdas de la fila situadas a la derecha de la actual que se unen a esta (incluyendo la celda en que se declara este parámetro). Es por defecto uno. Si se pone igual a cero, se unirán todas las celdas que queden a la derecha.
Rowspan	Especifica el número de celdas de la columna situadas debajo de la actual que se unen a esta.

Tabla 1.9 Propiedades de Celdas

1.17 Los caracteres especiales

Existen algunos caracteres especiales que para representarlos se debe hacer de manera especial: **&.....;**

Código	Resultado
&aacute;,, &Aacute;,, &eacute;,, &Eacute;,,...	á, Á, é, É, í, Í, ó, Ó, ú y Ú
ñ y ñ	ñ y ñ
&iquest;	¿
&iexcl;	¡
&ordm;	º
&ordf;	ª
&trade; o &#153;	™ o ™
&copy;	©
&reg;	®
&nbsp;	Espacio en blanco que no puede ser usado para saltar de línea.

Tabla 1.10 Caracteres especiales.

Ejemplo 1.1 Utilización de cabecera, encabezados, formato de texto, enlaces, etiqueta <a>, Imágenes como enlaces, listas y tablas.

```
<html>

<head><title> Figura 1 </title></head>

<body background="Menu.gif">

  <hr color=blue>

  <a name="aquí">

    <center><strong><font color=blue SIZE="+2"> Introducci&oacute;n a HTML </font></strong></center>

    <hr color=blue>
    <p align="left"><font size="+1"> HTML (HyperText Markup Language - Lenguaje para hipertexto) es un
    lenguaje utilizado para desarrollar Páginas y documentos Web. A diferencia de otros lenguajes
    convencionales, HTML utiliza una serie de etiquetas especiales intercaladas en un documento de texto sin
    formato. Dichas etiquetas serán posterior mente interpretadas por los exploradores encargados de visualizar
    la página o el documento Web con el fin de establecer formato.</font></p>

    <hr width="100%" size="10%" color="noshade">

  <table border="0" width="100%">

    <tr><td>
      <center><table border="5" width="95%">

        <caption>Titulares </caption>

        <tr> <th> Etiqueta </th> <th> Aplicaci&oacute;n </th> </tr>
        <tr> <td align="center"> H1 </td> <td> &#60;H1&#62; Titular nivel 1&#60;/H1&#62; Se verá as&iacute; </td> </tr>
        <tr> <td align="center"> H2 </td> <td> <h2> Titular nivel 2 </h2></td> </tr>
        <tr> <td align="center"> H3 </td> <td> <h3> Titular nivel 3 </h3></td> </tr>
        <tr> <td align="center"> H4 </td> <td> <h4> Titular nivel 4 </h4></td> </tr>

      </table>
    </td>
  </tr>
</table>

</body>
</html>
```

```
<tr> <td align="center"> H5 </td> <td> <h5> Titular nivel 5 </h5></td> </tr>
<tr> <td align="center"> H6 </td> <td> <h6> Titular nivel 6 </h6></td> </tr>
</table></center></td>

<td>
  <center><table border="5" width="100%">

    <caption>Listas </caption>

    <tr> <td align="center"> De definici&ocute;n </td> <td align="center"> Regular o
    Numeradas</td> <td align="center"> Sin numerar </td> </tr>
    <tr> <td> <dl> <dt>Elementos de HTML <dd>Estructura <dd>Titulares <dd>Realizado de
    Texto <dd>...
    <dt>Listas <dd>De Definici&ocute;n <dd>Numeradas <dd>Sin Enumerar <dd>... </dl></td>
    <td> <dl>Lenguajes de Programaci&ocute;n </dl> <ol> <li>PHP <li>JavaScripts <li>Ajax
    <li>Xajax <li>ASP.NET
    <li>jQuery <li>Android <li>... </ol> </td>
    <td> <dl>Versiones de Linux </dl> <ul> <li>Ubuntu <li>Kubunu <li>Xubuntu <li>Mandrake
    <li>DeBian <li>Suse
    <li>Solaris <li>... </ul> </td> </tr>
    <tr> <td colspan="2">Enlaces: <br>
    <br> Dentro de la misma p&aacute;gina.
    <br> A otras páginas de nuestro sitio web.
    <br> A páginas de otros sitios <br>.
    </td>

  <td>

  <br><a href="#aquí" target=_blank>Enlace hacia Arriba</a><br>
  <a href="http://localhost/html/index.html" target=_blank>Otra parte del Sitio</a><br>
  <a href="http://www.google.com" target=_blank>Enlace hacia Google</a>
  </td></tr>
</table></center>

</td></tr>
</table>
<hr width="100%" size="10%" color="noshade">
<br><caption>Inserci&ocute;n de imágenes: </caption>
<br><center>


</center>
</body>

</html>
```



Ilustración 1.2 Uso de cabecera, encabezados, formato de texto, enlaces, etiqueta <a>, Imágenes de enlaces, listas y tablas.

1.18 Caracteres de Control:

En el HTML existen cuatro caracteres de control, que se usan para formar etiquetas, establecer parámetros, etc. Para poder emplearlos sin riesgo deberemos escribir los siguientes códigos:

Código	Resultado
<	<
>	>
&	&
"	"

Tabla 1.11 Caracteres de Control.

1.19 Divisiones de texto en bloque

El elemento <div> permite dividir el texto en bloque, insertando entre los bloques un salto de línea simple.

Las alineaciones soportadas por <div>, mediante el parámetro <align>, son los siguientes; <left>, <right> y <center>.

El elemento <div> se puede utilizar para aplicar un mismo estilo a varios elementos distintos. Para ello se crea una división con estos elementos y se le asigna un estilo.

1.20 Marcos

Las páginas Web se pueden dividir en varias zonas independientes de forma que en cada una de ellas se muestre una información distinta. Estas zonas se denominan **MARCOS** (FRAME en inglés). Un marco puede contener un menú que será siempre visible aunque cambie en contenido del resto de la página.

En un documento con MARCOS, el elemento BODY se sustituye por el elemento FRAMESET, que especifica la forma en que se divide la página. Dentro de este elemento se encuentran tantos elementos FRAME como MARCOS contenga la página. Cada elemento FRAME hace referencia a un documento HTML que se mostrara en el MARCO correspondiente.

Ejemplo de un Marco horizontal y uno Vertical:

```
<frameset rows="20%, 80%">
  <frame src="hori1.html" nameE="Titulo" scrolling="NO">
  < frame cols="25%, 75 %">
    < frame src ="verti1.html" name="Menu" scrolling="NO">
    < frame src =" verti2.html" name="Info" scrolling="Auto">
</frame> </frame>
```

1.21 Formularios

Un formulario permite el envío de información desde el cliente hacia el Servidor Web. A través de una serie de controles, el usuario puede introducir información que es enviada hacia el Servidor para su procesamiento.

El funcionamiento de un formulario Web es el siguiente: La información introducida por el usuario es codificada y enviada hacia el servidor. Este puede procesar directamente la información o bien pasarla a un programa (Denominado Programa CGI, Common Gateway Interface) que la procesa y produce unos resultados que devuelve al servidor, quien a su vez las envía al cliente.

Los formularios tienen numerosas aplicaciones, como realizar consultas a Bases de Datos, Rellenar cuestionarios, solicitar información sobre algún tema, apuntarse en una lista de correo o crear un libro de visitas para una página.

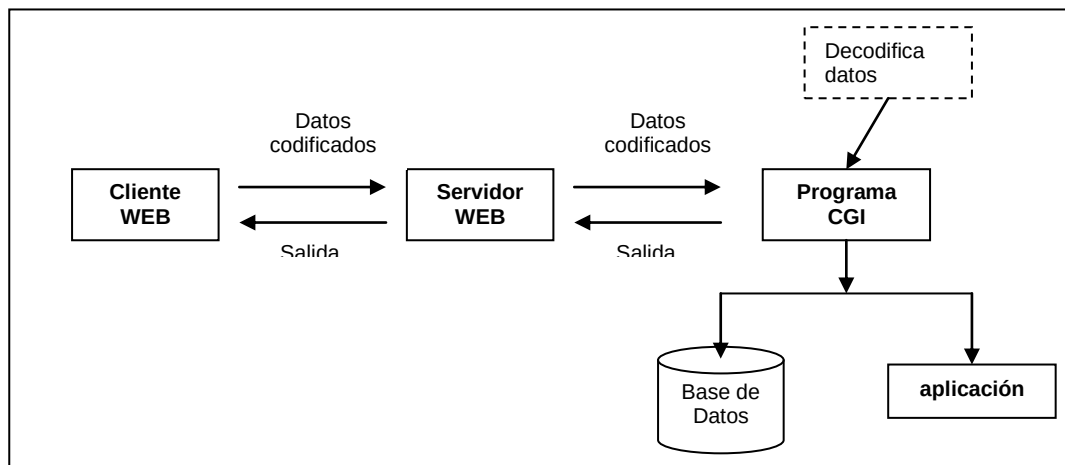


Ilustración 1.3 Flujo de datos entre el cliente web, el servidor web y el programa CGI.

1.21.1 El elemento FORM

Permite crear un formulario dentro de un documento HTML, sus atributos son los siguientes:

ACTION:	URL del programa encargado de procesar la información recogida en el formulario.
ANCTYPE:	Tipo de codificación empleada para enviar la información.
METHOD:	Método utilizado para enviar los datos, existen dos GET y POST.
TARGET:	Nombre del marco donde se va a visualizar la información devuelta por el servidor.

El atributo más importante es el ACTION ya que especifica el programa encargado de procesar el formulario.

Dentro del elemento FORM se colocaran los elementos para la entrada de datos. Entre ellos debe aparecer necesariamente un botón de envío del formulario.

1.21.2 Elemento de entrada de datos de un formulario

Los elementos que se pueden utilizar para introducir datos en un formulario son los siguientes:

- input
- select
- textarea

Cada elemento de entrada tiene un nombre, dado por el atributo name y un valor dado por el atributo value. Tanto el nombre como el valor se enviarán al servidor pulsando el botón de envío del formulario. Si el atributo value no está presente se tomara como valor del elemento introducido por el usuario.

1.21.3 Elemento input

Especifica varios tipos de controles para la entrada de datos. El tipo se especifica con el atributo TYPE que puede tomar los siguientes valores:

- **BUTTON:** Se utiliza únicamente para invocar un script contenido en el documento.

Sintaxis: `<input type="button" name="boton" value="Púlsame">`

- **CHECKBOX** (caja de verificación): El recuadro puede estar activado (ON) o desactivado (OFF), por defecto esta desactivado, a no ser que aparezca el atributo CHECKD (que carece de valor). El atributo VALUE establece el valor asignado a la caja de verificación cuando está activado; si no existe se toma el valor por defecto ON. Pueden existir varias cajas de verificación con el mismo nombre, con lo cual se asignan distintos valores a un mismo nombre de elemento.
- **FILE:** Lista de selección de fichero. Permite al usuario seleccionar un fichero para enviarlos al servidor.

Sintaxis:

```
<form action="URL del CGI" method="POST" enctype="multipart/form-data">
```

```
<input type="file" name="ficheros">
```

```
</form>
```

El método de envío debe tomar el valor POST y además debe añadirse el atributo ENCTYPE="multipart/form-data" al elemento form. Es responsabilidad del programa encargado de procesar el formulario al realizar un tratamiento adecuado del fichero enviado, tratamiento que generalmente consiste en comprobar si el fichero se ha recibido correctamente.

- **HIDDEN:** Elemento oculto, este elemento de entrada no se muestra al usuario aunque su contenido sea enviado al servidor. Es útil para pasar información entre el cliente y el servidor, y se utiliza típicamente para guardar el estado de la interacción cliente-servidor.
- **IMAGE:** Imagen activa, es similar al mapa sensible del lado del servidor.
- **PASSWORD:** Línea de texto en la que en lugar del texto tecleado aparecen asteriscos.
Clave de acceso: `<input type="password" name="clave">`
- **RADIO:** Botón de radio. Los botones de radio tienen sentido cuando hay múltiples botones con el mismo nombre pero todos están enlazados, de manera que al activar uno de ellos se desactivan los demás.
`<input type="Radio" NAME="boton" value="Opcion1" checked>Opción 1`
`< input type ="Radio" NAME="boton" value ="Opcion2" >Opción 2`
`< input type ="Radio" NAME="boton" value="Opcion3" >Opción 3`
- **RESET:** Botón de reset. Borra todos los datos introducidos en el formulario y restablece los valores por defecto.
`<input type="Reset" name="Borrar" value="Borrar formulario" >`
- **SUBMIT:** Botón de envío. Al pulsarlo se envían los datos del formulario al URL especificado en el atributo ACTION del elemento form.
`<input type="Submit" name="Enviar" value="Enviar formulario" >`
- **TEXT:** Caja de texto. El atributo SIZE opcional permite indicar su tamaño indicado en un número de caracteres.

`<input type="Text" name="Texto" value="Valor por defecto" size="20">`

1.21.4 Elemento select

Select muestra un menú desplegable con una lista de valores y permite seleccionar uno de ellos. Los valores se especifican mediante elementos OPTION que se colocan dentro del elemento select.

```
<select name="Nombre">
  <option selected>Opción 1
  < option >Opción 2
  < option >Opción 3
  < option >Opción 4
</select>
```

1.21.5 Elemento textarea

Textarea muestra una ventana en la que se puede escribir cualquier texto. El tamaño de la ventana se especifica con los atributos rows y cols, el texto puede tener un tamaño limitado y no pasa a la siguiente línea al llegar al final de la ventana, siendo necesario para ello un retorno de carro manual.

```
<textarea cols="50" ROWS="5" NAME="Nombre">
  El contenido inicial del elemento textarea es el valor inicial del texto.
</textarea>
```


1.22 CSS 3 (Hojas de Estilo)

Las hojas de estilo son un mecanismo para separar el formato de representación y presentación del contenido. Esto se consigue asociando atributos de presentación a cada una de las etiquetas de HTML o subclases de estas.

Para especificar estilos en una página, disponemos de la etiqueta `style`, que nos permite especificar in situ, y otra etiqueta `link` que nos permite indicar un fichero externo que contendrá nuestros estilos.

La etiqueta `style` debe estar en la cabecera de la página. El cual dispone de un parámetro `type` que nos permite indicar que sintaxis usaremos para definir los estilos, que en nuestro caso será `text/css`.

Estilo Genérico

Si se quiere aplicar un estilo a todas las páginas de un elemento y no a una sola se utilizan estilos genéricos. Los estilos genéricos se definen mediante el elemento `style` por ejemplo:

```
<style type="text/css">
  p {color: red; background: yellow ;}
</style>
```

La etiqueta `link` para definir una hoja de estilo externa tiene la siguiente sintaxis:

```
<link rel="stylesheet" href="miweb.css" type="text/css">
```

De hecho, es muy recomendable la utilización de la etiqueta `link` para la definición de las hojas de estilo asociadas a una página, ya que de este modo facilitaremos el mantenimiento, al concentrarse en un único fichero, todos los estilos de un sitio web.

1.22.1 Formato de las hojas de estilo

El formato de las hojas de estilo es el siguiente:

```
P {color: blue;}
```

```
< elemento HTML> {<formato>}
```

Cabe destacar que la sintaxis de CSS es sensible a mayúsculas y minúsculas. Esta sintaxis nos permitirá definir el formato que deseamos para los párrafos de nuestra web. Existe una extensión de dicha sintaxis que hace definir un estilo que solo se aplicará a ciertas partes del documento. En concreto nos permite definir clases de elementos a los que aplicaremos el estilo.

Por ejemplo, para definir una clase de párrafo que llamaremos prueba:

```
p.prueba {color:blue;background:yellow;}
```

Podemos usar luego el atributo `class` para definir la clase de párrafo.

```
<p class="prueba"> Un párrafo destacado </p>
```

Existe más de un método para asignar un estilo a párrafos individuales, aportando así mayor granularidad al concepto de clases. Para ello deberemos definir el estilo de un elemento individual de HTML mediante CSS usando la siguiente sintaxis:

```
#Paradox 1 {color: green; Background: yellow;}
```

A continuación, podemos otorgar a algún elemento de HTML dicha identidad mediante el atributo id:

```
<p class="destacado" > Un párrafo destacado </p>
```

```
<p> Un párrafo normal </p>
```

```
<p class="destacado" id="Párrafo 1"> Otro destacado pero el color en este viene dado por su identidad </p>
```

Etiquetas span y div

En algunas ocasiones es interesante asignar estilos a bloques de texto o de contenido que no formen parte de ningún bloque de HTML. La solución viene de la mano de la etiqueta div y span. Si queremos marcar un bloque de contenido como perteneciente a una clase concreta para poder definir un estilo para él o darle una especificación individual, debemos envolver dicho contenido con una etiqueta span o div. La diferencia entre usar una u otra consiste en que div se asegura de que tanto al inicio del bloque como al final haya un salto de línea.

Así podemos definir estilos para bloques de texto sin que tengamos que envolverlos con etiquetas que modifiquen su formato.

Por ejemplo:

```
<style type="text/css">
  all.dudoso
  {
    color: red;
  }

  all.revisado
  {
    color:blue;
  }
</style>
```

Y lo usaremos en nuestro documento HTML.

```
<p> Este párrafo <span class="dudoso"> largo </span> deberá ser corregida por el <span class="revisado">director general </span> </p>
```

Propiedades más importantes

Debido a las incompatibilidades existentes entre los diferentes navegadores es recomendable probar nuestras páginas con distintos navegadores en sus distintas versiones.

1.22.2 Propiedades relativas o tipos de letras

Las propiedades que nos permiten definir el aspecto tipográfico del texto son:

- Font-family: tipo de letra (que puede ser genérico entre serif, cursive, sans-serif, fantasy o mono-space). Podemos identificar no solo un tipo, sino una lista de tipos, genéricos o no, separados por comas (,).
- Font-size: tamaño del tipo de letra. Xx-small, x-small, small, medium, large, x-large, xx-large, además de valores numéricos de tamaños relativos a tamaños absolutos.
- Font-Weight: Grosor del tipo de letra. Los valores posibles son: normal, bold, bolder, lighter, además podemos poner valores numéricos de tamaño relativo a absoluto.

- Font-style: estilo del tipo de letra. Disponemos de normal, italic, italic-small-caps, oblique, oblique-small-caps, small-caps.

1.22.3 Propiedad del texto

Disponemos también de un grupo de propiedades que nos permiten alterar el texto de nuestra página y el formato de este.

- Line-height: interlineado en valor numérico o porcentual.
- Text-decoration: decoración de texto: none, underline (subrayado), overline (sobre rallado), line-through (tachado) o blink (parpadeo).
- Vertical-align: alineación vertical del texto. Disponemos de: baseline (normal), sub (subíndice), sup (superíndice), top, text-top, middle, bottom, text-bottom o un porcentaje.
- Text-transform: modificación del texto: capitalize (la primera letra en mayúscula, uppercase (convierte a minúscula el texto), lowercase (lo convierte a minúscula) o none.
- Text-align: alineación horizontal del texto: left, right, center o justify.
- Text-indent: sangrado de la primera línea del texto en valor absoluto o porcentual.

1.22.4 Propiedades de bloques

Las siguientes propiedades afectan a los bloques de texto (párrafos, etc.).

- Margin-top, margin-right, margin-bottom, margin-left: distancia mínima entre un bloque y los elementos adyacentes. Valores posibles: tamaño, porcentaje o auto.
- Padding-top, padding-right, padding-bottom, padding-left: relleno de espacio en el borde y el contenido del bloque. Valores posibles: tamaño valor absoluto, porcentual, o auto.
- Border-top-width, border-right-width, border-bottom-width, border-left-width: anchura del borde del bloque en valor numérico.
- Border-style: estilo de borde del bloque none-solid o 3D.
- Border-color: color de borde de bloque.
- Width, height: medidas del bloque. Valores en porcentaje, valores absolutos o autos.
- Float: justificación del contenido de un bloque. Valores: left, right o none.
- Clear: Colocación de los elementos respecto al actual. Valores posibles: left, right, both o none.

1.22.5 Otras propiedades

- color: color de texto. Valores en hexadecimal o nombre del color.
- Background: Color de fondo o imagen de fondo. Valores: un color a una URL del fichero de la imagen.

Background: url (fondo.gif);

- Display: decide el carácter de bloque o no de un elemento. Puede ser: inline (como la etiqueta <I> o), block (como <p>, list como o none. Que inhabilita el elemento.
- List-style: estilo de marcador de un elemento de una lista (podemos así usar un grafico como marcador). Valores posibles: disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, none o una url de una imagen.
- White-space: indica que tratamiento debe tener los espacios en blanco, si el habitual o si deben ser respetados tal cual, como en el bloque <pRE>. Valores: normal y PRE.

1.22.5.1 Capas

Permiten un mayor control en el posicionamiento de los elementos de nuestra página. Podemos definir las capas como páginas incrustadas dentro de otras páginas.

Es posible especificar los atributos de las capas (posición, visibilidad, etc.), mediante hojas de estilos, al igual que los demás elementos de HTML, las capas ofrecen de poder controlarse desde lenguajes de programación como JavaScript, que son la base de lo que conocemos como HTML dinámico. Desgraciadamente, las implementaciones de los diversos navegadores son incompatibles entre sí, por lo que nos veremos obligados a usar cantidades ingentes de código de programas para prever todas las posibilidades o bien a limitarnos a usar solo los mínimos comunes. Una de las pocas opciones que tenemos para hacer que funcionen las capas en la mayoría de navegadores estriba en definir las capas mediante hojas de estilo **CSS**.

Podemos ver en el ejemplo siguiente, como colocar una capa que identificaremos como capa1 mediante el uso de id.

```
<style type="text/css">
    #capa1{position:absolute; top:50px; left:50px;}
</style>
```

En este ejemplo colocamos la capa a 50 puntos de la esquina superior izquierda de la página. En este caso, para definir la capa usaremos una etiqueta de tipo <div>

```
<div id="capa1">      ...      Contenido de la capa      ...      </div>
```

Con este código hemos colocado la capa anterior en una posición específica de la página. De igual modo podemos colocar las capas en posiciones relativas respecto a la posición que ocuparía en texto en la página donde están escritas.

La definición es la siguiente:

```
<style type="text/css">
    #capa2 {position:relativa; left:20px; top:100px;}
</style>
```

1.22.5.1.1 Propiedades de las capas

- Left, top: Nos permitan indicar la posición de la esquina superior izquierda de la capa respecto a la capa donde este situada.
- Height, width: Indicara la altura y la anchura de la capa.
- Clip: Nos permitirá definir un área de recorte en el interior de la capa.

- Z-index: indica la profundidad en la pila de capas. A mayor z-index, menor profundidad y mayor visibilidad (se superpondrán a las de menor z-index). Por defecto, el z-index se asigna por orden de definición en el fichero HTML.
- Visibility: Especifican si la capa debe ser visible u oculta. Los valores posibles son: hidden (oculta) o inherit (hereda la visibilidad de la capa padre).
- Background-image: Imagen que formara el fondo de la capa.
- Background-color, layer-background-color: Definen el color de fondo de la capa para Internet Explorer y Mozilla/Netscape, respectivamente.

Ejemplo 1.2 Empleando Marcos, Formularios y Hojas de Estilo.

El Código HTML de la página Index.html:

```
<html>
<head><!--<META HTTP-EQUIV="Refresh" content="10"> -->
  <meta name="keyword" content="TUTORIAL DE HTML">
  <link rel="stylesheet" type="text/css" href="estilos.css">
  <link rel="stylesheet" type="text/css" href="colores.css">
</head>

  <frameset frameborder=no ROWS=12%,85%>
  <frame SRC="Banner.html">

  <frameset cols=14%,67%,19%>

  <frame SRC="Menu.html" >

  <frame SRC="Introduccion.html" NAME="MAESTRO" scrolling="auto">

  <frame SRC="Informacion.html" >
  </frameset>
  </frameset>
</html>
```

El Código HTML de la página Banner.html

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="estilos.css">
  <link rel="stylesheet" type="text/css" href="colores.css">

  <body><!--background="imagenes/BANNER2.gif" -->
    <div id="arriba"></div>
    <div id="cabecera">
      <a href="http://localhost/HTML/Introduccion.html" target="MAESTRO" class="cabecera" title="Inicio" >Inicio</a>
      
      <span class="textoexp" id="textoexpand">HTML BASICO</span></div>
      <div class="cabeceraEnlaces">
        <span class="cabeceraNormal">
          <form method=get>
            <input type="submit" name="Entrar" value="Entrar">
            <input type="text" size=6 name="password" value=password>
            <input type="text" size=6 name="login" value=Login>
          </form>
        </span>
      </div>
```

```
        <a href="Registrarse.html" target="MAESTRO">Crear Cuenta</a>
      /div>
    </div>
  </div>
</body>
</html>
```

El Código HTML de la página Menu.html es el siguiente:

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="estilos.css">
    <link rel="stylesheet" type="text/css" href="colores.css">
  </head>
  <body >
    <div class="barralzquierda">
  <div class="barralzquierdaTitulo">MENU</div>
    <a href="Introduccion.html" target="MAESTRO">Introducción</a>
    <a href="RegistrarProf.html" target="MAESTRO">Registrar Profesor</a>
    <a href="EstructuraH.html" target="MAESTRO">Estructura </a><hr>
    <a href="Cabecera.html" target="MAESTRO">Cabecera </a>
    <a href="Cuerpo.html" target="MAESTRO">Cuerpo </a>
    <a href="DandoFormato.html" target="MAESTRO">Dando formato</a><hr>
    <a href="Enlace.html" target="MAESTRO">Enlaces</a>
    <a href="imagenes.html" target="MAESTRO">Imagenes</a>
    <a href="Tablas.html" target="MAESTRO">Tablas</a><hr>
    <a href="Marcos.html" target="MAESTRO">Marcos</a>
    <a href="Formularios.html" target="MAESTRO">Formularios</a>
    <a href="css.html" target="MAESTRO">CCS</a>
    </div>
  </div>
</body>
</html>
```

El Código HTML de la página Registrarse.html

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="estilos.css">
  <link rel="stylesheet" typw="text/css" href="colores.css">
    <style type="text/css">

      div {    margin: 1em 0; }

      div label {      width: 10%;
                        float:left;
                        margin-left: 20em;
                      }

      input:focus {
                        border: 2px solid #004d69;
                        background: #c8eaf7;}
      fieldset{color:#004d69}
    </style>
  </head>
  <body>
    <div id="contenedor">
      <form>
```

```
<fieldset><legend align="center">===== REGISTRO DE USUARIO =====</legend>
  <p align="right">Todos los campos con este símbolo ( * ) son requeridos</p>
  <div>
    <label for="nombre">Nombre:*</label>
    <input type="text" id="nombre" size="20" />
  </div>

  <div>
    <label for="apellidos">Apellidos:*</label>
    <input type="text" id="apellidos" size="20" />
  </div>

  <div>
    <label for="email">Email:*</label>
    <input type="text" id="email" size="20" />
  </div>

  <div>
    <label for="login">Login:*</label>
    <input type="text" id="login" size="20" maxlength="20"/>
  </div>

  <div>
    <label for="contraseña">Password:*</label>
    <input type="password" id="contraseña" />
  </div>

  <center><input class="btn" type="submit" value="Registrarse" /></center>
</fieldset>
</form>
</div>
</body>
</html>
```

Código de la hoja de estilo “**Estilos.css**”:

```
/*##### Body #####*/

body {
  font-family:arial, verdana, tahoma, sans-serif; /* Propiedad básica de Tipografía*/
  font-size: 10pt; /*propiedad para definir el Tamaño de letra*/
  margin:0 auto; /* Utilizando la propiedad margin CSS, es muy sencillo centrar una página web horizontalmente*/
  overflow:hidden; /*la propiedad overflow para controlar la forma en la que se visualizan los contenidos*/
  /*que sobresalen de sus elementos */
}

img {
  border: none; /*para que la imagen no tenga bordes la imagenes*/
}

.textoexp {
  font-family:Arial; /*tipo de letra*/
  color:black; /*Color del texto*/
  font-size:30pt; /*Tamaño de la fuente*/
  text-align:left; /*Alineación de la fuente*/
}

/* ##### Cabecera #####*/
```

```
#cabecera      {
                height: 4em; /*Altura del contenedor*/
                padding: 0.25em 2.5mm 0 4mm; /*Relleno del contenedor cabecera */
            }

.cabeceraTitulo {
                font-size: 252%;
                text-decoration:none;
                font-weight: bold;
                font-style:italic;
                line-height: 1.5em;
            }
.cabeceraTitulo span { font-weight: normal;}
.cabeceraTitulo a:hover {}
.cabeceraNormal {
                font-size:85%;
                font-weight:bold;
                font-style:italic;
                font-family:Arial;
            }

.cabeceraNormal input:focus {
                border: 2px solid #004d69;
                background: #c8eaf7;
            }

.cabeceraEnlaces {
                font-size:87%;
                padding:0.5ex 0.15ex 0.5ex 0;
                position:absolute;
                right:0;
                top:0;
            }

.cabeceraEnlaces * {
                text-decoration:none;
                padding: 0 1ex 0 0;
                float: right;
            }

.cabeceraEnlaces a:hover { text-decoration: underline; }

.barraMenu * {
                text-decoration: none;
                font-weight:bold;
                padding: 0 2ex 0 1ex;
            }

.barraMenu a :hover{ /*colores.css*/ }

/*##### Izquierda #####*/

.barraIzquierda {
                font-size:11pt;
                width:12.65em;
                float:left;
            }
```



```
.barralzquierda a, .barralzquierda span {
    text-decoration:none; font-weight: bold; /*Sirve para definir la anchura de los
                                                caracteres,*/
    line-height:0.5em;
    padding:0.75ex 1ex;
    display: block; /*sirven para controlar la visualización de los elementos. El
                                                valor block permite mostrar un como si fuera un elemento de bloque*/
}

.barralzquierda a:hover{ /*colores.css*/}

.barralzquierda .barralzquierdaTitulo {
    font-weight: bold;
    padding: 0.75ex 1ex;
}

.barralzquierda .barraTexto {
    font-weight:normal;
    padding: 1ex 0.75ex 1ex 1ex;
}

.barralzquierda .laPágina{ /*colores.css*/}

/*##### Derecha #####*/

.barraDerecha {
    font-size: 11pt;
    width: 12.65em;
    margin: 1ex 0.8ex 0 0;
    float: right;
    clear: right;
    position:absolute;
    border-style: solid;
    border-width: 1px;
}

.barraDerecha a { font-weight: bold;}

.barraDerecha a:hover { text-decoration: none; }

.barraDerecha .barralzquierdaTitulo {
    font-weight:bold;
    margin:1em 1ex;
    padding: 0.75ex 1ex;
}

.barraDerecha .barraTexto {
    font-weight:normal;
    line-height:1.5em;
    padding: 0 1.5em 1em 2ex;
}
```

Código de la hoja de estilo “Colores.css”

```
/*##### colores.css #####*/

.barraDerecha a,.barralzquierda a{ color:#1c377a;}

.barraizquierda span { color:#0a6b98;}

.cabeceraTitulo, .cabeceraEnlaces *, .barralzquierda, .barralzquierdaTitulo,
```

```
.barralzquierda ,.barraDerecha { color:#2e6672; }

.barraMenu a:hover, .barralzquierda a:hover , .cabeceraEnlaces a:hover ,
.cabeceraTitulo a:hover{color:#a7914d;}

.cabeceraTitulo span {color:#20496f;}

.barraMenu, .barraMenu * { color:black;}

.cabeceraEnlaces { color:#bfd0fb;}

/* ##### colores de fondo #####*/

body {
    background:url(imagenes/Baner_Menu.gif);
    background-attachment:fixed;

    /*Le estamos usando una imagen de fondo en el cuerpo de la página*/

#textCuerpo, .barraMenu a:hover, .barraDerecha{ color:#1a3940; }

#cabecera
    {
        background:url(imagenes/BANNER3.gif);
        background-attachment:fixed;
    }

.barraMenu{ background-color:black;}

.barralzquierda .barralzquierdaTitulo, .barraDerecha .barralzquierdaTitulo { background-color: #a3cfd9; }

/* #####Colores de borde #####*/

h1, #textoCuerpo, .barraDerecha , .bordes
{ border-color: #bfd0fb; }
```

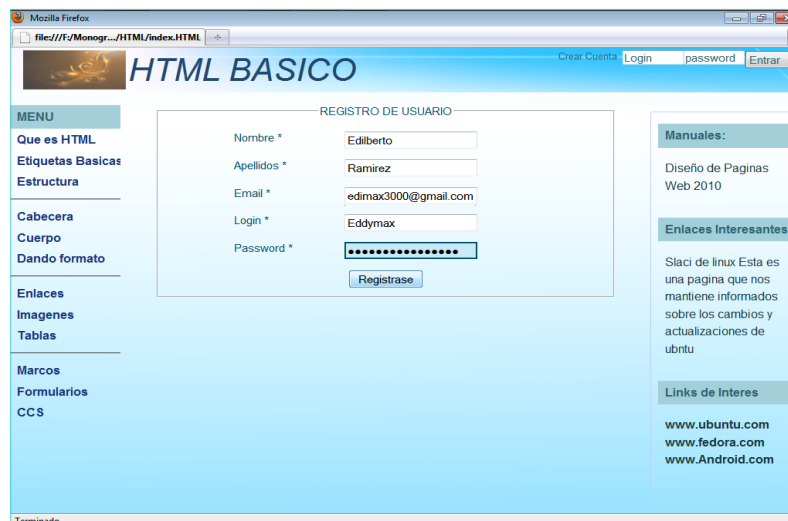


Ilustración 1.4 Este resultado lo obtenemos de la aplicación de Marcos, Formularios y Hojas de Estilo.

1.23 HTML5

Queriendo llevar la plataforma web a un nuevo nivel, un pequeño grupo de personas que iniciaron las aplicaciones web con hipertexto (WHATWG) desde el 2004. Crearon las especificaciones para HTML5, ellos iniciaron también a trabajar en nuevas características específicas con aplicaciones para la web. Ellos sintieron que esa área estaba muy carente y fue entonces cuando el termino WEB2.0 fue acuñado. Esto realmente fue la segunda nueva web, los sitios web eran estáticos y cedieron el paso a sitios web más dinámicos y sociables que requerían muchas más características.

La W3C se involucró nuevamente con HTML en el 2006 y publico el primer bosquejo para HTML5 en el 2008, otro año paso y esto es lo que tenemos hoy en la actualidad, porque HTML5 soluciona los problemas muy prácticos, los productores de navegadores están implementando estas nuevas especificaciones, aunque las especificaciones no han sido completamente cerradas. La experimentación por los navegadores han provisto mejoras a las especificaciones. HTML5 rápidamente está envolviendo las direcciones y prácticas reales mejorando la plataforma web.

HTML5 está basado en varios principios de diseño, escritos fuera de las especificaciones WHATWG, que realmente expresa una nueva visión de posibilidad y aspecto práctico:

- Web Hypertext Application Technology Working Group (WHATWG): fundado en el año 2004 por trabajadores individuales para navegadores y vendedores de aplicaciones, Mozilla, Google, y proveedores de colaboración abiertos a vendedores y otras partes interesadas.
- World Wide Web Consortium (W3C): La W3C contienen grupos de trabajos para HTML, es decir actualmente han repartido especificaciones para HTML5.
- Internet Engineering Task Force (IETF): Esta fuerza especial contiene los grupos responsables de los protocolos para Internet como HTTP. HTML5 define un nuevo WebSocket API que depende de un nuevo protocolo de WebSocket, que está bajo los grupos de trabajo de desarrolladores de la IETF.

Los siguientes fragmentos de código son completamente válidos para HTML5:

```
id="prohtml5"  
id=prohtml5  
id="prohtml5"
```

Nuevos elementos incorporados en HTML5:

HTML5 introduce algunos nuevos elementos de marcas, cada uno dentro de 7 grupos diferentes tipos de contenidos, estos elementos se muestran en la tabla siguiente.

Content Type	
Embedded	Contiene otros recursos importantes dentro del documento, por ejemplo audio, videos, lienzos y marcos.
Flow	Elementos usados en el body de las aplicaciones de los documentos, por ejemplo for, h1, y small.
Heading	Sección de cabecera, por ejemplo h1, h2, y hgroup.
Interactive	Contiene las interacciones con los usuarios, por ejemplo controles de audio o videos, botones y áreas de texto.
Metadata	Elementos que comúnmente se encuentran en la sección del head, donde se pone la presentación o comportamiento del resto del documento, por ejemplo script, style, y title.
Phrasing	Texto y marcas de elementos de texto, por ejemplo mark, kbd, sub, y sup.
Sectioning	Elementos que definen secciones en el documento, por ejemplo article, aside, y title.

Tabla 1.12 HTML5 tipos y contenidos.

Todos estos elementos pueden estar diseñados con CSS. Como agregado, algunos de ellos tales como canvas, audio, y video, pueden ser usados por ellos mismos, aunque son acompañados por API que tienen en cuenta controles programáticos nativos.

Elementos de sección	Description
Header	Contenido de encabezamiento. (Para una página o sección de la página).
Footer	Contenido del pie de página.
Section	Una sección de la página web.
Article	Contenido de artículo independiente.
Aside	Contenido relacionado a citas de distribución.
Nav	Ayuda a la navegación

Tabla 1.13 Nuevas secciones de elementos para HTML5.

Todos estos elementos pueden estar diseñados con CSS. A decir verdad, como describimos en la utilidad de diseño al principio, HTML5 empuja la separación de contenido y presentación, así que usted debe diseñar sus páginas en HTML5 usando páginas de estilo CSS.

Ejemplo 1.3 HTML5 con CSS.



Ilustración 1.5 Ejemplo de HTML5 con CSS.



Ilustración 1.6 Esta es la misma página pero con estilos diferentes.

Código de la hoja de estilo de la página:

```
##### html5.css #####

body {
    /* Formato del contenido de la etiqueta body */
    background-color:#08088A; /* Color del fondo */
    font-family:Geneva,Arial,Helvetica,sans-serif; /* Tipos de letras */
    margin: 2px auto; /* Margen */
    max-width:900px; /* Ancho máximo*/
    border:solid; /* Para que el borde se muestre en un color resaltado */
    border-color:#8181F7; /* Para el color del borde */
}

header {
    /* Esta etiqueta se usa para el contenido de la cabecera */
    background-color:#61380B;
    display:block; /* Desplegar el contenido en bloque */
    color:#CECEF6; /* Color de las letras que están contenidas en esta etiqueta */
    text-align:center; /* Alineación del texto */
}

header h2 {
    /* Especificaciones para las etiquetas h2 dentro de header */
    margin: 2px;
}

h1 {
    /* Especificaciones para las etiquetas h1 de la página */
    font-size: 60px; /* Tamaño de las letras contenidas por la etiqueta h1 */
    margin: 2px;
}

h2 {
    /* Especificaciones para las etiquetas h2 de la página */
    font-size: 24px; /* Tamaño de las letras contenidas por la etiqueta h1 */
    margin: 2px;
    text-align:center;
    color: #8181F7;
}

h3 {
    /* Especificaciones para las etiquetas h3 de la página */
    font-size: 16px;
    margin: 2px;
    text-align:center;
    color:#CECEF6;
}

h4 {
    /* Especificaciones para las etiquetas h4 de la página */
    color: #F47D31;
    background-color: #fff;
    -webkit-box-shadow: 2px 2px 20px #888; /* Webkit: Chrome, Safari */
    -webkit-transform: rotate(-45deg); /* Webkit: Chrome, Safari */
    -moz-box-shadow: 2px 2px 20px #888; /* Mozilla: Firefox, Flock */
    -moz-transform: rotate(-45deg); /* Mozilla: Firefox, Flock */
    position: absolute;
    padding: 0px 150px; /* Relleno */
    top: 50px;
    left: -120px;
    text-align:center; /* Alineación del texto */
}

nav {
    /* Contiene elementos para la navegación */
    display:block; /* Desplegar en bloque el contenido */
    width:25%; /* Ancho del 25% de la pantalla */
    float:right; /* Flotando al lado derecho de la pantalla */
}
```

```
nav a:link, nav a:visited { /* Especificaciones para los enlaces contenidos en las etiquetas nav */
    display: block;
    border-bottom: 3px solid #FFFF00;
    padding: 10px; /* Relleno */
    text-decoration: blink; /* blink: para que el texto parpadee */
    font-weight: bold;
    margin: 5px;
}

nav a:hover { /* Para agregar un color al fondo y al texto cuando el mouse este encima */
    color: white;
    background-color:#8181F7;
}

nav h3 {
    margin: 5px;
    color: Purple;
}

#container { /* Es el Id de la <div> que está en el body */
    background-color:#fff;
}

section { /* Es un nuevo elemento para definir secciones en el documentos html */
    display:block;
    width:50%;
    float:right;
}

article { /* Define un artículo dentro del documento */
    background-color: #eee;
    display:block;
    margin: 25px;
    padding: 10px;
    -webkit-border-radius: 15px; /* Webkit: Chrome, Safari */
    -moz-border-radius: 15px; /* Mozilla: Firefox, Flock */
    border-radius: 15px;
    -webkit-box-shadow: 2px 2px 20px #eee; /* Webkit: Chrome, Safari */
    -webkit-transform: rotate(-15deg); /* Webkit: Chrome, Safari */
    -moz-box-shadow: 2px 2px 20px #888; /* Mozilla: Firefox, Flock */
    -moz-transform: rotate(-15deg); /* Mozilla: Firefox, Flock */
}

article header { /* Especificaciones para las cabeceras del articulo */
    -webkit-border-radius: 15px;
    -moz-border-radius: 15px;
    border-radius: 15px; /* Para definir los radios */
    padding: 10px;
}

article h1 { /* Para los h1 contenidos en la etiqueta articulo */
    font-size: 20px; /* Tamaño de fuente */
}

aside { /* Para designar textos que es tangencial al tema principal */
    display:block;
    width:25%;
    float:right; /* left */
}

aside h3 { /* Especificaciones para h3 dentro de aside */
    margin: 15px;
    color: Purple;
}
```

```
aside p {          /* Especificaciones para el texto dentro de aside */
    margin: 15px;
    color: pink;
    font-weight: bold;
    font-style: italic;
}

footer {           /* Provee de una nueva sección en HTML5, usado en los navegadores modernos */
    clear: both;
    display: block;
    background-color: #F47D31;
    color: #FFFFFF;
    text-align: center;
    padding: 15px;
}

footer h2 {        /* Especificaciones para la etiqueta h2 del footer */
    font-size: 14px;
    color: Purple;
}

/* links */
a {               /* Muestra el color del texto de los enlaces cuando no han sido visitados */
    color: #F47D31;
}

a:hover {         /*
    text-decoration: underline;
}
```

Este es el código de la página HTML

```
<html>
<head>
    <meta charset="utf-8" >
    <title>MI PERFIL EN HTML5</title>
    <link rel="stylesheet" href="html5.css" > <!-- Enabled to use style sheets html5.css -->
</head>
<body>
    <header>
        <h1>"Mi Perfil Personal" </h1>
        <h2>Resumen General</h2>
        <h4> Yo uso HTML5 <blink>!</blink></h4>
        <h4> Porque es muy vers&acute;til</h4>

    </header>

    <div id="container">
        <nav>
            <h3>Mis Links de Navegaci&acute;n Favoritos son estos:</h3>
            <a href="http://www.example.com">Primer Enlace</a>
            <a href="http://www.example.com">Segundo Enlace</a>
            <a href="http://www.example.com">Tercer Enlace</a>
        </nav>

        <section>
            <article>
                <header>
                    <h1>Articulo Relacionado al T&iacute;tulo</h1>
                </header>
```

```
<p>Mi nombre es -Nombre-, me interese por hacer esta p&aaacute;gina para
que conozcan mis habilidades y para que les sirva como ejemplo para que
hagan su perfil propio.</p>
</article>
<article>
  <header>
    <h1>Resumen Personal</h1>
  </header>
  <p>Bueno, yo inicie mi estudios en la UNAN-Le&oacute;n en la
  carrera de Ingeniero en Sistemas, durante la carrera fui aprendiendo
  que si no compartes lo poco que conoces, eres un total fracaso.</p>
</article>
</section>
<aside>
  <h3>Síntesis de mi experiencia laboral</h3>
  <p>Esta es la parte buena, es donde describimos las más relevantes vivencias
  por las que hemos pasado y de las que hemos adquirido nuevos
  conocimientos.</p>
</aside>
<footer>
  <h2>Pie de P&aacute;gina</h2>
</footer>
</div>
</body>
</html>
```

1.24 Programa CGI

Un programa CGI es un programa que se ejecuta en el SERVIDOR WEB y se invoca desde una página web. El programa CGI toma habitualmente como parámetros de entrada los datos introducidos desde un formulario HTML. Mediante el programa CGI se puede ampliar de manera prácticamente ilimitada el tipo de operaciones que se pueden realizar desde una página web.

Modo de operación de un programa CGI:

1. El programa recibe la información que le envía el cliente web. Es posible que un programa CGI no necesite recibir datos para cumplir su misión.
2. La información es decodificada y procesada adecuadamente.
3. El programa genera un resultado (Habitualmente un documento HTML) y lo devuelve al servidor que a su vez lo devuelve al cliente.

El programa puede procesar él mismo los datos recibidos o bien pasárselos a otro programa, como por ejemplo una aplicación de consultas de bases de datos, en este caso actúa como intermediario o pasarela entre el cliente web y la aplicación externa (de aquí la palabra gateway, pasarela en inglés de nombre CGI).

1.25 Publicación de páginas Web.

Para completar el proceso de desarrollo de un sitio Web queda un paso final, que consiste en colocar las páginas en un SERVIDOR WEB para que sean accesibles a todo el mundo a través de Internet, esta operación se conoce como **PUBLICAR LAS PÁGINAS**.

Para poder publicar una página es condición necesaria disponer de un espacio reservado en un servidor web, lo que se denomina una cuenta. Suele incluir además uno o varios buzones de correo electrónico. Si la página

utiliza elementos tales como programación CGI o páginas ASP, se debe contar con la correspondiente autorización para ello, lo cual dependerá de las características concretas de la cuenta.

Existen muchos tipos de cuenta, aunque se pueden clasificar en dos categorías diferentes: las cuentas de pago y las cuentas gratuitas. En la primera se paga una cuota mensual o anual, por lo que además del espacio en el servidor web y los buzones de correo se disponen de otros servicios adicionales. En las cuentas gratuitas en cambio, no se paga cuota alguna pero a cambio la cantidad y calidad de los servicios que presta el servidor web son limitados.

La forma habitual de trabajo a la hora de crear páginas web consiste en desarrollar las páginas en modo local, esto es en la propia computadora. Una vez que las páginas están creadas y han sido verificadas, se transfieren a su ubicación definitiva en el servidor web.

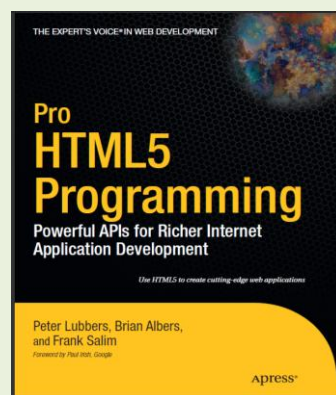
Bibliografía del Capítulo1

Desarrollo de aplicaciones web.
Carles Mateu.



Desarrollo web ---- (librito pasta delgada).

Pro HTML5 Programming Peter Lubbers, Brian Albers y Frank Salim.





Capítulo 2

JAVASCRIPT

2 VISIÓN DE CONJUNTO

Muchos desarrolladores que son nuevos en JavaScript se sienten confundidos por el nombre debido a la información confusa que han leído sobre lenguajes de script, lenguajes de programación, JavaScript, Java, UVSCRIPT y JSCRIPT.

Java es un lenguaje de programación limitado en características. Los programas JavaScript denominados scripts se incluyen en una página web dentro de la etiqueta HTML <scripts>. El navegador descarga una JavaScript cuando se descarga la página web. Se puede ejecutar un JavaScript sin que nada se muestre en pantalla, cuando se realizan cálculos, o puede ocupar toda la ventana del navegador cuando se muestra un formulario JavaScript. Un programa JavaScript no puede acceder al disco duro de su ordenador.

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Legalmente, JavaScript es una marca registrada de la empresa Sun Microsystems, como se puede ver en <http://www.sun.com/suntrademarks/>

2.1 Valores y Variables

Las páginas web contienen numerosa información con algunas imágenes esparcidas para llamar su atención, en HTML sitúa la información que quiere mostrar entre una variedad de etiquetas HTML. La información que se sitúa en una página web o JavaScript se denomina valor.

Java script utiliza seis tipos de valores:

- Número.
- Cadena.
- Boolean.
- Nulo.
- Objeto.
- Función.

Antes de poder utilizar una variable, debe decirle al navegador que cree una. Esto lo realiza al declarar una variable. Siempre que quiera que el navegador haga algo tiene que escribir una sentencia dentro de su JavaScript. Considere una sentencia como una frase que envía una orden al navegador.

Las variables son fundamentalmente para los lenguajes de programación, elementos de datos que representan una ubicación de almacenamiento de memoria en el equipo, las variables con contenedores que contienen datos tales como números y cadenas, poseen un tipo y un valor.

```
Num = 5;           // Nombre "Num", valor 5, tipo numérico.  
Amigo = "Ángel"    // Nombre "Amigo", valor "Ángel", tipo cadena (string).
```

A los lenguajes de programación que requieren se especifique un tipo de datos, se les llama "inflexibles". JavaScript por el contrario, es un lenguaje dinámico o mal escrito, lo que significa que no es necesario especificar el tipo de datos de una variable.

Ejemplo 2.1 Uso de Variables en JavaScript.

```
<html>
<head><title>Usando variables </title>
  <script type="text/javascript">
1      var language="Alemán"; // Esta Variable esta inicializada
2      var name; // Esta variable no tiene definición.
3      age; // Este es un error.
  </script>
</head>
<body bgcolor="silver">
  <big>
    <script type="text/javascript">
      document.write("El lenguaje es " + language + "<br />");
      document.write("El nombre es "+ name + "<br />");
4      document.write("La edad es "+ age + "<br />");
    </script>
  </big>
</body>
</html>
```

Explicación:

1. La variable llamada lenguaje está definida e inicializada. La palabra clave **var** no es requerida para que funcione la asignación.
2. Por no inicializar la variable llamada **name**, la palabra clave **var** si es requerida.
3. La variable llamada **age** no está asignada, ni esta inicializada con ningún valor. La palabra clave var es requerida, sin ella el programa producirá error, el cual se mostrara en el navegador Firefox a como se muestra en la imagen.
4. La línea que no aparece en el navegador es la de la edad porque no tiene la palabra clave var, ni tiene asignado nada.

La palabra clave que se necesita para declarar una variable es **var**. El nombre de la variable puede constar de cualquier letra, dígito y un subrayado pero no puede empezar con un dígito.

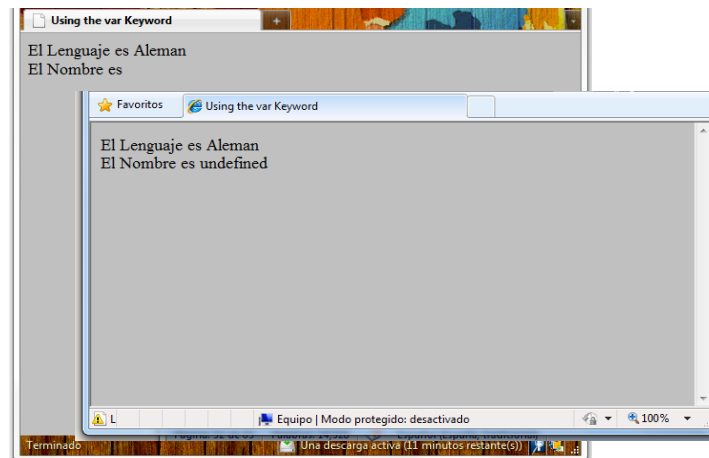


Ilustración 2.1 Esta es la salida del ejemplo 2.1

2.2 Cajas de Diálogo

El objeto del documento es definido dentro de una ventana. La ventana es también un objeto y tiene sus propios métodos. El objeto de la ventana utiliza cajas de diálogo para interactuar con el usuario. Las cajas de diálogo son creadas con tres métodos:

- `alert ()`
- `prompt ()`
- `confirm ()`

2.2.1 El método `alert()`

La ventana de `alert()`, se utiliza para enviar una advertencia al usuario. Por ejemplo, es posible que el usuario no haya escrito su dirección de e-mail correctamente al llenar el formulario, o que su navegador no soporta un determinado plug-in, y así sucesivamente. También se utiliza generalmente para la depuración, para saber los resultados de un cálculo, si el programa se está ejecutando en un orden esperado, y así sucesivamente.

El método `alert()`, crea una pequeña ventana independiente llamado cuadro de diálogo que contiene el mensaje personalizado para el usuario colocado después de un pequeño triángulo, y debajo tiene un botón de Ok. Cuando el cuadro de diálogo aparece, toda la ejecución se detiene hasta que el usuario hace clic en el botón Ok en el cuadro emergente. La apariencia exacta de este cuadro de diálogo puede variar ligeramente en diferentes navegadores, pero su funcionalidad es la misma.

El `alert()` no necesita un nombre delante del objeto ventana como `window.alert()`, a diferencia del `write()`. Debido a que la ventana es el objeto de nivel superior del navegador.

El mensaje para el cuadro de diálogo del `alert`, puede ser cualquier expresión válida, variable o cadena de texto entre comillas, y se envía como un solo argumento al método `alert()`. Etiquetas HTML no se procesan dentro de la cadena de mensaje, pero puede utilizar la secuencia de escape `\n` y `\t`, una precaución. No use excesivamente el cuadro del `alert` en sus aplicaciones web porque puede verse seriamente molesto por los usuarios visitantes.

Sintaxis:

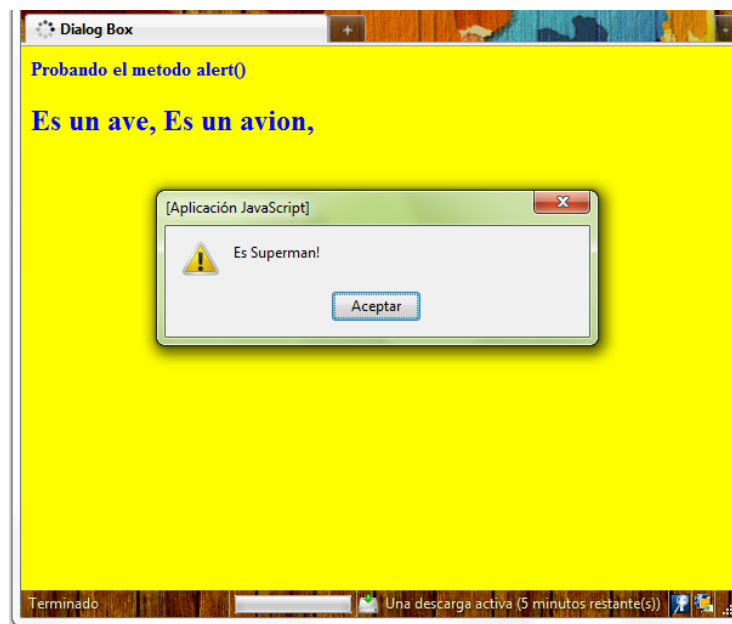
```
alert("expresión");  
alert("Cadena de texto");  
alert("su número telefónico es correcto");  
alert(a + b);
```

Ejemplo 2.2 Utilización de `alert()`.

```
<html>  
  <head><title>Dialog Box</title></head>  
  <body bgcolor="yellow" text="blue">  
    <b>Probando el método alert()</b><br />  
    <h2>  
1      <script type="text/javascript">  
2        document.write("Es un ave, ");  
        document.write("Es un avión,<br />");  
3        alert("Es Superman!");  
      </script>  
    </h2>  
  </body>  
</html>
```

Explicación:

1. El programa JavaScript inicia en la etiqueta **<script>**, el motor comienza a ejecutar JavaScript hasta que llega a la etiqueta de finalización del script **</script>**. JavaScript no entiende las etiquetas HTML a menos que se incrusten en una cadena.
2. El método **document.write()** se envía a la salida del navegador.
3. El método **alert()** crea una pequeña caja de diálogo, independiente del documento, detendrá el procesamiento de todo hasta que el usuario haga clic en el botón Aceptar. El pequeño cuadro de diálogo se puede mover por toda la pantalla con el ratón.

**Ilustración 2.2** Utilizando el método alert().**Ejemplo 2.3** Utilización de alert() con variables.

```
<html>
  <head>
    <title>Using JavaScript alert box</title>
1    <script type="text/javascript">
2      var message1="Estamos orgullosos de ";
      var message2="Estudiar en UNAN ";
3      alert("Bienvenido a \n Programación con JavaScript !");
4      alert(message1 + message2);
    </script>
  </head>
</html>
```

Explicación:

1. El programa JavaScript inicia en la etiqueta **<script>**.
2. Se declaran dos variables **message1** y **message2** de tipo cadenas de texto y se inicializan.
3. El método **alert()** contiene una cadena de texto, con una secuencia de escape en medio de la cadena (**\n**), lo que provoca un salto de línea en la cadena, cuando el cuadro de diálogo aparezca en la pantalla el programa se detendrá hasta que el usuario haga clic en el botón Aceptar.

4. El `alert()` no solo acepta cadenas de texto, sino también variables como argumentos, el signo “ + ” se utiliza para concatenar el contenido de las dos variables y forma una sola cadena.

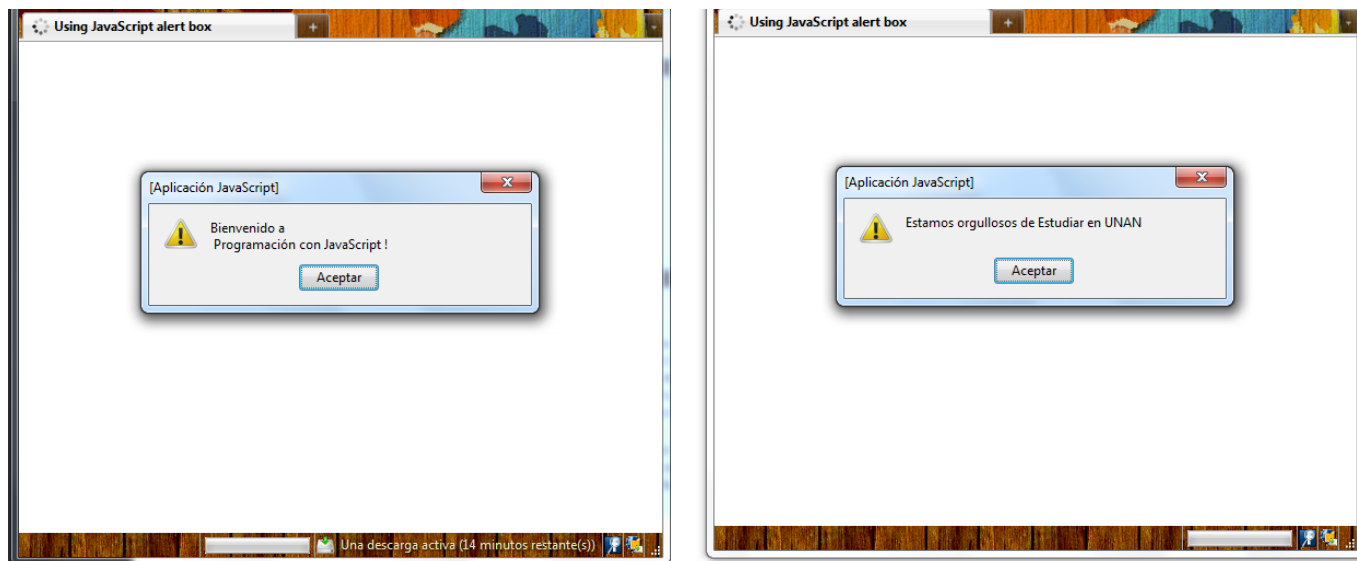


Ilustración 2.3 Después de aparecer la primera caja y el usuario dio clic en el botón Aceptar, la segunda caja aparece.

2.2.2 El método `prompt()`

El método `prompt()` le pide al usuario una pequeña cantidad de información, como una contraseña, realización de un formulario de entrada o información personal como sobrenombre o título. El cuadro de diálogo del sistema aparece son un simple campo de texto en una caja. Después que el usuario escribe texto en el cuadro de diálogo. Este método toma dos argumentos: una cadena de texto que se muestra normalmente para el usuario, se le pide al usuario que haga algo y otra cadena de texto que es el valor predeterminado de configuración inicial de la caja. Si este argumento es una cadena vacía, nada se muestra en el cuadro, el indicador `()` siempre devuelve un valor. Si el usuario hace clic en el botón Ok, todo el texto en el cuadro se vuelve, de lo contrario se devuelve `null`.

Sintaxis:

```
prompt(mensaje);  
prompt(mensaje, texto por defecto);  
prompt("Cuál es su nombre? ", " ");  
prompt("De donde es su nombre? ", nombre);
```

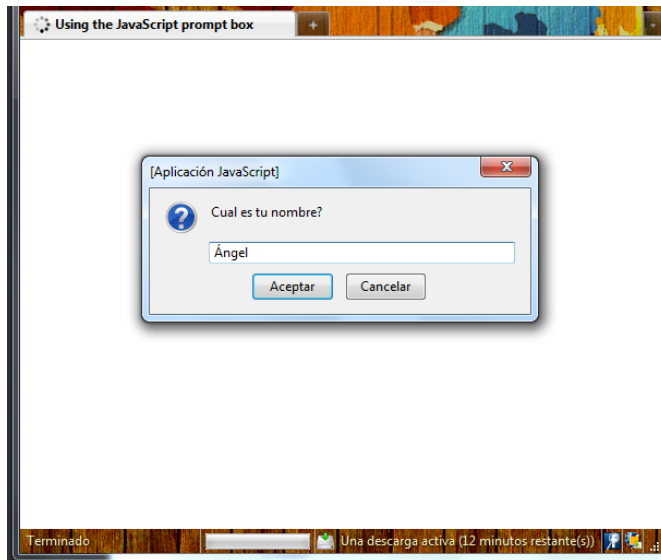
Ejemplo 2.4 Utilización del `prompt`

```
<html>  
  <head>  
    <title>Usando el prompt JavaScript </title>  
  </head>  
  <body>  
    <script type = "text/javascript">  
1      var name=prompt("Cuál es tu nombre?", "");  
2      alert("Bienvenido al mundo Web dinámico! " + name);  
3      var age=prompt("Dime tu edad.", "Tu edad: ");  
4      if ( age == null){ // Si el usuario hace clic en el botón cancel  
5        alert("No compartiste tu edad con migo");  
      }  
      else{  
        alert(age + " Pero si eres joven");  
      }  
    </script>  
  </body>  
</html>
```

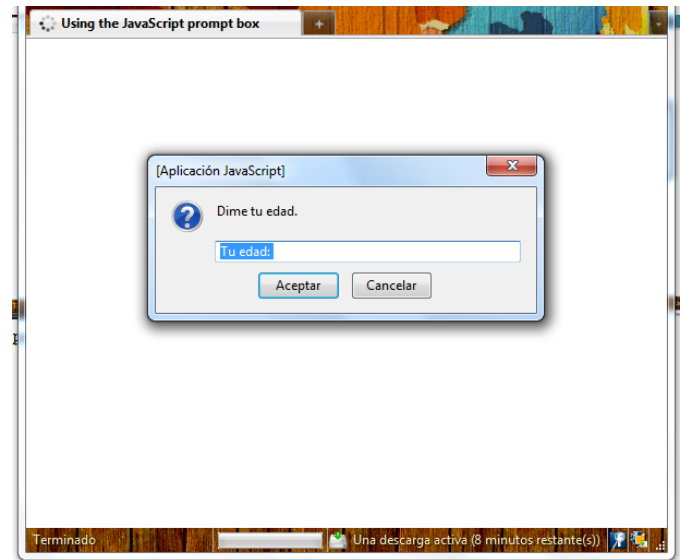
```
        }  
6      alert(prompt("Donde vives? ", " "));  
    </script>  
  </body>  
</html>
```

Explicación:

1. El prompt() toma dos argumentos, uno es el texto que el usuario introduce cuando contesta la pregunta, este texto aparecerá sobre el cuadro de diálogo del sistema. El segundo argumento proporciona el texto por defecto que aparecerá en el extremo izquierdo, dentro del cajetín. Si el segundo argumento es una cadena vacía, el cuadro del mensaje estará vacío.
2. Después de que el usuario proporcione su respuesta en el cuadro de texto del sistema, la respuesta se devuelve y se le asignará a una variable con nombre, el método alert() despliega el valor en la pantalla.
3. La variable llamada edad se le asignará cualquier tipo de información que el usuario introduzca en el cajetín del sistema. Esta vez, un segundo argumento: "Su edad", se enviará al método prompt(). Cuando el cuadro de mensaje aparezca en la pantalla, su edad: aparecerá dentro de la caja donde el usuario escribirá su respuesta.
4. Si el usuario hace clic en el botón cancelar, el valor de retorno del método prompt() será null. Esto es para ver si el valor de la edad es nulo.
5. Si el valor retornado fuera nulo, esta línea se imprimirá en la caja de diálogo del alert().
6. El método prompt() es enviado como argumento al método alert(), después que el usuario ha hecho clic en el botón Ok de la caja del prompt, el valor de retorno se envía al método alert(), y a continuación aparecerá en la pantalla.



Prompt sin un segundo argumento por defecto.



Prompt con un segundo argumento por defecto.

Ilustración 2.4 Utilizando en método prompt().

2.2.3 El método confirm ()

El cuadro de diálogo de confirmación se usa para confirmar la respuesta de un usuario a una pregunta. El usuario debe estar de acuerdo por la acción que se ha completado, verá esto a menudo cuando haga un pedido en línea o al eliminar un archivo en el que una confirmación de sí o no determina lo que sucederá después.

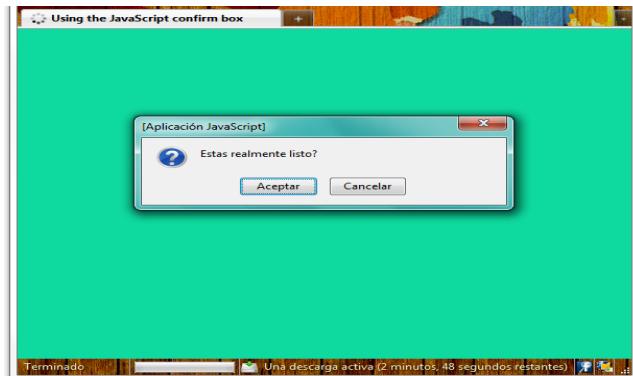
Un signo de interrogación aparecerá en la caja con un botón de Aceptar y un botón de Cancelar. Si el usuario hace clic en el botón Aceptar, se devuelve verdadero, si el usuario hace clic en el botón Cancelar, se devuelve falso. Este método tomará un único argumento que le pedirá al usuario.

Ejemplo 2.5 Utilización del método Confirm()

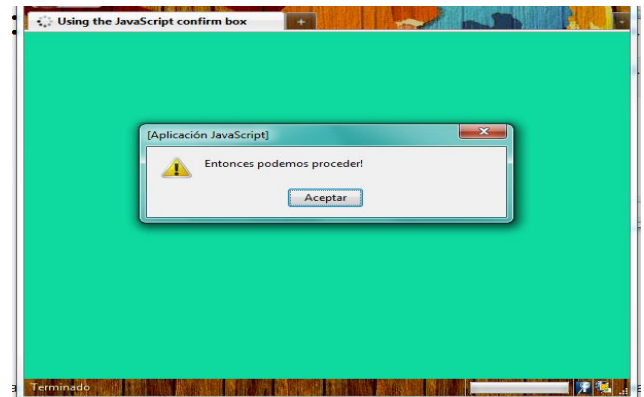
```
<html>
<head>
<title>Usando el método confirm de JavaScript </title>
</head>
<body bgcolor="verdana">
<script type="text/javascript">
1      if(confirm("Estas realmente listo?") == true){
2          alert("Entonces podemos proceder!");
        }
        else{
3          alert("Entonces trataremos nuevamente cuando te sientas mejor? ");
        }
    </script>
</body>
</html>
```

Explicación:

1. El cuadro de diálogo de confirmación solo toma un argumento, la pregunta que desea hacer al usuario. Devuelve verdadero si el usuario hace clic en el botón Aceptar y falso si el usuario hace clic en el botón cancelar. El usuario tiene que hacer clic en cualquiera de ellos para continuar. Si el retorno es verdadero, entonces el alert() en la siguiente línea se ejecutará. La forma en que se usa aquí es para probar una alternativa verdadera o falsa para que el ejemplo sea un poco más significativo.
2. Si el usuario hace clic en Aceptar, la caja de diálogo del alert() mostrará el mensaje.
3. Si el usuario hace clic en Cancelar, la caja de diálogo del alert() mostrará este mensaje.



Caja de diálogo del Confirm



La confirmación de la Caja de Texto.

Ilustración 2.5 Utilizando el método confirm().

2.3 Operadores y Expresiones en JavaScript

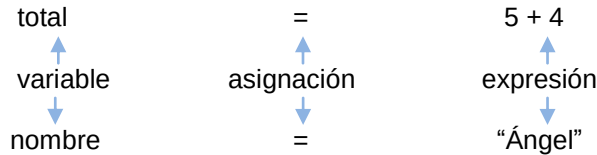
Los objetos de datos se pueden manipular de varias maneras por el gran número de operadores proporcionados por JavaScript. Los operadores son símbolos, tales como +, -, =, > y <, que producen un resultado sobre la base de algunas reglas. El operador manipula los objetos de datos llamados operandos, por ejemplo, 5 y 4 están operando en la expresión 5 + 4. Operadores y operandos se encuentran presentes en las expresiones. Una expresión combina un conjunto de valores para hacer un nuevo valor, $n = 5 + 4$. Y cuando usted termina una expresión con un punto y coma o una nueva línea, dispone de una nueva declaración (por ejemplo, $n = 5 + 4$);).

Usted ya ha utilizado algunos de los operadores: el operador de concatenación para unir dos cadenas, el operador **typeof** para determinar qué tipo de datos se está utilizando, y el operador de asignación que se utiliza para asignar

un valor a una variable. Ahora echemos un vistazo a unos nuevos operadores de JavaScript y veremos cómo se manipulan sus operandos.

Una instrucción de asignación evalúa la expresión en el lado derecho del signo igual y asigna el resultado a la variable en el lado izquierdo del signo igual. El signo igual es el operador de asignación.

```
var total = 5 + 4;
var nombre = "Ángel";
```



Operador	Descripción	Asociatividad
()	Paréntesis	Izquierda a Derecha
++ --	Auto incremento, decremento	Derecha a Izquierda
!	Operador Lógico NOT	Derecha a Izquierda
* / %	Multiplicar, dividir, modulo	Izquierda a Derecha.
+ -	Sumar, sustraer	Izquierda a Derecha.
+	Concatenación	Izquierda a Derecha.
> >=	Mayor que, Mayor o Igual que	Izquierda a Derecha.
== !=	Igual que, Diferente de	Izquierda a Derecha.
=== !==	Idéntico al (mismo tipo), No idéntico al	Izquierda a Derecha
&	Y bit a bit	Izquierda a Derecha.
	OR bit a bit	Izquierda a Derecha
^	XOR bit a bit	Izquierda a Derecha
~	NOT bit a bit	Izquierda a Derecha
<<	Desplazamiento a la izquierda bit a bit	Izquierda a Derecha
>>	Desplazamiento a la derecha bit a bit	Izquierda a Derecha
>>>	Cero de relleno bit a bit, desplazado a la derecha	Izquierda a Derecha
&&	Y lógico de	Izquierda a Derecha
	OR lógico	Izquierda a Derecha
?:	Ternario, a condición	Derecha a Izquierda
= += -= /= %= <= >=	Asignación	Derecha a Izquierda
,	(coma)	

Tabla 2.1 Precedencia y Asociatividad.

Ejemplo 2.6 Operadores y expresiones

```
<html>
  <head><title>Primer Ejemplo de JavaScript </title>
    <script type = "text/javascript">
1      var resultado = 5 + 4 * 12 / 4;
    </script>
  </head>
  <body bgcolor="pink" text="blue">
    <big>
      <script type="text/javascript">
2      document.write("resultado = " + resultado,"<br />");
      </script>
    </big>
  </body>
</html>
```

Explicación:

1. El orden de la asociatividad es de izquierda a derecha. La multiplicación y división son de mayor precedencia que la suma y la resta, y éstos a su vez son de mayor prioridad de asignación. Para ilustrar esto, vamos a utilizar paréntesis al grupo de los operandos. De hecho, si usted quiere prevalece la fuerza. Los dos ejemplos siguientes producen el mismo resultado:

```
Var resultado = 5 + 4 * 12 / 4;  
Podría haberse escrito:  
Resultado = (5 + ( 4 * 12 ) / 4 ) ;
```

2. La expresión es evaluada y el resultado es asignado a la variable, resultado. El valor de resultado es mostrado en el buscador, ver la figura siguiente.

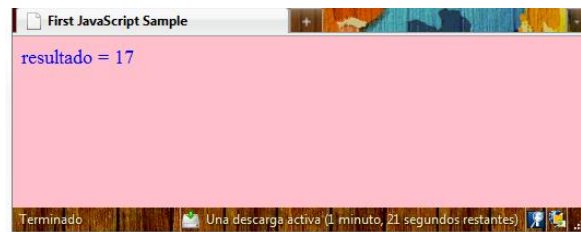


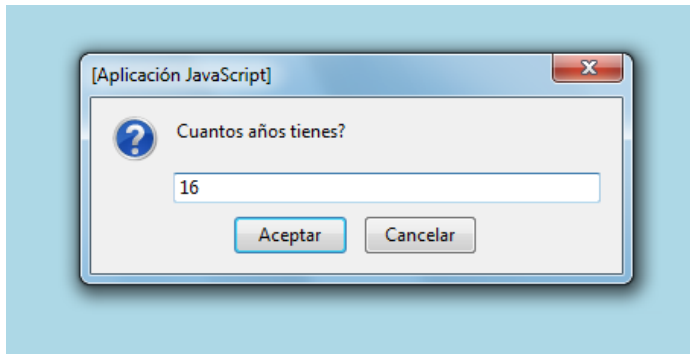
Ilustración 2.6 Operadores y expresiones.

Ejemplo 2.7 Operador lógico AND.

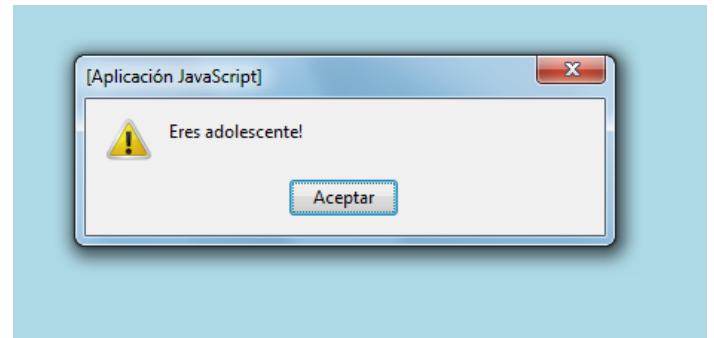
```
<html>  
  <head>  
    <title>Operador Lógico AND </title>  
  </head>  
  <body bgcolor="lightblue">  
    <big>  
      <script type="text/javascript">  
1      var respuesta = prompt("¿Cuántos años tienes? ", " ");  
2      if (respuesta > 12 && respuesta < 20) {  
          alert("Eres adolescente!");  
3      }  
      </script>  
    </big>  
  </body>  
</html>
```

Explicación:

1. Se pide al usuario que ingrese su edad. A la variable llamada respuesta se le asigna el valor ingresado por el usuario, vea la figura siguiente.
2. Si el valor de la variable respuesta es mayor que 12 y menor de 20, la declaración adjunta dentro de las llaves se ejecutará: el cajetín del alert() aparecerá en el que dice eres adolescente.
3. Si el usuario pone cualquier otro valor, no pasará nada.



El usuario ingreso su edad.



Si el usuario ingreso un valor mayor que 12 y menor que 20, aparecerá este cajetín de alert().

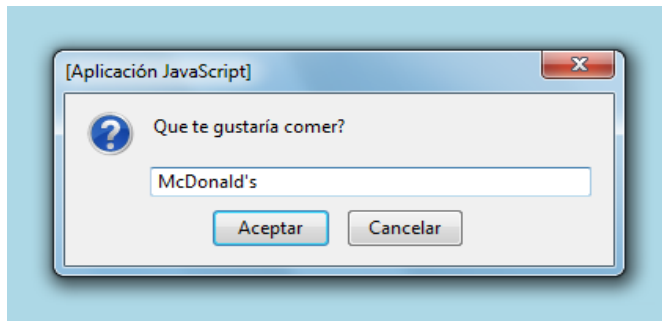
Ilustración 2.7 Salida implementando el operador lógico AND.

Ejemplo 2.8 Operador Lógico OR

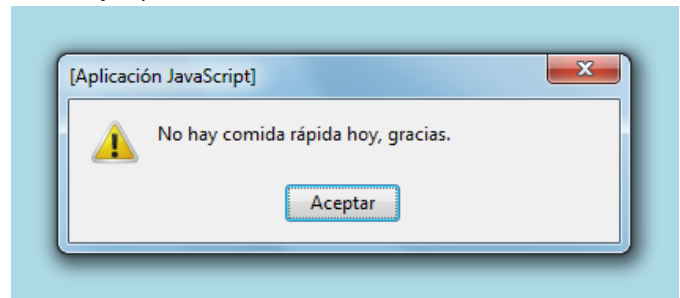
```
<html>
<head>
<title>Operador Lógico OR </title>
</head>
<body bgcolor="lightblue">
<big>
<script type="text/javascript">
1      var respuesta = prompt("¿Qué te gustaría comer? ", "");
      if ( respuesta == "McDonald's" || respuesta == "Taco Bell" ||
2      respuesta == "Wendy's"){
        alert("No hay comida rápida hoy, gracias.");
      }
</script>
</big>
</body>
</html>
```

Explicación:

1. El usuario debe elegir un lugar para comer, a la variable llamada respuesta se le asigna el valor introducido en el cajetín.
2. Si el valor de la variable respuesta es McDonald's o Taco Bell o Wendy, la declaración incluida dentro de las llaves se ejecutará: aparecerá un recuadro de alert que dice no hay comida rápida hoy, gracias.

Ilustración 2.8 Salida del ejemplo 2.8

El usuario ingresa un valor.



El usuario introdujo uno de los valores que están siendo evaluados.

2.4 Funciones

Una calculadora de bolsillo realiza ciertas funciones. Cuando usted pulsa un botón, está enviando información al calculador, se realiza un cálculo y se envía un resultado. Usted no se preocupa por lo que pasa dentro del calculador, solo quiere ver el resultado. Eso es lo que una función hace.

Las funciones son unidades autónomas de un programa que se crean para solucionar tareas específicas tales como hacer los cálculos para el pago de una hipoteca, la visualización de imágenes al azar, o comprobar si los valores de una entrada son válidos. Pueden ser reutilizadas una y otra vez y así no tenemos la necesidad de programar repitiendo código. También sirven para fraccionar un programa en módulos más pequeños que facilitan la mantención y comprensión del mismo.

JavaScript tiene un sin número de funciones integradas, pero usted también puede crear sus propias funciones.

Por definición, una función es un bloque de sentencias que realizan más de una tarea, también devuelven un valor. Una función por ser parte independiente de su programa, no se ejecuta hasta que es llamada. Una función es como un cuadro negro en el que entra la información y devuelve o retorna un valor de salida. Lo que sucede dentro del cuadro negro es transparente para el usuario. El programador es el único que por haber escrito la función se preocupa por esos detalles.

Las funciones deben declararse antes de poder ser utilizadas. Normalmente las funciones se ponen en la etiqueta `<head>` del documento HTML para asegurarse de que están definidas. Las definiciones de funciones son a menudo almacenadas en archivos externos con la extensión de JavaScript.

Para definir una función se utiliza la palabra clave `function` seguida del nombre de la función y un conjunto de paréntesis. En los paréntesis se pondrán los parámetros que recibirá la función. Las declaraciones de la función se encierran entre llaves.

Sintaxis: `function adiós() { document.write ("Bye, adiós, adieu, au revoir..."); }`

Una vez definida la función, ya se puede usar mediante llamadas a la función en cuestión, por ejemplo, `adiós()`. Una función puede ser llamada directamente desde dentro de la etiqueta `<script>`, desde un vínculo o al momento de activarse un evento. Cuando se llama a una función, el nombre de esta va seguido por un conjunto de paréntesis que pueden contener mensajes que se envían a la función. Estos mensajes son los argumentos.

Definición de una función:

```
function nombre_función () {declaración; declaración; }
```

```
function nombre_función (parámetro, parámetro) { declaración; declaración; }
```

Llamada a la función:

nombre_función ();

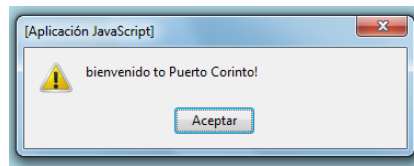
nombre_función (argumento1, argumento2, ...);

Ejemplo 2.9 Una función simple

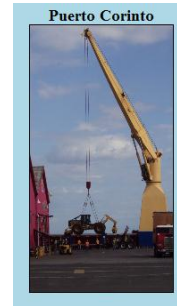
```
<html>
  <head><title>Una Función Simple </title>
    <script type="text/javascript">
1      function bienvenido(){ // Definición de la función dentro de la etiqueta <head>
2        var place="Puerto Corinto";
3        alert("bienvenido a "+ place + "!");
4      }
    </script>
  </head>
  <body bgcolor="lightblue">
    <big>
      <div align="center">
        <script type="text/javascript">
5          bienvenido();
        </script>
        <b>Puerto Corinto</b><br />
6        
      </div>
    </big>
  </body>
</html>
```

Explicación:

1. Las funciones **deben definirse antes de que se puedan usar**, por lo que las funciones son normalmente colocadas en un programa JavaScript, entre el código HTML **<head></head>**. En este ejemplo, la función está definida pero no hará nada hasta que se le llame desde algún lugar del archivo. La palabra clave es **function**, seguida por el nombre definido por el usuario de la función llamada bienvenida y un conjunto de **paréntesis**. Lo que hace la función realmente se define en un conjunto de sentencias encerradas **entre llaves**.
2. Este es el código que se ejecuta cada vez que se invoca la función. Se llama a la definición de la función. Cuando es llamada, la cadena Puerto Corinto se le asigna a la variable y en un cuadro de diálogo de alert() se mostrará el mensaje "Bienvenido a Puerto Corinto" en el navegador.
3. Al final, la definición se cierra con una llave.
4. Cuando la función da inicio() es llamada, las líneas de código dentro de su definición se ejecutan.
5. La función está definida en el encabezado del documento y llamada en el cuerpo del documento antes de la etiqueta . La imagen no aparecerá hasta que el usuario haga clic en el botón Aceptar del cuadro de diálogo del alert().



Después de llamar a la función.



Después de hacer clic en el botón Aceptar

Ilustración 2.9 Salida del ejemplo 2.9

2.5 Pasando Argumentos a funciones

Si un usuario quiere enviar valores a una función, los valores se encierran en paréntesis y después del nombre de la función se envían los argumentos separados por una coma (,) cuando se invoca la función. Los argumentos son recibidos por la función en una lista de argumentos correspondientes llamados parámetros. Los nombres de los parámetros no son necesariamente los mismos nombres de la lista de parámetros, pero que corresponden con los mismos valores.

Las funciones pueden tener variables locales, estas variables desaparecen cuando se cierra la función. JavaScript no guarda un registro del número de argumentos enviados a la función para asegurarse de que coinciden con el número de parámetros especificados en la definición de la función en el otro extremo. Si hay solo dos parámetros definidos dentro de la función, y se le pasan tres a la función, el tercero se pasa por alto. Si se envían tres argumentos y hay cuatro parámetros dentro de la función, el cuarto parámetro no está definido. Es similar al envío de mensajes de un contestador automático. Si se envía un mensaje y la maquina está llena, el mensaje es ignorado, y si usted envía un mensaje y hay espacio para más mensajes, el mensaje enviado se guarda, y el espacio no utilizado está allí pero todavía no está definido.

Una función puede ser llamada también directamente desde un enlace, para este caso la llamada a la función se coloca entre comillas y se asigna al atributo href de la etiqueta <a>. Cuando el usuario hace clic sobre el enlace, en lugar de ir a otra página se hace la llamada una función de JavaScript.

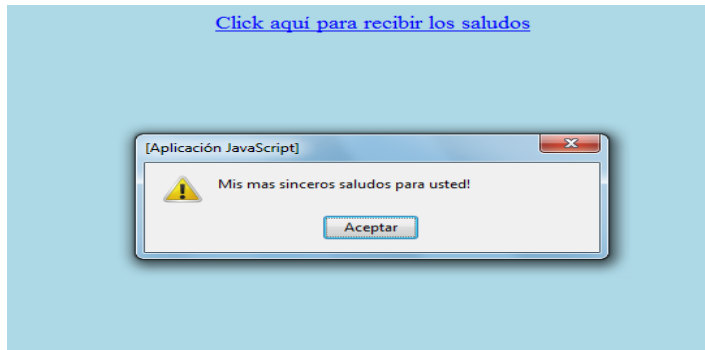
Para una mejor comprensión, veamos el siguiente ejemplo:

Ejemplo 2.10 Funciones

```
<html>
  <head><title>Funciones</title>
1  <script type="text/javascript">
2    function saludos(){ // Definición de la función
      document.bgColor="lightblue";
      alert("Mis más sinceros saludos para usted!");
    }
  </script>
</head>
<body bgcolor="silver">
  <div align="center">
3  <a href="JavaScript:saludos()"><big>Clic aquí; para recibir los
    saludos</big> </a><br/>
  </div>
</body>
</html>
```

Explicación:

1. El programa JavaScript inicia en la cabecera del documento. La función es definida en el encabezado del documento para garantizar que se definió antes de ser llamada.
2. La definición de la función saludos() es muy simple. Hace que el color de fondo del documento pase a color azul claro y hace aparecer un recuadro de alert () con el mensaje de saludos.
3. Al atributo href de la etiqueta <a> se le asigna una cadena formada por el código de JavaScript, seguido por el nombre de la función a ser llamada. Cuando el usuario hace clic en este enlace, JavaScript llama a la función, saludos ().



Después que el vínculo es clicado, se cambia el color de fondo de la página causando que el cuadro de diálogo del alert() aparezca.

Ilustración 2.10 Salida del ejemplo 2.10

2.6 Retornando Valores

Las funciones pueden retornar valores con una instrucción de retorno. La palabra clave return es opcional y solo puede existir dentro de una función. Cuando la palabra return se alcanza en una función, no se produce una transformación en la función.

El retorno se puede utilizar para devolver el resultado de alguna tarea, como el resultado de un cálculo, o para salir de una condición si una condición se produce. Cuando una función no tiene la palabra return, devuelve el valor indefinido (undefined).

Ejemplo 2.11 Retornando Valores

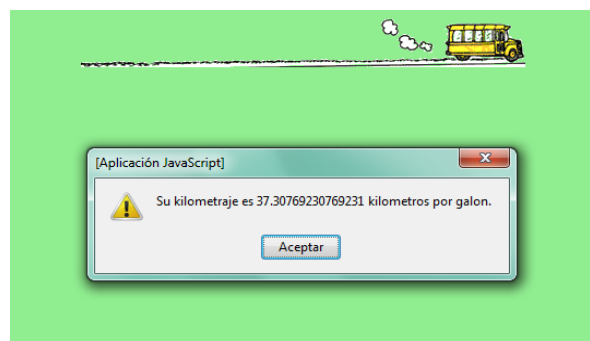
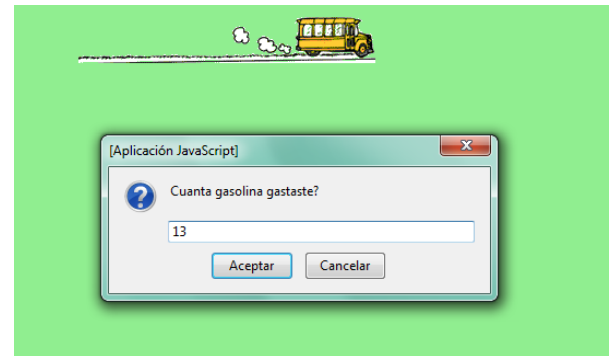
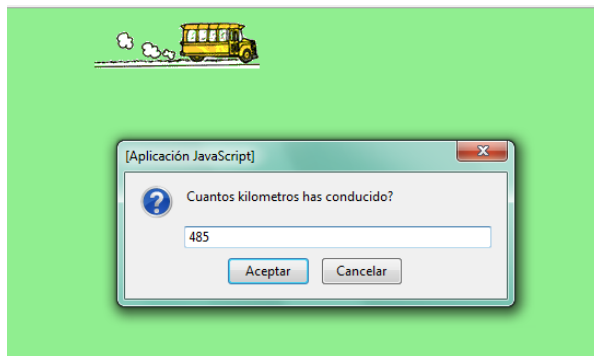
```
<html>
<head><title>Retornando Valor</title>
  <script type="text/javascript">
1    function kilometraje(kilómetros, gasolina){
2      return kilómetros/gasolina; // Retorna el resultado de una división
    }
  </script>
</head>
<body bgcolor="lightgreen">
  <font face="arial" size="+1">
    <div align="center">
      
      <script type="text/javascript">
3        var distancia=eval(prompt("¿Cuántos kilómetros has conducido? ", ""));
        var cantidad=eval(prompt("Cuanta gasolina gastaste?", ""));
4        var calcular = kilometraje(distancia, cantidad);
```



```
5 // Retornar el valor asignado a la variable calcular  
    alert("Su kilometraje es"+ calcular +" kilómetros por galón.\n");  
    </script>  
  </div>  
</font>  
</body>  
</html>
```

Explicación:

1. La función llamada kilometraje () se definió en este programa en la parte del <head> del documento.
2. La sentencia return envía de regreso a la llamada de la función el resultado de la división. El valor retornado se asigna a la variable calcular.
3. El usuario deberá ingresar la cantidad de kilómetros recorridos y la cantidad de gasolina utilizando, estos valores se le asignaran a las variables distancia y cantidad, respectivamente.
4. La función kilometraje () es llamada, pasándole dos argumentos. Debido a que la función kilometraje () está al lado derecho del operador de asignación (el signo =), lo que la función devuelve se le asigna a la variable resultado.
5. En un cuadro de diálogo de alert (), se muestra el resultado devuelto por la función, que es la cantidad de kilómetros por galón.



El número de kilómetros por galón es retornado por la función kilometraje ().

Ilustración 2.11 Salida del ejemplo 2.11

2.7 Sentencias condicionales

Una sentencia condicional es un tipo de sentencia que le dice al navegador que evalúe una condición de si un id de usuario y contraseña son válidos, y basándose en esta evaluación ejecutar o saltarse una o más sentencias en el java script.

Los tipos de sentencias condicionales son tres:

- If
- Switch
- Sentencia en bucle

El bloque de instrucciones está normalmente encerrado entre llaves, y las instrucciones se ejecutan secuencialmente. Si solo hay una declaración después de la expresión condicional, las llaves son opcionales.

2.7.1 If / else.

Las sentencias en JavaScript se pueden manejar de la misma manera con el if /else. Esta construcción permite una resolución de dos vías. La expresión se evalúa entre paréntesis, y si el resultado es verdadero, el bloque de código después de la expresión se ejecuta, de lo contrario el bloque después de ese se ejecuta.

Sintaxis:

```
if (condition){
    statements1;
}
else{
    statements2;
}
```

2.7.2 El operador condicional ?.

Llamado también operador ternario, debido a que a menudo se utiliza como un acceso directo para el if / else, esta sentencia condicional es revisado de Nuevo aquí.

Sintaxis:

conditional expression ? expression : expression

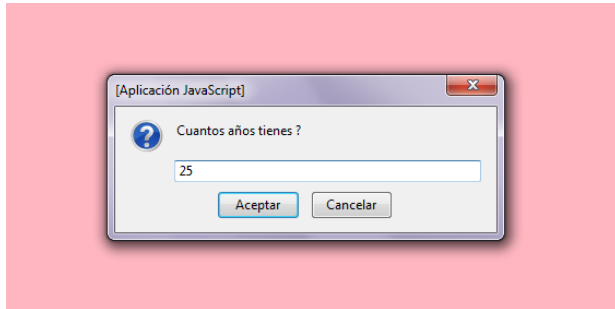
Ejemplo 2.12 Operador Condicional

```
<html>
<head>
  <title>Operador Condicional </title>
</head>
<body bgcolor="lightpink">
  <big>
    <script type="text/javascript">
1    var edad = prompt("¿Cuántos años tienes ? ", "");
2    var precio = (edad > 55 ) ? 0 : 7.50;
3    alert("Usted paga $" + precio + 0);
    </script>
  </big>
</body>
</html>
```

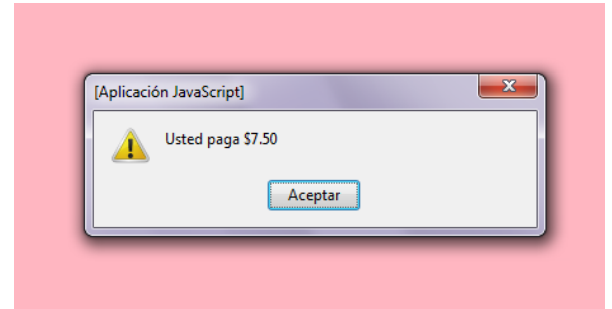
Explicación:

1. Se le pide al usuario en la entrada un valor, el cual es asignado a la variable edad.
2. Si el valor de la edad es mayor que 55 años, el valor a la derecha del signo ? se asigna a la variable precio, de lo contrario el valor que está después de : es el que se le asigna a la variable precio.

3. En la caja de diálogo alert () se muestra el valor de la variable precio.



Cuando se le pide la edad al usuario



Después que el usuario hace clic en el botón aceptar.

Ilustración 2.12 Salida del ejemplo 2.12

2.7.3 Sentencia if/ else if

Esta construcción proporciona una estructura de decisión múltiple.

Ejemplo 2.13 Control de Flujo condicional

```
<html>
<head>
  <title>Control de flujo condicional</title>
</head>
<body>
  <h2>
    <script type="text/javascript">
      <!--
      var edad=eval( prompt("¿Cuántos años tienes? ", ""));
      if( edad > 0 && edad <= 12 ){
        alert("Usted paga la tarifa de un niño. ");
      }
      else if( edad > 12 && edad < 60 ){
        alert("Usted paga la tarifa regular de un adulto. ");
      }
      else {
        alert("Usted paga la tarifa para personas mayores! ");
      }
      //-->
    </script>
  </h2>
</body>
</html>
```

Explicación:

- En un cuadro de diálogo se mostrará el mensaje “¿Cuántos años tienes?”. Cualquiera que sea el número ingresado por el usuario, este se almacena en la variable edad.
- Si el valor de la variable edad es mayor que 0 y la cantidad también es menor o igual a 12, se ejecuta la línea del primer alert () del programa y finaliza el script.
- Si la expresión de la evaluación del primer if es falsa, el intérprete de JavaScript pondrá a prueba esta línea, y si la edad es mayor de 12 y menos de 60, el bloque de sentencias que siguen se ejecutarán el control sale del script.

- El número de la línea del tercer alert (), se ejecutará si todas las expresiones anteriores son falsas. A esta declaración se le llama opción por defecto y no es requerida.
- Finalmente sigue la finalización del programa.

2.7.4 Switch

La sentencia switch es una alternativa a if/else if (comúnmente llamada declaración de caso) y puede hacer que el programa sea más legible al manejar múltiples opciones.

Sintaxis:

```
switch (expresión){
    case nivel :
        declaración (es);
        break;
    case nivel :
        declaración (es);
        break;
    ...
    default : declaración;
```

El valor de la expresión switch se compara con las expresiones, de las etiquetas de llamada después de la palabra clave del caso. Las etiquetas de casos son constantes, ya sea de cadenas o numéricas. Cada etiqueta se termina con dos puntos (:). La etiqueta default es opcional, pero su acción se toma si ninguno de los otros casos coinciden con la expresión del interruptor. El valor predeterminado es opcional. Si una sentencia break se omite, todos los estados por debajo de la etiqueta se ejecutan hasta que un break sea alcanzado.

Ejemplo 2.14 Sentencia Switch

```
<html>
  <head>
    <title>Sentencia Switch</title>
  </head>
  <body>
    <script type="text/javascript">
      <!--
1      var dia_de_semana=Math.floor((Math.random()* 7)+1);
          // Obtener un número aleatorio entre 1 y 7
          // El lunes es 1, el martes es 2, etc.
2      switch(dia_de_semana){
3          case 1:
3          case 2:
3          case 3:
3          case 4:
          alert("El horario de lunes a jueves es de 08:00am a 10:00pm");
4          break;
          case 5:
          alert("El horario de los viernes es de 09:00am a 06:00pm");
          break;
          case 6:
          alert("El horario de los sábados es de las 11:00am a 03:00pm");
          break;
5          default:
          alert("Nosotros estamos cerrando los domingos y festivos");
6          break;
7          }
      </script>
    </body>
  </html>
```

Explicación:

1. La función de números aleatorios genera un número entre 1 y 7 inclusive cuando se ejecuta el script. El número aleatorio se almacena en una variable llamada `day_of_week`.
2. El valor `dia_de_semana` de la expresión `switch` se compara con los valores de cada uno de los casos por debajo de las etiquetas.
3. El primer caso que se analiza es si el número generado es 1, 2, 3 o 4. Para los cuales se generará el mensaje contenido en el primer `alert()`.
4. La sentencia `break` hace que el programa continúe después de la llave de cierre del `switch`, sin ella el programa continuaría la ejecución de instrucciones en el siguiente caso y así sucesivamente hasta que se alcance la terminación del programa y no queremos eso. Cuando se alcanza la sentencia `break` el control del programa se pasa hasta la línea después de la llave de cierre del `switch`.
5. La sentencia por defecto se ejecuta si ninguno de los casos anteriores se corresponden.
6. La última sentencia `break` no es necesaria, pero es buena práctica en caso de optar por sustituir el valor por defecto con una etiqueta o caja de diálogo adicional.
7. La llave final, termina la sentencia `switch`.

2.7.5 Bucles

Los bucles son usados para ejecutar una sentencia de código repetidamente hasta que una condición se cumple. JavaScript cuenta con tres bucles básicos:

- While
- For
- Do while

2.7.5.1 El bucle while

La sentencia `while` ejecuta su bloque de instrucciones, siempre y cuando la expresión que se evalúa es cierta; es decir, no nulo o no falso. Si la condición nunca cambia y es verdadera, el bucle se repetirá siempre (bucle infinito). Si la condición es falsa, el control va a la línea siguiente después del bloque de sentencias de la sentencia `while`.

Ejemplo 2.15 Construcción de bucles

```
<html>
<head>
  <title>Construcción de bucles</title>
</head>
<body>
  <h2>Ciclo While </h2>
  <font tamaño="+2">
    <script type="text/javascript">
1      var i=0; // inicializando el contador del bucle
2      while ( i < 10 ){ // prueba
3        document.writeln(i);
4        i++; // Incrementando el contador
5      } // fin del bloque del ciclo
6    </script>
    </font>
  </body>
</html>
```

Explicación:

1. La variable "i" se inicializa en 0.

2. La expresión después del while es puesta a prueba. Si “i” es menor que 10, el bloque de llaves y sus instrucciones son ejecutadas. Si la expresión evaluada es falsa, el control del programa se salta el bloque del bucle.
3. El valor de “i” es desplegado en la ventana del navegador.
4. El valor de “i” se va incrementando en 1. Si este valor nunca cambiara, el bucle nunca finalizaría.
5. Las llaves del bucle encierran el inicio y finalización del bloque de sentencias del while.
6. El programa JavaScript finaliza en este punto, justo en la llave de cierre del bucle.



Bucle while

Ilustración 2.13 Salida del ejemplo 2.15

2.7.5.2 Bucle do / while

La sentencia do/while ejecuta un bloque de sentencias repetidamente hasta que la condición se vuelve falsa. Este ciclo necesariamente ejecuta las sentencias del bloque al menos una vez antes de evaluar la expresión. El bucle do/while es soportado por mozilla firefox, e Internet Explorer 4.0.

Ejemplo 2.16 Construcciones de bucles

```
<html>
  <head>
    <title>Construcciones de bucles</title>
  </head>
  <body>
    <h2>Bucle Do While </h2>
    <font size="+2">
      <script type="text/javascript">
1      var i=0;
2      do{
3      document.writeln(i);
        i++;
4      } while ( i < 10 )
      </script>
    </font>
  </body>
</html>
```

Explicación:

1. La variable “i” se inicializa en 0.
2. Se introduce al bloque del “do”. Este se ejecutara antes de que se evalúe la expresión. Incluso si la expresión resulta ser falsa, este bloque se ejecutará la primera vez.
3. El valor de “i” se mostrará en la ventana del navegador.
4. Ahora, la expresión while se evalúa para ver si el valor es verdadero. Si es así, el control vuelve a la línea del “do” y el bloque de instrucciones se vuelve a ejecutar.



Bucle do / while

Ilustración 2.14 Salida del ejemplo 2.16

2.7.5.3 Bucle for

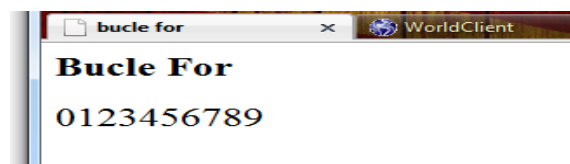
El bucle for consiste en la palabra clave “for” seguida por tres expresiones separadas por punto y coma (;) y encerradas entre paréntesis. Cualquiera o todas las expresiones pueden ser omitidas, pero los punto y coma no se pueden omitir. La primera expresión se utiliza para establecer el valor inicial de las variables y se ejecuta solo una vez, la segunda expresión se utiliza para comprobar si el bucle debe continuar o parar, y la tercera expresión se está actualizando para, es decir se está incrementando o disminuyendo el valor de este contador, que por lo general determina cuántas veces se repetirá el bucle.

Ejemplo 2.17 Bucle for

```
<html>
  <head>
    <title>bucle for</title>
  </head>
  <body>
    <h2>Bucle For</h2>
    <font size="+2">
      <script type="text/javascript">
1        for( var i = 0; i < 10; i++ ){
2          document.write(i);
3        }
      </script>
    </font>
  </body>
</html>
```

Explicación:

1. La expresión inicia con el paso 1, con la inicialización de la variable “i” en 0. Esta es la única vez que se ejecuta este paso. La segunda expresión, la evaluación para ver si “i” es menor que 10, y si lo es, las declaraciones después de la llave de apertura se ejecutan. Cuando las declaraciones del bloque han sido ejecutadas y se llega a la llave de cierre, el control vuelve a la última expresión de las tres. La variable “i” ahora se incrementa en uno y la expresión segunda se vuelve a evaluar. Si es verdadera, el bloque de sentencias se introduce y se ejecuta nuevamente.
2. El valor de “i” se muestra en el navegador.
3. La llave de cierre marca el fin del ciclo.

**Ilustración 2.15** Salida del ejemplo 2.17

Ejemplo 2.18 Ejemplo de bucles anidados:

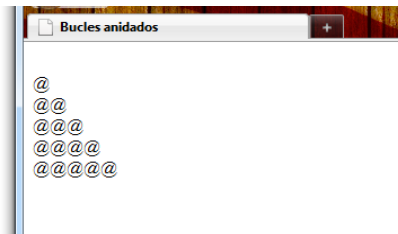
```
<html>
<head>
  <title>Bucles anidados</title>
</head>
<body>
  <script type="text/javascript">

1    var str = "@";
2    for ( var row = 0; row < 6; row++){
3      for ( var col=0; col < row; col++){
        document.write(str);
      }
      document.write("<br />");
    }

  </script>
</body>
</html>
```

Explicación:

1. A la variables “str” se le asigna el valor “@”.
2. Entramos al bucle externo, las variables se inicializan en 0. Si el valor de la fila es inferior a 6, el bloque de sentencias del bucle se continuaran ejecutando.
3. Al entrar al bucle interior, la variable “col” se inicializa a 0. Si el valor de “col” es menor que el valor de la fila, se ejecuta el bloque de código del bucle y se imprime en la pantalla una “@”. El valor de la columna se vuelve a incrementar en 1, se evalúa nuevamente la condición y se imprime nuevamente una “@”, cuando este bucle se completa, se pondrán en marcha nuevamente los estados de las variables del “for” externo.



Ejemplo de bucles anidados, filas y columnas.

Ilustración 2.16 Salida del ejemplo 2.18

2.8 Manejadores de Eventos

2.8.1 Introducción a los manejadores de eventos

JavaScript reacciona a los eventos. Nosotros hemos incluido esta parte porque los eventos son parte inherente de casi todas las páginas web porque ellos hacen las páginas interactivas y dinámicas. Los eventos en JavaScript son asíncronos, lo que significa que pueden suceder en cualquier momento, son acciones que se ejecutan cuando un usuario visita una página web, por ejemplo, cuando el usuario envía un formulario o se mueve con el ratón sobre un enlace o imagen, el usuario podría desencadenar un evento.

Cuando ocurre un evento, el navegador ejecuta algunas acciones por defecto que son necesarias, por ejemplo, si hace clic en un enlace, se abre la ubicación en el href. JavaScript también puede responder a la acción de un usuario y ejecutar código en respuesta.

JavaScript tiene tres modelos de eventos para los eventos de la programación: el modelo de línea, el modelo de secuencias de comandos y el modelo DOM. En esta parte se discutirán los dos primeros modelos, el modelo DOM se describe en el capítulo siguiente.

2.8.1.1 Modelo en línea del Manejador de Eventos.

Esta es la forma más antigua y más fácil de control eventos. Con el modelo en línea, los controladores de eventos son atributos de una etiqueta HTML y se controla con el evento para el cual fue llamado. Por ejemplo, el manejador `onClic` controla el evento clic, el manejador `onSubmit` se encargará de presentar el evento, y así. A pesar de ser una forma muy popular de hacer frente a eventos, la desventaja de usar el modelo en línea es que JavaScript es intrusivo, es decir que no está separado del código HTML. El modelo en línea se utiliza aquí para mostrarle lo que los controladores de eventos hacen, y cómo afectan a los objetos en un documento.

Como hemos visto en los ejemplos anteriores, si el usuario hace clic en el botón de enviar, JavaScript puede comprobar para ver si un formulario fue llenado correctamente, o si se mueve sobre un enlace, en JavaScript puede reemplazar una imagen con una nueva. Si el usuario hace clic en un botón, por ejemplo, JavaScript puede controlar el evento llamando a una función para que realice alguna tarea designada, como para abrir una nueva ventana o poner el foco en una ventana, realizar un cálculo.

Esta es una lista de algunos eventos que se pueden manejar en JavaScript:

- Mouse actions.
- Keyboard actions.
- Actions on form fields.
- When the page is first loaded or unloaded.
- After a specified time has passed.
- When an error has occurred.

2.8.2 HTML y el controlador de eventos.

Los manejadores de eventos JavaScript no se ponen entre las etiquetas `<script>` `</script>`. Los controladores de eventos son atributos de las etiquetas HTML (Especificados en las especificaciones de HTML 4). Si el evento es asociado a una etiqueta de formulario, entonces será un atributo de la etiqueta `<form>`, y si se asocia con un enlace, será un atributo de la etiqueta `<a href>`, y así sucesivamente. Una vez que se ha decidido que evento desea controlar, se suele asignar una función al controlador de eventos. Esto se conoce como el registro del evento.

Para registrar un evento, una cadena que representa un comando se asigna al controlador de eventos. El comando, por lo general es una función de JavaScript y se ejecuta cuando se activa el evento por el usuario. Considerando que una propiedad o método podrían estar asociados con un solo objeto, los eventos se asocian generalmente con más de un objeto. El controlador de eventos `onClic`, por ejemplo, puede estar asociado con la etiqueta de entrada de un formulario, pero también puede estar asociado a una etiqueta de enlace, a un área de mapa de imágenes, o a un simple botón.

(Hay que tener en cuenta la convención de la ortografía utilizada por los controladores de eventos. La primera palabra, en adelante, es todo en minúsculas, y la primera letra de cada palabra en mayúscula. A menos que el caso esté siendo utilizado como un método en un programa JavaScript, que no distingue entre mayúsculas y minúsculas. Ejemplo (`onCliclk` ó `OnClick`)).

Veamos el siguiente ejemplo:

Ejemplo 2.19 Evento `onClic`

```
<html>
  <head><title>Llamada a función levántame</title>
  <script type="text/javascript">
```

```
1 function levantame(){ // La Función es definida aquí
2   setTimeout('alert("Hora de levantarse!")',5000);
3 }
4 </script>
5 </head>
6 <body bgcolor="lightblue">
7   <form>
8     <input type="button"
9       value="Levántame"
10      onClick="levantame()">
11   </form>
12 </body>
13 </html>
```

Explicación:

1. La función levantame() se define en el programa JavaScript, entre las etiquetas <script> </script>. Cuando el usuario hace clic en el botón del formulario, la función asignada al controlador de eventos es llamada. La función es en sí misma definida en un programa JavaScript, aunque se le hace la llamada desde afuera.
2. El temporizador se establece para 5000 milisegundos. El cuadro de diálogo del alert aparecerá en la pantalla 5 segundos después que el usuario hace clic en el botón.
3. Finaliza el programa JavaScript.
4. Finaliza la etiqueta <head>.
5. Se inicia la etiqueta <form>.
6. El tipo de form usado es un “button” input type.
7. El valor que es mostrado en el botón, es el texto “Levántame”.
8. El evento onClic es asignado al nombre de la función. Cuando el usuario hace clic en el botón, el manejador del evento onClic wakeUpCall() será llamado.

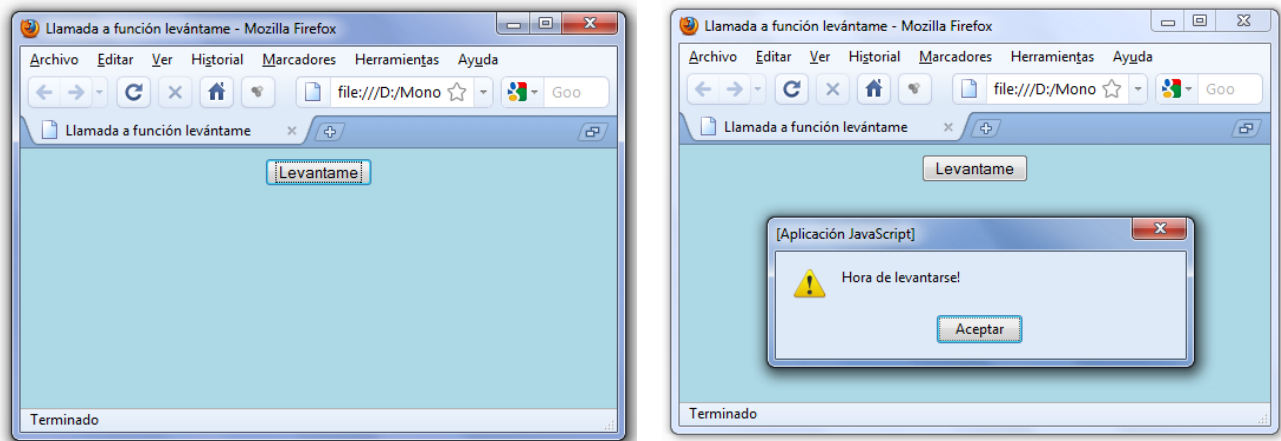


Ilustración 2.17 Antes de hacer clic en el botón del centro; después de hacer clic en el botón y esperar cinco segundos, aparecerá el cuadro de diálogo del alert().

Esta es una lista de los manejadores de eventos de JavaScript y sus usos se muestran en la siguiente tabla:

EventHandler	Se aplica a	Cuando ocurren
onAbort	Imágenes	Cuando la imagen que se está cargando es interrumpida.
onBlur	Con todos los objetos (Windows, frames, etc.)	Cuando el foco se mueve fuera del objeto o sale de la caja de texto.
onChange	Input, Select and Textareas	Cuando el usuario cambia el valor de un elemento y se pierde el foco. Usado para validar formularios.
onClick	Links, buttons, form objects, areas de mapa de imagines.	Cuando un usuario hace clic en un objeto. Retorna falso cuando se cancela la acción.

onDbClick	Links, buttons, form, objects	Al hacer doble clic en un objeto
onDragDrop	Windows	Cuando se deja caer un objeto, tal como un archivo, en un navegador.
onFocus	Script	Cuando un error ocurre en el script, o si hay un error en la sintaxis.
onKeyDown	Documents, images, links, forms.	Cuando se presiona una tecla.
onKeyPress	Documents, images, links, forms	Cuando una tecla es presionada y se suelta.
onKeyUp	Documents, images, links, forms.	Cuando se suelta una tecla.
onLoad	Body, frameset, imágenes	Después que la imagen o documento ha sido cargado.
onMouseOut	Links (y las imágenes enlaces)	Cuando se mueve el mouse fuera de un link.
onMouseOver	Links (y las imágenes enlaces)	Cuando el puntero del mouse se mueve sobre un link. Retorna verdadero para prevenir que el vínculo se muestre en la barra de estado.
onMove	Windows	Cuando el navegador es movido
onReset	Forms reset button	Cuando el botón de reset es clicado. Retorna falso para detener el reset.
onResize	Windows	Cuando el navegador de Windows cambia de tamaño
onSelect	Form elements	Cuando un elemento de un formulario es seleccionado.
onSubmit	Form elements	Cuando se quiere enviar un formulario al servidor. Retorna falso para parar la propuesta del servidor.
onUnload	Body, Framesets	Después que el documento o frameset es cerrado o reseteado.

Tabla 2.2 Manejadores de Evento de JavaScript y lo que hacen.

2.8.2.1 Parámetros para un manejador de evento

Aquí hay dos partes para poner un manejador de eventos:

1. El manejador de evento es asignado como un atributo de una etiqueta HTML tal como un documento, formulario, imagen, o vínculo (link). Si quieres que el evento afecte o actúe en un documento, entonces esto vendría a ser un atributo de la etiqueta <body>; si quieres que el evento afecte a un botón, entonces vendría a ser un atributo de la etiqueta del formulario <input>; y si quieres que el evento afecte a un vínculo, entonces esto viene a ser un atributo en la etiqueta <a href>. Por ejemplo, si queremos que el evento sea activado cuando se termine de cargar el documento, el manejador de evento onLoad se debe usar, y si el evento ocurre cuando el usuario haga clic en un dispositivo input, tal como un botón, el manejador de eventos onClick es el que debemos utilizar.

```
<body onLoad="alert ('Bienvenido a mi sitio web');">
  <form>
    <input type="button" value="Hacerme clic"
      onClick="alert('ja ja ja ja ja ja');"/>
  </form>
</body>
```

2. El siguiente paso es registrar o asignar un valor al manejador de evento. El valor puede estar dentro de un método tal como un alert() o un confirm(), en una función definida por el usuario, o en una cadena declara en JavaScript. Aunque el manejador de evento es un atributo en una etiqueta de HTML, si un usuario lo define en una función, entonces la función deberá ser definida en un programa de JavaScript o directamente en una declaración por separado. Y estará contenido entre comillas. La función manejadora deberá estar encerrada dentro de comillas dobles o simples, y si tiene dobles comillas dentro de la función, salga de la comilla simple con una barra invertida, o rodeándola completamente con llaves.

```
built-in method      - - >      onClick="windows.open('micasa.html', 'newWin')"
```

```
user-defined function - - >      onUnload="timeOver();"
```

group of statements

- - >

```
onChange="if (!checkval (this.value, 1, 10 ) ) {  
this.focus(); this.select();}"
```

Ejemplo 2.20 Utilizando un evento.

```
<html>  
<head><title>Un evento</title></head>  
1 <body bgcolor="magenta" onUnload="alert('Mucho tiempo, desconocido!');">  
  <center>  
2    <form>  
3      <input type="button"  
4      value="Haga clic aquí para ser alertado"  
5      onClick='alert("Tenga cuidado! Un asteroide se está acercando a la  
        tierra!")' />  
6    </form>  
  </center>  
</body>  
</html>
```

Explicación:

1. La etiqueta <body> contiene el manejador de evento onUnload . Cuando el usuario busca en otra página, el método alert() será provocado. Normalmente usted debería usar este evento para una limpieza rápida o salir de una función, tal como cerrar una ventana o limpiar una página. Iniciando por el tiempo consumido, el proceso en este punto sería molesto para el usuario, porque está tratando de salir de esta página sin tener una demora. El único propósito de este ejemplo es demostrar cuando ocurre el evento.
2. El formulario inicia en la etiqueta <form>.
3. El tipo input de este formulario es "button".
4. El valor en el botón es "Haga clic aquí para ser alertado".
5. El evento onClick es un atributo del formulario HTML, en la etiqueta input. Cuando el usuario hace clic con el mouse sobre el botón (El evento onClick), el método alert() es llamado, vea la figura siguiente.
6. Finalmente se alcanza la etiqueta de fin del formulario HTML y termina.

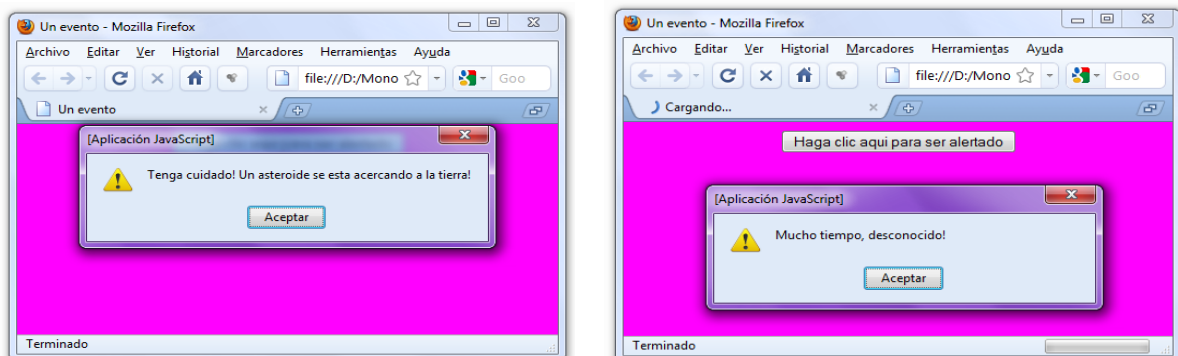


Ilustración 2.18 Cuando el usuario hace clic en el botón, el evento onClick es activado (izquierda); cuando la página es refrescada, el evento onUnload es activado (derecha).

2.8.2.2 Retornando valores

Algunas veces los manejadores de eventos retornan valores si es necesario y una acción segura se ejecuta. Los navegadores por defecto pueden tener acciones que suprimen los retornos de valores a falso, o un informe de un formulario puede ser completado para retornar un valor verdadero. Por ejemplo, si el manejador onSubmit obtiene un valor verdadero retorna al formulario una función o método, entonces un formulario podría enviar al servidor, y si no, el formulario podría estar detenido. Esto se estudió en este capítulo en la parte de la validación de un formulario.

Ejemplo 2.21 Retornando valores con eventos.

```
<html>
  <head><title>Un formulario en HTML y el Manejador de Evento onSubmit </title>
  <script type="text/javascript">
1      function checkForm(Tuinfo){
2          if(Tuinfo.namestring.value == "" ||
            Tuinfo.namestring.value == null){
// Revisando si existe algo en el cajetín o si es un valor nulo
3              alert("Por favor teclee su nombre");
4              return(false);
            }
            else{
5                return(true);
            }
        }
    </script>
  </head>
<body><big>
  <b>
6      <form name="info" action="/cgi-bin/bookstuff/form1.cgi"
          method="post"
7          onSubmit="return checkForm(document.info)">
          <br /><br />
            Teclee su nombre aquí:
8          <input type="text" name="namestring" size="50" />
          <br /><br />
9          <input type="submit" value="Submit" />
          <input type="reset" value="Limpiar" />
        </form>
      </big>
    </body>
  </html>
```

Explicación:

1. La llamada a la función `checkForm()` tiene un argumento, `Tuinfo`, el cual es una referencia del formulario definida en **<form name="info">**.
2. Si el usuario no ingresa ningún texto dentro de la caja de texto, el valor del `input type` será `null`. La expresión `if (Tuinfo.namestring.value == "" || Tuinfo.namestring.value == null)` chequea el campo para saber si está vacío.
3. Si el usuario no ingreso nada en la caja de texto, una caja de dialogo `alert()` aparecerá en la pantalla, y después que el usuario haga clic en el botón Ok, él tendrá una oportunidad para llenar el formulario nuevamente.
4. Si la función retorna el valor falso, el formulario no será enviado al servidor.
5. Si la función retorna verdadero, el formulario será enviado al servidor.
6. El formulario inicia en la etiqueta `form` del documento HTML. El `form`, `document.forms[0]`, se nombre "info". El atributo de acción contiene la URL del programa que procesara el formulario, es un script CGI en el servidor.
7. El manejador de evento `onSubmit` es un atributo en la etiqueta `<form>` de HTML y es provocada cuando el usuario hace clic en el botón submit. El manejador de evento es una función llamada `checkForm()`. El parámetro es el nombre del formulario, `document.info` (también podría usar el arreglo de nombre: `document.forms[0]`). La palabra de retorno es requerida cuando use el manejador de evento `onSubmit`. Uno de los dos valores será retornado: Cualquiera verdadero o falso.
8. El tipo `input` para este formulario es un texto del campo de la caja. Este nombre es "namestring" y puede soportar una cadena de hasta 50 caracteres.
9. El botón submit es de tipo `input`. Cuando el usuario hace clic en el botón, el manejador de evento `onSubmit` en la línea **`onSubmit="return checkForm(document.info)"`** es activado.

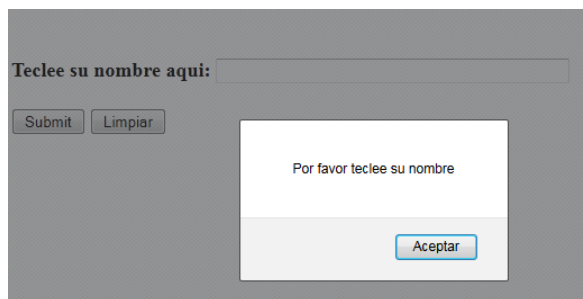


Ilustración 2.19 Utilizando el evento onSubmit y valores de retorno. Si el valor devuelto es true se envía el Formulario, de lo contrario, se detiene.

Métodos de Eventos	Manejadores de Eventos	El objeto que afecta
blur()	onBlur	Remueve el foco de una ventana, frame, etc.
click()	onClick	Simula un click del mouse en los campos de un formulario.
focus()	onFocus	Pone el foco en una ventana, frame , etc.
reset()	onReset	Limpia los campos de un formulario.
Select()	onSelect	Selecciona el texto en el campo de un formulario.
Submit()	onSubmit	Envía un formulario.

Tabla 2.3 Métodos de Eventos de objetos.

2.8.3 Manejo de un evento de ventana o marco

La página web principal es una ventana, a menos que se divida en cuadros. Hay un número de eventos que afectarán a las ventanas y marcos, los cuales se describen en la **Tabla 2.4**. Los siguientes ejemplos ilustran algunos de los eventos que afectan a las ventanas y marcos.

Manejadores de Eventos	Cuando es Invocado
onBlur	Cuando el ratón se mueve fuera de la ventana o marco y pierde el foco.
onFocus	Cuando se hace doble click con el mouse en una ventana o marco y se pierde el foco.
onLoad	Cuando el documento o imagen se finaliza al cargar
onMove	Cuando una ventana es movida.
onUnload	Cuando se cierra o resetea (Actualiza) una página

Tabla 2.4 Eventos de Ventanas y Marcos.

2.8.3.1 Los eventos onLoad y onUnload

El controlador de eventos onLoad se invoca en el momento que se da la ocurrencia de un evento de carga, es decir, cuando un documento, su conjunto de marcos o imágenes han terminado de cargarse por completo. Esto incluye el

punto en el que todas las funciones se han definido y los guiones han sido ejecutados, y todos los formularios están disponibles. Este evento puede ser útil en la sincronización de la carga de un conjunto de marcos, en particular cuando puede haber imágenes de gran tamaño que necesitan ser cargados o todos los marcos de los datos no han llegado desde el servidor.

Ejemplo 2.22 El controlador de eventos `onUnload` se invoca cuando la página se sale o se restablece

```
<html>
<head><title>Eventos load y unload</title>
  <script type="text/javascript">
1  var sec=0;
2  function now(){
    var newdate= new Date();
    var hour=newdate.getHours();
    var minutes=newdate.getMinutes();
    var seconds=newdate.getSeconds();
    var timestr=hour+":"+minutes+"."+seconds;
3  window.setInterval("trackTime()", 1000);
4  alert("Tu documento se ha finalizado de cargar\n" + "En un tiempo de: "+timestr);
  }
5  function trackTime(){
6    sec++;
  }
7  function howLong(){
    alert("Usted ha estado buscando durante "+ sec+" segundos");
  }
  </script>
</head>
8 <body background="onl.jpg" onLoad="now();"
9  onUnload="howLong();">
  <font face="arial,helvetica" size=5> <center>
    Cuando sales o recargas esta página, <br />una caja de diálogo de alert
    aparecerá.
  </font></center>
</body>
</html>
```

Explicación:

1. Una variable global llamada **sec** es declarada.
2. Como usuarios, hemos definido la función `now()`, conteniendo varias objetos de fecha con sus métodos para calcular el tiempo. Esta función se utiliza para hacer un seguimiento del tiempo que el usuario se desplaza desde el momento en que la página se carga hasta que se sale de ella.
3. El objeto ventana `setInterval()` se pone para llamar la función `trackTime()` cada 1,000 milisegundos (1 segundo), iniciando cuando el documento es cargado hasta que es descargado.
4. La caja de diálogo del `alert` aparece cuando la página finaliza de cargarse.
5. La función **`trackTime()`** es definida por el usuario para realizar un seguimiento del número de segundos que ha transcurrido desde que la página se cargó.
6. La variable llamada **sec** se incrementa en una cada vez que **`trackTime()`** es llamada.
7. Esta función es llamada cuando la página se cierra o es recargada, este es el caso provocado por el controlador **`onUnload`** en la línea 9.
8. Cuando el documento se ha terminado de cargar, el controlador de eventos **`onLoad`** se activa. El controlador de eventos **`onLoad`** es un atributo de la etiqueta. El controlador de eventos es asignado a una función llamada **`now()`** que establece un temporizador que se apaga cada segundo mientras la página se abre. Después de un segundo pasa a otra función llamada **`trackTime()`** la cual mantendrá la actualización de una variable que almacena el número de segundos que han transcurrido. En el atributo de fondo de la etiqueta HTML `<body>` se establece en una imagen de para el fondo.

9. El controlador de eventos onUnload se activa cuando el usuario vuelve a recargar la página, tal a como se muestra en la figura 2.20



Ilustración 2.20 Si sales de la página o la recargas, el cuadro del alert aparecerá.

2.8.3.2 Los métodos focus() y blur()

Los métodos focus() y blur() se comportan exactamente como sus nombres lo indican. Estos métodos se aplican a un objeto, como una ventana o el objeto formulario de un documento, y se llaman desde un programa JavaScript. Cuando el método focus() se aplica a un objeto, causará que el objeto tenga el enfoque, y cuando el método blur() se aplica a un objeto, éste perderá su foco.

Ejemplo 2.23 Aplicación de los métodos focus() y blur()

```
<html>
<head><title>Los métodos focus y blur </title>
<script type="text/javascript">
1  function newWindow(){
2    winObj=window.open("summertime.jpg","summer","width=350,height=200,resizable=yes,
scrollbars=yes,location=yes");
3    winObj.moveTo(0,0); // Mover a la esquina izquierda de la pantalla
4    winObj.focus(); // Para que la nueva ventana obtenga el foco
    winObj.blur();
  }
5  function closeWindow(){
6    winObj.close(); // Cerrar la nueva ventana
  }
</script>
</head>
<body bgColor="lightgreen">
<h2>Escena de verano de la Madre Patria</h2>
<form>
7  <input type=button value="Abrir ventana" onClick="JavaScript:nuevaVentana();" />
7  <input type=button value="Cerrar ventana" onClick="JavaScript:closeWindow();" />
</form>
</body>
</html>
```


Explicación:

1. Una función definida por el usuario llamada `newWindow()` crea un nuevo objeto ventana con el método `open()`, se especifica un número de opciones para definir mejor la ventana **`function newWindow()`**.
2. El objeto nueva ventana contiene una imagen llamada `summertime.jpg`.
3. La nueva ventana aparecerá en la esquina superior izquierda, que corresponde con la posición del pixel (0,0).
4. La nueva ventana recibe el foco **`winObj.focus()`**. Estará por encima de todas las otras ventanas.
5. La función definida por el usuario **`function closeWindow()`**, es responsable de cerrar la nueva ventana.
6. El método `close()` del objeto de la ventana hace que la nueva ventana sea cerrada **`winObj.close()`**.
7. Cuando el usuario hace clic en el botón, el manejador de evento `onClick` se dispara, y la nueva ventana aparecerá **`onClick="JavaScript:newWindow()`**.
8. Cuando el usuario hace clic en este botón, el manejador de evento `onClick` se dispara y la nueva ventana se cerrará.



Ilustración 2.21 Esta es la Ventana Padre.



Ilustración 2.22 Esta es la nueva ventana que aparecerá en la parte superior izquierda de la pantalla.

2.8.4 Gestión de eventos del mouse.

En los ejemplos anteriores, hemos visto los usos del controlador de eventos OnClick para iniciar una acción cuando un usuario hace clic con el ratón en un botón o en un enlace. Hay una serie de otros eventos que pueden ser despididos por alguna acción del ratón. Cuando el usuario mueve el puntero del ratón sobre un enlace, imagen u otro objeto, el controlador de eventos onMouseOver se activa, y cuando el usuario mueve el puntero del ratón fuera de un objeto, el evento onMouseOut se dispara. En la Tabla 2.5 se enumeran los eventos que se desencadenan cuando se detecta el movimiento del ratón.

Manejadores de Eventos	Cuando es Invocado
onClick	Cuando se hace click con el ratón sobre un vínculo o un botón de un formulario.
onDbclick	Cuando se hace doble click con el mouse en un vínculo, documento, formulario o imagen.
onMouseDown	Cuando el click es presionado en un vínculo o documento.
onMouseMove	Cuando el ratón es movido por encima de un vínculo, objeto, o más elementos.
onMouseOut	Cuando el ratón fuera de un de un vínculo o mapa de imágenes.
onMouseOver	Cuando el ratón es movido encima de un vínculo, mapa de imágenes o más elementos.
onMouseUp	Cuando se suelta el ratón de un vínculo o documento.

Tabla 2.5 Eventos del ratón.

2.8.5 Como utilizar los eventos del ratón

Los manejadores de eventos onMouseOver y onMouseOut se producen cuando el usuario mueve el puntero del ratón sobre o fuera de un objeto. El evento onMouseMove se produce cuando el ratón sólo toca el objeto.

Ejemplo 2.24 Eventos del ratón.

```
<html>
<head><title>Eventos del rat&ocute;n</title>
1  <script type="text/javascript">
2      var counter=0;
3      function alertme(){
4          alert("Me voy de paseo, es hora de cerrar!");
5          window.close();
6      }
7      function track_Moves(){
8          counter++;
9          if(counter==1){
10             alert(counter + " El ratón se movió !");
11         }
12         else{
13             alert(counter + " El ratón se movió!");
14         }
15     }
16 </script>
17 </head>
18 <body bgColor="CCFF00" onDbClick="alertme()">
19     <p><font face="arial" size=3>
20         Haga doble clic en cualquier parte de esta página para salir de ella!
21     </p><p>
```

```
8      Cuando el ratón se mueve fuera del vínculo, un evento es disparado.
      <a href="#" onMouseOver="alert('Event:onMouseOver');" />
      onMouseOver
      </a></p><p>

      Cuando el ratón se mueve fuera de un enlace, se activa un evento.
      <a href="#" onMouseOut="alert('Event:onMouseOut');" />
      onMouseOut
      </a></p><p>
      Cuando el ratón se mueve dentro o fuera del botón, una función<br />
      es llamada y mantiene un registro de cuántas veces el ratón toca
      el botón.
9      <form>
10     <input type="button" value="onMouseMove"
      onMouseMove="track_Moves();" />
      </form>
      </p>
</body>
</html>
```

Explicación:

1. El programa Script inicia con la etiqueta <script>.
2. Se inicializa la variable global llamada counter.
3. Si el usuario hace doble clic en la página, un cuadro de diálogo de alert aparece, si el usuario hace clic en el botón Aceptar de la ventana del alert se cerrara.
4. El método window.close(), hace que la ventana actual se cierre.
5. Esta función se llama cuando el controlador de eventos onMouseOver se dispara. Este evento ocurre cuando el usuario toca el ratón sobre un objeto, en este caso, el botón objeto.
6. El controlador se incrementa en uno cada vez que el usuario toque el botón.
7. El controlador de eventos ondblclick es un atributo de la etiqueta <body> HTML. Cuando el usuario hace doble clic en el ratón, la función alertMe() será llamada, y la ventana se cerrara.
8. El controlador de eventos onMouseOver es un atributo de la etiqueta <a href>. Se activa en cualquier momento en el que el usuario mueva el ratón sobre el enlace. Al producirse el evento, se método llama a un alert.
9. El formulario inicia con la etiqueta <form>, el tipo de entrada es un botón.
10. Cuando el usuario toca con el ratón el botón, el controlador de eventos onMouseMove es disparado, y la función track_Moves() es llamada. Esta función simplemente incrementa la variable contador en uno cada vez que se llama, y luego alerta al usuario.

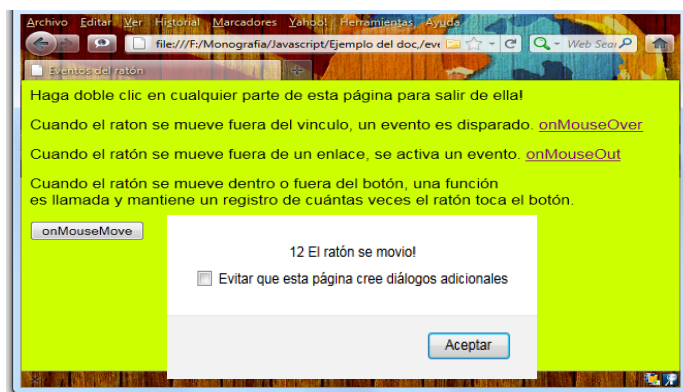
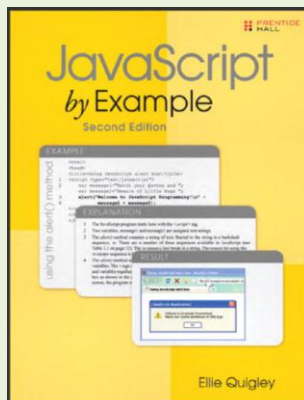


Ilustración 2.23 Enlaces y manejadores de eventos.

Bibliografía del Capítulo

JavaScript by Example. Ellie Quigley.





Capítulo 3

DOM



3 DOM (Document Object Model)

Desafortunadamente, las posibilidades teóricas de DOM son mucho más avanzadas de las que se pueden utilizar en la práctica para desarrollar aplicaciones web. El motivo es que el uso de DOM siempre está limitado por las posibilidades que ofrece cada navegador. Mientras que algunos navegadores como Firefox y Safari implementan DOM de nivel 1 y 2 (y parte del 3), otros navegadores como Internet Explorer (versión 7 y anteriores) ni siquiera son capaces de ofrecer una implementación completa de DOM nivel 1.

El DOM (modelo de objeto de documento) es una manera de representar un XML o un documento de HTML como un árbol de nodos. Usando los métodos y propiedades de DOM, usted puede acceder a cualquier elemento de la página, modificar o remover elementos, y agregar nuevos. El DOM es un lenguaje independiente API (Interfaz de Programación de Aplicaciones) y pueden estar implementadas no únicamente en JavaScript, sino para cualquier otro lenguaje. Por ejemplo, usted puede generar páginas en el servidor implementando PHP con DOM.

El Modelo de Objeto de Documento no puede funcionar sin un documento. Cuando se crea una página Web cargada en un buscador web, el DOM se mueve en tiempo de vida, toma el documento que usted ha escrito y lo convierte en un objeto.

En condiciones normales de cada día en inglés, la palabra “objeto” no es muy descriptiva. En lenguaje de programación, por el contrario, “objeto” tiene un significado muy específico.

Los objetos son auto-contenedores de paquetes de datos. Las variables asociadas con un objeto se denominan propiedades del objeto, mientras que las funciones que pueden ser ejecutadas por un objeto se llaman métodos del objeto.

Hay tres tipos de objetos en JavaScript:

- Objetos definidos por el usuario, creados por el programador.
- Objetos nativos, como arreglos, matemáticas y las fechas que se construyen para JavaScript.
- Objetos de Host que son proporcionados por el navegador.

Desde los primeros días de JavaScript, algunos objetos muy importantes se han hecho disponibles para scripting. El más fundamental de ellos es el objeto de ventana.

Este objeto es una representación de la ventana del navegador en sí. Las propiedades y métodos del objeto Windows se refieren a menudo al Explorador de modelos de objetos.

El explorador de modelo de objetos tiene métodos como `window.open` y `window.blur`. Estos métodos, por cierto, son responsables de todos esos molestos pop-up y pop-under que funcionan con ventanas y que ahora son una plaga en la Web. No es de extrañar porque JavaScript tiene una mala reputación!

Afortunadamente, no vamos a estar tratando con el explorador de modelos de objetos, en su lugar nos centraremos en lo que hay dentro de la ventana del buscador. El objeto que se encarga de los contenidos de una página web, es el objeto document.

Trataremos con las propiedades y métodos del objeto document, eso explica la letra D (documento) y la letra O (objeto) en el DOM. Pero ¿qué pasa con la letra M?

Marcar M para modelo.

La M en DOM representa modelo, pero podría fácilmente usarse para mapa. Un modelo, como un mapa, es una representación de algo. Un modelo de tren representa un tren real. Un plano de las calles de una ciudad representa una ciudad real. El DOM representa la página web que está cargada actualmente en la ventana del navegador. El navegador proporciona un mapa (o modelo) de la página. Usted puede usar JavaScript para leer este mapa.

Los mapas hacen uso de las convenciones así como de la dirección, curvas de nivel, y la escala. El orden para poder leer un mapa, es necesario que comprenda estas convenciones, que es lo mismo con el modelo de objetos de

documento. Con el fin de obtener información del modelo, es necesario comprender que convenciones se están usando para representar el documento.

La convención más importante utilizada para representar un documento es la representación de un documento como un árbol. El documento se representa como un árbol genealógico.

Una familia de árboles es otro ejemplo de modelo. Una familia de árbol representa una familia real, describe las relaciones entre los miembros de la familia, y utiliza convenciones, como padre, hijo, y hermano. Estos pueden ser usados para representar algunas relaciones muy complejas; un miembro de una familia puede ser un padre de otro, al mismo tiempo ser el hijo de otra familia miembro y ser el hermano miembro de otra familia.

El modelo de árbol genealógico funciona igual de bien en la presentación de un documento escrito en (x) html.

Echemos un vistazo a esta página web muy básica:

Ejemplo 3.1 Diseño Web con JavaScript y DOM.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <meta http-equiv="content-type" content="text/html;
      charset=utf-8" />
    <title>Lista de compras</title>
  </head>
  <body>
    <h1>¿Qué Comprar?</h1>
    <p title="un gentil recordatorio">No olvide comprar estas cosas.</p>
    <ul id="compras">
      <li>Una lata de frijoles</li>
      <li>Queso</li>
      <li>Leche</li>
    </ul>
  </body>
</html>
```

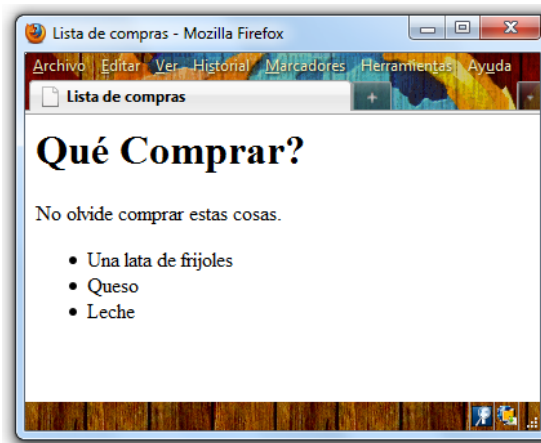


Ilustración 3.1 Diseño Web con JavaScript y DOM.

Este modelo se puede usar para representar la ilustración 3.1

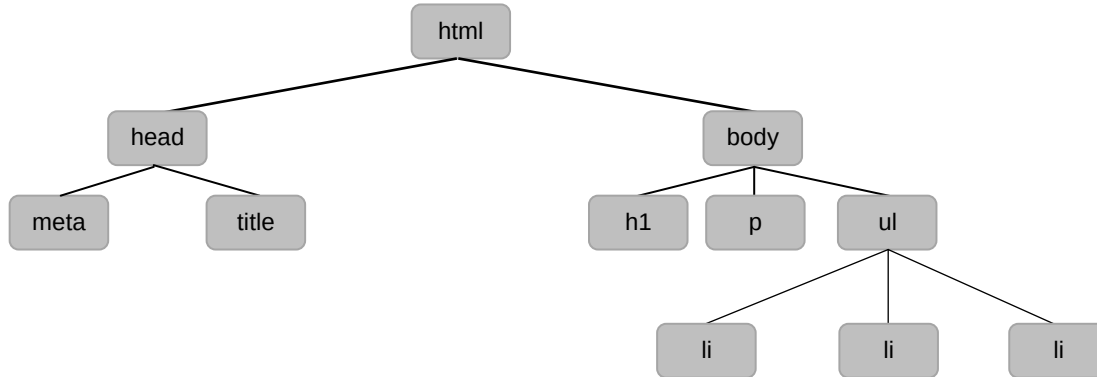


Ilustración 3.2 Árbol de elementos de una página web básica.

La imagen siguiente se tomó de la herramienta de DOM inspector de firefox. Esta herramienta no se instala de forma predeterminada, por lo que si está ejecutando Firefox es posible que tenga que instalar la herramienta. El inspector de DOM se puede conseguir como un add-on de: <https://addons.mozilla.org/es-ES/firefox/search/?q=DOM&cat=all&x=0&y=0>

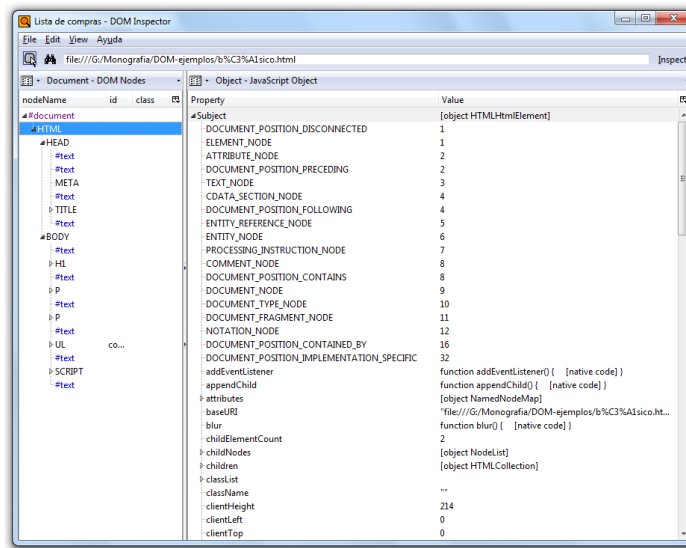


Ilustración 3.3 Una vez instalado, el Inspector DOM es accesible a través de La barra de Herramientas --> DOM Inspector.

En el inspector de DOM, usted tiene el árbol de DOM en el marco de la izquierda y cuando selecciona un nodo, en el marco derecho aparece la información para el nodo seleccionado, basta con hacer clic sobre el nodo.

Ahora caminaremos a través de la estructura de la página web para ver de qué se compone, y para demostrar porque es tan bien representada por el modelo de árbol de la ilustración 3.2.

El documento comienza por la apertura de la etiqueta <html>. Todos los demás elementos de la página web están contenidos dentro de este elemento, lo que significa que es uno de los padres. Debido a que todos los demás elementos se encuentran dentro, la etiqueta <html> no tiene en sí los padres. Además, no tiene hermanos. Si esto fuera un árbol, la etiqueta <html> sería la raíz.

El elemento raíz es `html`. Para todos los efectos, el elemento HTML es el documento.

Si nos acercamos a un nivel más profundo, nos encontramos con dos ramas: `<head>` y `<body>`. Existe uno al lado del otro, esto los hace hermanos. Ellos comparten el mismo padre, `<html>`, pero también ambos contienen hijos, así que ellos son padres propios.

El elemento `<head>` tiene dos hijos: `<meta>` y `<title>` (hermanos entre sí). Los hijos del elemento `<body>` son `<h1>`, `<p>` y `<ul>` (todos hermanos). Si profundizáramos aún más, nos encontramos con que `<ul>` es también uno de los padres. Tiene tres hijos, todos elementos `<li>`.

Al usar una simple convención de las relaciones familiares, podemos acceder a mucha información sobre la relación entre los elementos.

Por ejemplo;

¿Cuáles la relación entre `<h1>` y `<p>`? La respuesta es que son hermanos.

¿Cuál es la relación entre `<body>` y `<ul>`? `<body>` es el padre de `<ul>`. `<ul>` es un hijo de `<body>`.

Si puede pensar en los elementos de un documento en términos de un árbol de relaciones familiares, entonces usted está utilizando los mismos términos que el DOM. Sin embargo, en lugar de utilizar el término “árbol de familia”, “es más preciso llamar a un documento de un árbol de nodos.

3.1 Nodos

El término nodo proviene de las redes, donde se utiliza para denotar un punto de conexión en una red. Una red es un conjunto de nodos. En el mundo real, todo está hecho de átomos. Los átomos son los nodos del mundo real. Pero los átomos pueden dividirse en partes más pequeñas, las partículas subatómicas. Estas partículas también se consideran nodos.

Ocurre algo similar con el modelo de objetos de un documento. Un documento es una colección de nodos, con nodos en las ramas y hojas en el árbol del documento. Hay un número de diferentes tipos de nodos. Al igual que los átomos contienen partículas subatómicas, algunos tipos de nodos contienen otros tipos de nodos.

3.1.1 Elementos de nodos

El equivalente del átomo del DOM es el nodo elemento.

Cuando se describe la estructura del documento de la lista de la compra, se hace en términos de elementos tales como `<body>`, `<p>` y `<ul>`. Los elementos son los componentes básicos de los documentos en la web, y es la disposición de estos elementos en un documento la que le da su estructura.

La etiqueta proporciona el nombre de un elemento. Los elementos de párrafos tienen el nombre de `p`, las listas desordenadas tienen el nombre de `ul`, y los elementos de la lista tienen el nombre de `li`.

Los elementos pueden contener otros elementos. Todos los elementos de la lista de nuestro documento figuran dentro de un elemento de la lista no ordenada. De hecho, el único elemento que no figura dentro de otro elemento es el elemento `<html>`. El cuál es la raíz de nuestro árbol de nodos.

3.1.2 Nodos de texto

Los elementos son sólo un tipo de nodo. Si un documento consistía únicamente de elementos vacíos, que tendrían una estructura, pero el documento no contiene mucho contenido. En la web, donde el contenido es el rey, la mayoría del contenido es proporcionado por el texto.

En nuestro propio ejemplo, el elemento `<p>` contiene el texto “No te olvides de comprar estas cosas”. Este es un texto del nodo. En html, los nodos de texto son siempre encerrados dentro de los nodos de elementos. Pero no todos los elementos contienen los nodos de texto. En nuestro documento de lista de la compra, el elemento `<ul>` no contiene ningún texto directamente. Contiene otros nodos elementos (el elemento `<li>`), y estos contienen los nodos de texto.

3.1.3 Atributos de nodos

Los atributos se utilizan para dar información más específica sobre un elemento. El atributo de título, por ejemplo, se puede utilizar en casi cualquier elemento para especificar exactamente lo que el elemento contiene:

```
<p title="un gentil recordatorio">No se olvide de comprar estas cosas.</p>
```

En el DOM, `title = “un gentil recordatorio”` es un nodo de atributo, como muestra en la siguiente Ilustración 3.1.3.1. Debido a que los atributos siempre se colocan dentro de las etiquetas de apertura, los atributos de los nodos están siempre contenidos dentro de los nodos de elemento. No todos los elementos contienen atributos, pero todos los atributos están contenidos por los elementos.

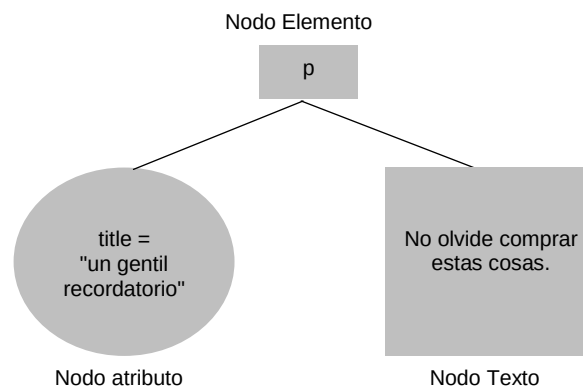


Ilustración 3.1.3.1 Un nodo de elemento que contiene un nodo de atributo y un nodo de texto.

En nuestro documento de ejemplo, verá que la lista desordenada (`<ul>`) ha sido marcada con el atributo `id`. Probablemente usted está familiarizado con los atributos `id` y `class` usados en CSS.

Además del tipo de etiqueta HTML y su contenido de texto, DOM permiten el acceso directo a todos los atributos de cada etiqueta. Para ello, los nodos de tipo Element contienen la propiedad `attributes`, que permite acceder a todos los atributos de cada elemento. Aunque técnicamente la propiedad `attributes` es de tipo `NamedNodeMap`, sus elementos se pueden acceder como si fuera un array. DOM proporciona diversos métodos para tratar con los atributos:

- `getNamedItem(nombre)`, devuelve el nodo cuya propiedad `nodeName` contenga el valor `nombre`.
- `removeNamedItem(nombre)`, elimina el nodo cuya propiedad `nodeName` coincida con el valor `nombre`.
- `setNamedItem(nodo)`, añade el nodo a la lista `attributes`, indexándolo según su propiedad `nodeName`.
- `ítem(posición)`, devuelve el nodo que se encuentra en la posición indicada por el valor numérico `posición`.

Los métodos anteriores devuelven un nodo de tipo `Attr`, por lo que no devuelven directamente el valor del atributo. Afortunadamente, DOM proporciona otros métodos que permiten el acceso y la modificación de los atributos de forma más directa:

- `getAttribute(nombre)`, es equivalente a `attributes.getNamedItem(nombre)`.
- `setAttribute(nombre, valor)` equivalente a `attributes.getNamedItem(nombre)`.

value= valor.

- removeAttribute(nombre), equivalente a attributes.removeNamedItem(nombre).

3.1.4 Acceso directo a los nodos

Los métodos presentados hasta el momento permiten acceder a cualquier nodo del árbol de nodos DOM y a todos sus atributos. Sin embargo, las funciones que proporciona DOM para acceder a un nodo a través de sus padres obligan a acceder al nodo raíz de la página y después a sus nodos hijos y a los nodos hijos de esos hijos y así sucesivamente hasta el último nodo de la rama buscado.

Cuando se trabaja con una página web real, el árbol DOM tiene miles de nodos de todos los tipos. Por este motivo, no es eficiente acceder a un nodo descendiendo a través de todos los ascendentes de ese nodo.

Para solucionar este problema, DOM proporciona una serie de métodos para acceder de forma directa a los nodos deseados. Los métodos disponibles son: getElementByTagName(), getElementByName() y getElementById().

3.1.4.1 getElementById

El DOM tiene un método llamado getElementById, que hace exactamente lo que dice, permite llegar directamente al nodo de un elemento con el **id** especificado. Recordemos que JavaScript distingue entre mayúsculas y minúsculas, por lo que getElementById debe ser escrito con mayúsculas y minúsculas preservadas.

Si escribes este método así **getElementById** o **getElementbyId**, no conseguirás los resultados que esperas. Este método es una función asociada con el objeto document. Las funciones van siempre seguidas de paréntesis que contienen los argumentos de la función. getElementById toma un solo argumento; el id del elemento al que quieres llegar, contenido entre comillas simples o dobles.

```
document.getElementById(id)
```

Aquí está un ejemplo:

```
document.getElementById("compras")
```

Esto hace referencia al elemento único que ha asignado html id "compras" en el objeto document. Este elemento también corresponde a un objeto. Usted puede probar esto por sí mismo mediante el operador typeof. Esto le dirá si algo es una cadena, un número, una función, un valor booleano, o un objeto.

No recomendamos esta forma de JavaScript para añadir un documento, sino, exclusivamente para las pruebas a propósito. Inserte este fragmento de código JavaScript en el documento de la lista de compra. Póngalo antes de la etiqueta de cierre </body>:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8" />
    <title>Lista de compras</title>
  </head>
  <body>
    <h1>¡Qu&eacute; Comprar?</h1>
    <p title="un gentil recordatorio">No olvide comprar estas cosas.</p>
    <ul id="compras">
      <li>Una lata de frijoles</li>
      <li>Queso</li>
    </ul>
  </body>
</html>
```

```
</li>Leche</li>
</ul>

<script type="text/javascript">
    alert(typeof document.getElementById("compras"));
</script>

</body>
</html>
```

Al cargar el archivo xhtml nuevamente en el navegador web, le dará la bienvenida con un molesto cuadro emergente indicando la naturaleza de `document.getElementById("compras")`. Se trata de un objeto.



Ilustración 3.1.4.1 Alert() mostrando la naturaleza del nodo element.

De hecho, cada elemento en un documento es un objeto. Usando el DOM, usted puede obtener cada uno de estos elementos. Es obvio que no debería dar una identificación única para cada elemento en un documento. Que sería una exageración. Afortunadamente, el objeto `document` proporciona otro método para obtener elementos que no tienen identificadores únicos.

3.1.4.2 `getElementsByTagName`

Si usted utiliza el método `getElementsByTagName`, usted tiene acceso instantáneo a una amplia gama de población con cada aparición de una etiqueta especificada. Al igual que `getElementById`, esta es una función que toma un argumento. En este caso, el argumento es el nombre de una etiqueta:

```
element.getElementsByTagName(etiqueta)
```

Se ve muy similar a `getElementById`, pero hay que tener en cuenta que esta vez se puede obtener elementos en plural. Tenga cuidado al escribir sus scripts para que no escriba `getElementById` en lugar de `getElementsByTagName`.

Aquí está en acción:

```
Document.getElementsByTagName("li")
```

Este es un arreglo de población con todos los elementos de la lista del objeto `document`. Al igual que con cualquier otro arreglo, usted puede usar la propiedad `length` para obtener el número total de elementos contenido en el arreglo.

Elimine la declaración de `alert()` que coloco entre las etiquetas `<script>` anteriormente y sustitúyalo por la siguiente:

```
alert(document.getElementsByTagName("li").length);
```

Esto le dirá el número total de elementos de la lista en el documento: tres, en este caso, todos los valores de la matriz son un objeto. Usted puede probar esto por sí mismo a través de la matriz y usando `typeof` en cada valor. Por ejemplo, intente esto con un bucle `for`:

```
for (var i=0; i < document.getElementsByTagName("li").length; i++) {  
    alert(typeof document.getElementsByTagName("li")[i]);  
}
```

Incluso si sólo hay un elemento con el nombre de etiqueta especificado, todavía `getElementsByTagName` devuelve una matriz. La longitud de la matriz será simplemente una.

Ahora comenzará a notar que empieza a ser tedioso escribir a `document.getElementsByTagName("li")` cada vez, y que el código está empezando a ser difícil de leer. Puede reducir la cantidad de escritura innecesaria y mejorar la legibilidad mediante la asignación de una variable para contener `document.getElementsByTagName("li")`.

Vuelva a colocar la declaración de `alert()` entre las etiquetas `<script>` con estas declaraciones:

```
var items = document.getElementsByTagName("li");  
for (var i=0; i < items.length; i++) {  
    alert(typeof items[i]);  
}
```

Ahora tendrá tres cajas de `alert()` molestas, cada una de ellas diciendo lo mismo.

También puede utilizar un comodín con `getElementsByTagName`, lo que significa que usted puede hacer una matriz con cada elemento. El símbolo de comodín (el asterisco) debe estar entre comillas para distinguirla del operador de multiplicación. El comodín le dará el número total de nodos de elementos de nodos del documento.

```
alert(document.getElementsByTagName("*").length);
```

También se puede combinar `getElementsByTagName` con `getElementById`, hasta ahora sólo hemos aplicado `getElementsByTagName` al objeto del documento, pero si usted está interesado en la búsqueda de cuántos elementos se encuentran dentro de la lista, con el id “compras”, podría aplicarse `getElementById` al objeto específico.

```
var compras = document.getElementById("compras");  
var articulos = compras.getElementsByTagName("*");
```

Ahora la matriz de elementos contiene sólo los elementos contenidos en la lista “compras”. En este caso, pasa a ser el mismo que el número total de los elementos de la lista en el documento.

```
alert (articulos.length);
```

Si necesita alguna otra prueba, usted puede probar que cada uno es un objeto.

```
for (var i=0; i < items.length; i++) {  
    alert(typeof items[i]);  
}
```

Como ha podido ver, cada nodo de elemento en un documento es un objeto. No sólo eso, sino que todos y cada uno de esos objetos vienen con un arsenal de métodos, cortesía de DOM. Estos métodos podemos usarlos para recuperar información de cualquier elemento de un documento, incluso podemos alterar las propiedades de un elemento.

Este es un pequeño resumen de lo que hemos visto hasta esta parte:

- Un documento es un árbol de nodos.
- Existen diferentes tipos de nodos; elementos, atributos, textos y así sucesivamente.
- Puedes acceder directamente a un nodo de elemento específico usando `getElementById`.
- Puedes ir directamente a una colección de nodos de elementos con `getElementsByTagName`.

- Cada uno de estos nodos son un objeto.

3.2 Atributos HTML y propiedades CSS en DOM

Los métodos presentados anteriormente para el acceso a los atributos de los elementos, son genéricos de XML. La versión de DOM específica para HTML incluye algunas propiedades y métodos aún más directos y sencillos para el acceso a los atributos de los elementos HTML y a sus propiedades CSS.

La principal ventaja del DOM para HTML es que todos los atributos de todos los elementos HTML se transforman en propiedades de los nodos. De esta forma, es posible acceder de forma directa a cualquier atributo de HTML.

3.2.1 `getAttribute`

Hasta ahora, hemos visto dos formas diferentes de llegar a los nodos de elementos, ya sea utilizando `getElementById` o `getElementsByTagName`. Cuando ya tenemos el elemento, podemos encontrar los valores de cualquiera de sus atributos. Usted puede hacer esto con el método **`getAttribute`**.

`getAttribute` es una función. Toma solo un argumento – el atributo que tú quieres obtener.

`object.getAttribute(attribute)`

A diferencia de los otros métodos que hemos visto, no se puede utilizar **`getAttribute`** en el documento objeto. Sólo se puede utilizar en un objeto de nodo elemento.

Por ejemplo, se puede utilizar en combinación con `getElementsByTagName` para obtener el título atributo de cada elemento `<p>`:

```
var paras = document.getElementsByTagName("p");
for (var i=0; i < paras.length; i++) {
    alert(paras[i].getAttribute("title"));
}
```

Si incluye este código al final de nuestra lista de compras del documento y después cargamos la página en el navegador, le dará la bienvenida con un cuadro de `alert()` conteniendo el texto, “un suave recordatorio”.

En nuestra lista de compras, hay un solo elemento `<p>` y tiene un atributo de título. Si había más elementos `<p>` y que no tenían los atributos título a continuación, `getAttribute("title")` devolverá el valor `null`. En JavaScript, nulo significa que no hay ningún valor. Puedes probar esto por ti mismo mediante la inserción de la siguiente línea:

`<p>Esto es solo una prueba</p>`

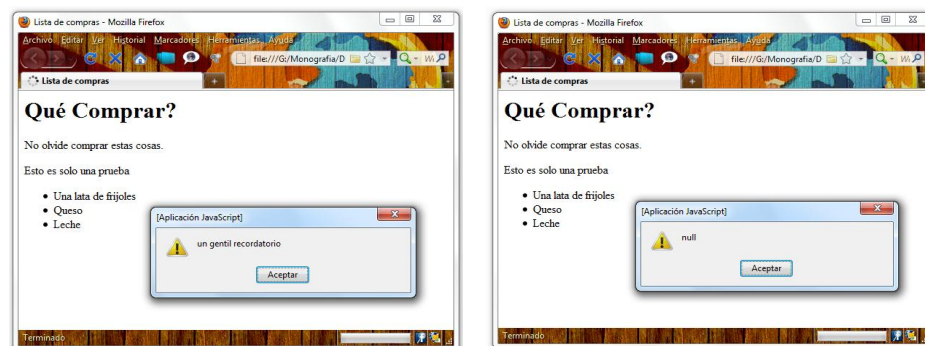


Ilustración 3.2.1 El segundo `alert()`, está totalmente vacío o simplemente dice "nulo", dependiendo de cómo el explorador elige mostrar los valores nulos.

Nosotros podemos modificar el script para que solo aparezca un mensaje cuando exista solo un atributo. Bastara con agregar una sentencia if para comprobar que el valor devuelto por `getAttribute` no es nulo. Para hacer esto utilizaremos algunas otras variables para hacer más entendible el código.

```
var paras = document.getElementsByTagName("p");
for (var i=0; i< paras.length; i++) {
    var title_text = paras[i].getAttribute("title");
    if (title_text != null) {
        alert(title_text);
    } //Cierre del if
} // Cierre del for
```

Si ahora volvemos a cargar la página, sólo veremos un cuadro de `alert()` que contiene el valor “un suave recordatorio”.

Podemos acortar el código aún más. Siempre que quiera comprobar que algo existe o que no está nulo, en realidad lo que se comprueba es para ver si algo existe. como la condición en una sentencia if. If (algo) es una forma abreviada de la escritura if (algo!=null). La condición de la sentencia if será verdadera si algo existe, y será falsa si no existe.

Entonces podemos reforzar nuestro código simplemente escribiendo `if(title_texto)` en lugar de `if(title_text!=null)`. Mientras estamos en ello, podemos poner la declaración de `alert` en la misma línea con la instrucción if de modo que se lea así:

```
var paras = document.getElementsByTagName("p");
for (var i=0; i< paras.length; i++) {
    var title_text = paras[i].getAttribute("title");
    if (title_text) alert(title_text);
}
```

3.2.2 `setAttribute`

Todos los métodos que hemos visto hasta ahora se han ocupado de la recuperación de información. **`setAttribute`** es un poco diferente. Le permite cambiar el valor de un atributo a un nodo. Al igual que `getAttribute`, este método es una función que sólo funciona en los nodos del elemento. Sin embargo, `setAttribute` toma dos argumentos.

```
object.setAttribute(attribute,value)
```

En este ejemplo, vamos a obtener el elemento con el id “compras”, y darle un título atributo con el valor de “una lista de bienes”:

```
var compras = document.getElementById("compras");
compras.setAttribute("title","una lista de bienes");
```

Ahora podemos utilizar `getAttribute` para probar que el atributo título se ha establecido:

```
var compras = document.getElementById("compras");
alert(compras.getAttribute("title"));
compras.setAttribute("title","una lista de bienes");
alert(compras.getAttribute("title"));
```


Al cargar la página nuevamente se mostrarán dos cajas de alert():

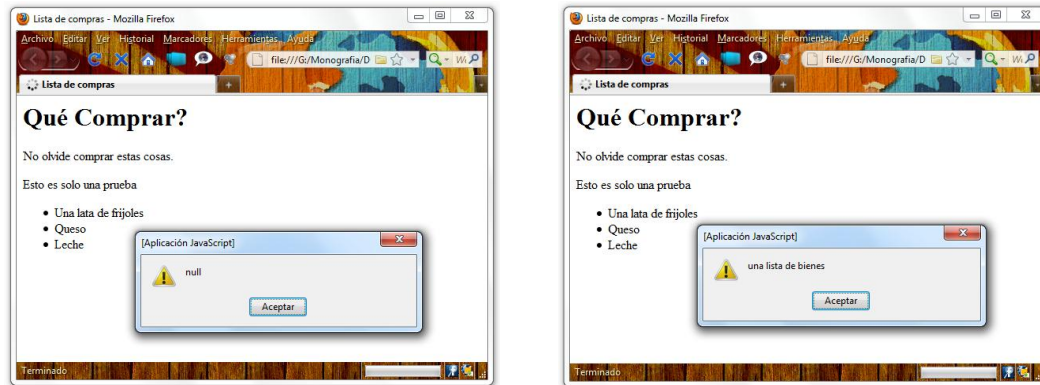


Ilustración 3.2.2.1 La primera, que se ejecuta antes de `setAttribute`, se muestra vacía o "nulo". La segunda, se ejecuta después de que se ha establecido el atributo de título, dice "una lista de bienes"

En este ejemplo, nosotros podemos establecer un atributo que previamente no existía. El método `setAttribute` crea el atributo y establece su valor. Si utiliza `setAttribute` en un elemento de nodo que ya tiene especificado uno, el valor anterior se sobrescribirá.

En nuestro documento de lista de compra, el elemento `<p>` ya tiene un atributo con un título valor "un recordatorio". Se utiliza `setAttribute` para cambiar ese valor.

```
var paras = document.getElementsByTagName("p");
for (var i=0; i< paras.length; i++) {
    var title_text = paras[i].getAttribute("title");
    if (title_text) {
        paras[i].setAttribute("title"," Nuevo título de marca de texto");
        alert(paras[i].getAttribute("title"));
    } // cierre del if
} // cierre del for
```

Se aplicará al valor "nuevo título de marca de texto" para el atributo título de cada elemento `<p>` en el documento que ya tenía un atributo de título. En nuestra lista de compra del documento, el valor "un recordatorio" se ha sobrescrito.

Vale la pena señalar que, aun cuando un documento ha sido cambiado por `setAttribute`, no verá el cambio si utiliza la opción de ver el código fuente de la web en el navegador. Esto se debe a que DOM actualiza dinámicamente el contenido de la página después de que se ha cargado. El poder real del DOM es que el contenido de una página puede ser actualizada sin refrescar la página en el navegador.

Hemos mostrado cuatro métodos proporcionados por el DOM:

- `getElementById`.
- `getElementsByTagName`
- `getAttribute`
- `setAttribute`

Estos cuatro métodos serán los pilares básicos para muchas de las secuencias de comandos DOM que va a escribir.

3.2.3 Hermanos, cuerpo, primer y último hijo.

El `nextSibling` y `previousSibling` son otras dos propiedades muy prácticas para navegar por el árbol DOM, una vez que haya una referencia a un elemento:

```
<<< var para = document.getElementById ('Closer')
>>> para.nextSibling

“\n”

>>> para.previousSibling

“\n”

>>> para.previousSibling.previousSibling

<p>

>>> para.previousSibling. previousSibling. previousSibling

“\n”

>>> para.previousSibling.previousSibling.nextSibling.nextSibling

<p id="closer">
```

Los elementos del cuerpo `<body>` se utilizan tan a menudo que tiene su propio atajo:

```
>>> document.body

<body>

>>> document.body.nextSibling

Null

>>> document.body.previousSibling

<head>
```

El `firstChild` y `lastChild` también podrían ser útiles. El `firstChild` es el mismo que `childNodes [0]` y `lastChild` es el mismo que `childNodes.length – 1`.

```
>>> document.body.firstChild

“\n”

>>> document.body.lastChild

“\n”

>>> document.body.lastChild.previousSibling

Comment length = 21 nodeName=#comment

>>> document.body.lastChild.previousSibling.nodeValue
```

La siguiente ilustración muestra las relaciones familiares entre el cuerpo y tres párrafos en el mismo. Para simplificar, se quitan todos los espacios en blanco y los nodos de texto han sido removidos de este diagrama.

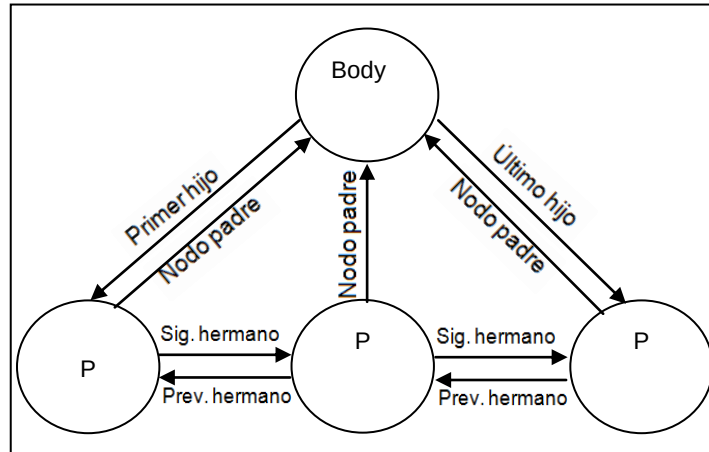


Ilustración 3.2.3.1 Relación entre la familia del árbol.

3.2.4 Creando un blog

En los ejemplos anteriores, se muestra cómo utilizar las propiedades y métodos DOM, en este ejemplo estaremos agregando texto dinámicamente a una página web (**Crearemos un blog**). Al usuario se le presenta un área para que escriba en el blog utilizando métodos DOM para definir la estructura de una nueva entrada en el documento. Cada entrada del blog se añadirá a la anterior, fecha de publicación del blog y el mensaje de blog.

Ejemplo 3.2.4.1 Mi blog.

```
<html>
<head><title>Creando un Blog</title>
1  <style type="text/css">
    body{ background-color:#7fffd4;}
        div#div1{ background: white;
        border-width:1px;
        margin-left:20%;
        margin-right:20%;
        }

        p { margin-left:1%;
        font-family: arial;}
    </style>
    <script type="text/javascript">
2      function BlogEntry(){ //Create a blog class
        var date; //Properties
        var message;
        var name;
        this.setDate=setDate; // Métodos
        this.setBlogger=setBlogger;
        this.setMessage=setMessage;
        this.showBlog=showBlog;

3      function setDate(){ // Obteniendo la fecha actual
        var d=new Date();
        var year=d.getFullYear();
        var month=d.getMonth() + 1;
        var day=d.getDate();
        date=month + "/" + day + "/" + year;
```

```
4      }
      function setBlogger(nameVal){
          name=nameVal;
      }
      function setMessage(messageVal){
          message=messageVal;
      }
5      function showBlog(){
        // Creando elementos HTML
6          var para = document.createElement("p");
          var count=document.getElementsByTagName("p").length;
        // Obteniendo número de párrafos
7          para.setAttribute("id","para" + count); // Identificación
        // alert("There are " + count+ " paragraphs so far.");
8          var aBR = document.createElement("br");
          var aBold = document.createElement("b");
9          var divObj = document.getElementById("div1");
        //Obteniendo referencias a los div
10         divObj.appendChild(para); // Anexar un párrafo
        // Creando nodos
11         var txt1=document.createTextNode("Publicado por: " + name + " el: ");
          var txt2=document.createTextNode(date);
          var txt3=document.createTextNode(message);
12         para.appendChild(txt1);
13         para.appendChild(txt2);
          aBold.appendChild(aBR);
          para.appendChild(aBold);
          aBold.appendChild(txt3);
          para.appendChild(aBold);
14         // alert(document.getElementById("para"+count).innerHTML);
        }
15     } // Fin de la clase
16     function addBlog(){ /* Añadir una entrada al blog a través de DOM y obtener los datos del
                           formulario */
17         message=document.getElementById("text").value;
          bloggername=document.getElementById("name").value;
18         var blog = new BlogEntry(); // Crear un objeto nuevo en el blog
          blog.setMessage(message); // Objetos de llamadas a métodos
          blog.setBlogger(bloggername);
          blog.setDate();
          blog.showBlog();
        }
        </script>
</head>
<body>
    <div align="center">
        
        </div>
        <h2 align="center">Bienvenido a mi Blog</h2>
19        <div id="div1">
            </div >
            <br />
            <div align="center">
                <form method="post">
                    Ingrese su nombre (optional) <br />
                    <input type="text" id="name" />
                    <br /><br />
20                    <textarea id="text" name="text" rows="5" cols="40" onfocus="this.value="">
                    </textarea><br />
21                    <input type="button" value="Agregar al blog" onclick="addBlog();" />
                </form>
            </div>
        </body>
</html>
```

Explicación:

1. Primero está la página de estilo que se utiliza en la página.
2. La función `BlogEntry()` define una clase que consiste en las propiedades y métodos para objeto nuevo que se crea dentro del blog.
3. Este método consigue el formato de la fecha actual con el constructor `Date()` de JavaScript. Se le asignará a la propiedad de fecha del objeto.
4. Los métodos `setBlogger(nameVal)` y `setMessage(messageVal)`, establecen las propiedades para el nombre del blogger y el mensaje que se escribe.
5. El método `showBlog()` es donde hacemos uso de DOM mediante la creación de elementos y los nodos de texto. Cada mensaje del blog se colocará en un párrafo añadido a un contenedor `<div>`. Se incluirá el nombre de blogger y la fecha en una línea con el texto en negrita en las siguientes líneas. Cada vez que se introduce un nuevo comentario, se adjunta un párrafo en la parte inferior.
6. Un nuevo párrafo elemento se crea y se le asigna una referencia a él.
7. Se establece un atributo de identificación para cada párrafo después de su creación, cada vez que se añade un nuevo párrafo `document.getElementsByTagName("p")` devuelve el número de elementos `<p>` en el documento, haciendo el recuerdo yendo uno por uno, este valor de cuenta se concatena con la cadena "para" producir una nueva identificación para cada nueva creación.
8. Una referencia en un elemento HTML se crea y asignado a `aBR`. En la línea siguiente, se crea una referencia a una etiqueta `<b>` de HTML.
9. El método `getElementById()` devuelve una referencia a un objeto identificado como `div1` en la línea `<div id="div1">`. Aquí es donde los párrafos se agregan y los mensajes aparecen en el blog.
10. Ahora el elemento `<p>` (párrafo) se añade al objeto `div`.
11. Tres nodos de texto se crean con los valores que se insertan en el párrafo. Las propiedades del objeto `blog`, nombre, fecha y el mensaje, contiene los valores de texto que serán asignados a los nodos de texto.
12. El texto del párrafo se añade ahora como un nodo secundario.
13. La siguiente línea de texto, se anexa seguido por el elemento `br` y el elemento `b`.
14. El `alert()`, mostrará el texto (`innerHTML`) para cada párrafo después de haber sido creado, por ejemplo: Publicado por Edgard Martínez el 21-04-11 `<B>` Este es un día brumoso y estoy en estado de ánimo confuso. `</b>`.
15. La clase `BlogEntry` finaliza con una llave de cierre.
16. La función `addBlog()` es una función definida por el usuario, no forma parte de la clase. Su función es crear un nuevo objeto del blog y asignarle propiedades.
17. Los datos del formulario (nombre del blogger y el mensaje de blog) se recuperan con el método `document.getElementById()`.
18. Se crea una instancia del nuevo objeto `blogentry` y una referencia asignada a la variable `blog`. Los métodos para el objeto están llamados a establecer las propiedades del objeto `blog`.
19. Después de configurar todas las imágenes, este es el contenedor `<div>` donde los mensajes del blog se almacenan y muestran.
20. El usuario escribe su blog en una caja de texto, después de escribir el mensaje, si el usuario sale del área de la caja de texto y hace clic en el cuadro, el controlador de evento `onfocus` se activará mediante el ajuste de su valor a una cadena vacía `<textarea id="text" name="text" rows="5" cols="40" onfocus="this.value="">`.
21. Cuando el usuario haga clic en el botón, el manejador de evento `onclick` llamará a la función `addBlog()` para crear un nuevo objeto del blog y establece sus propiedades con los datos recuperados del formulario `message=document.getElementById("text").value;`
`bloggername=document.getElementById("name").value;`

3.3 Tablas HTML en DOM

Las tablas son elementos muy comunes en las páginas HTML, por lo que DOM proporciona métodos específicos para trabajar con ellas. Si se utilizan los métodos tradicionales, crear una tabla es una tarea tediosa, por la gran cantidad de nodos de tipo elemento y de tipo texto que se deben crear.

Afortunadamente, la versión de DOM para HTML incluye varias propiedades y métodos para crear tablas, filas y columnas de forma sencilla.

Firefox no requiere un TBODY o un TFOOT. Internet Explorer requiere que usted cree un elemento TBODY e insertarlo en la tabla cuando se utiliza el DOM. Sin ella obtendrá una página en blanco. Debido a que está manipulando la estructura del documento directamente.

Ejemplo 3.3.1 Creando una tabla con DOM.

```
<html>
  <head><title>Creando una tabla con DOM</title><head>
    <style type="text/css">
1      table { width:380px;
        border: 2px solid darkgreen;
        background-color:#8fbc8f; }
        th { font-size: larger;
        font-family:arial}
        td { border: 1px solid #888878;
        padding: 5px;
        background-color:#fff8dc;
        font-family:arial;
        }
    </style>
    <script type="text/javascript">
2      window.onload = function(){
3        // Creando los elementos de la tabla
        var Table = document.createElement("table");
        Table.setAttribute("id","myTable"); // Creando id
        var THead = document.createElement("thead");
        var TBody = document.createElement("tbody");
        var Row, Cell;
        var i, j;
        // Declarando un arreglo de dos-dimensiones de los datos de la tabla
4      var header=new Array( "Estado","Capital","Abreviatura");
5      var state =[ ["Nicaragua","Managua","Nic"], ["California","Sacramento","CA"],
        ["Maine","Augusta","ME"], ["Montana","Helena","MT"], ["New York","Albany","NY"]];
        // Insertando los elementos creados dentro de la tabla.
6      Table.appendChild(THead);
        Table.appendChild(TBody);
        // Insertando una fila en el encabezado
7      Row = document.createElement("tr");
8      THead.appendChild(Row);
        // Creando e insertando celdas dentro de la fila del encabezado de la tabla
9      for (i=0; i<header.length; i++){
10         Cell = document.createElement("th");
11         Cell.innerHTML = header[i];
        Row.appendChild(Cell);
        }
        // Insertando filas y celdas en el cuerpo.
12      for (i=0; i<state.length; i++){
13         Row = document.createElement("tr");
        TBody.appendChild(Row);
14         for (j=0; j<state[i].length; j++){
15             Cell = document.createElement("td");
16             Cell.innerHTML = state[i][j];
            Row.appendChild(Cell);
        }
        }
        // Insertando la tabla en el árbol del documento
17      Tcontainer=document.getElementById("TableContainer");
        Tcontainer.appendChild(Table);
    }
  </script>
</head>
<body>
```

```
18      <div id="TableContainer"></div>
      </body>
</html>
```

Explicación:

1. La página de estilo se crea para la tabla que nosotros estamos creando, incluyendo la tabla misma, su cabecera y celdas.
2. Cuando se carga la página, la función `startUp()` es llamada.
3. La siguiente declaración crea la tabla, su cabecera y el cuerpo para el árbol de DOM.
4. El arreglo contiene la cabecera y los valores de la tabla.
5. El arreglo de dos dimensiones contiene los valores que serán insertados en las celdas de la tabla.
6. Ahora que el nodo tabla ha sido creado, insertaremos el nodo tabla dentro del árbol principal de DOM, esto es para que el buscador lo pueda mostrar.
7. La siguiente fila se crea para la cabecera en la tabla y retornará una referencia.
8. La fila se inserta en dentro del árbol DOM para la cabecera de la tabla.
9. El primer ciclo `for` estará iterando para cada celda que necesite crear la cabecera de la tabla.
10. Con la instrucción **`Cell = document.createElement("th");`** se crean las celdas para las cabeceras de los datos.
11. Para el arreglo de datos para la cabecera ("Estado", "Capital", "Abreviatura"), la propiedad `innerHTML` asignara los datos de las celdas cada vez por medio del bucle. Después los datos son añadidos a cada uno de las celdas.
12. El segundo ciclo `for` iterará hasta que todas las filas y celdas de la tabla sean llenados.
13. Las filas de la tabla se crean una por una y aparecen en el árbol de DOM.
14. El ciclo `for` anidado iterará para cada una de las celdas en cada fila para que podamos acceder a los datos en la matriz bidimensional.
15. Primero tenemos que crear una fila para cada celda de la tabla, esto lo logramos con la instrucción **`Cell = document.createElement("td");`**.
16. En cuanto se crea la celda de la tabla, los datos actuales del arreglo bidimensional, llamadas estados, deberán ser insertados en cada celda de la tabla y añadidos a la estructura de DOM. Para esto es la línea **`Cell.innerHTML = state[i][j];`**.
17. Una referencia al contenedor `<div>` de la línea **`<div id="TableContainer"></div>`** se retorna para el método `getElementId()`, y la tabla terminada entera es añadida al `div`.
18. La parte de la página HTML es muy pequeña. Consta de las etiquetas `<div>` `</div>`, esto mantiene la tabla construida con los métodos DOM y JavaScript. El resultado es la ilustración 3.3.1.

Estado	Capital	Abreviatura
Nicaragua	Managua	Nic
California	Sacramento	CA
Maine	Augusta	ME
Montana	Helena	MT
New York	Albany	NY

Ilustración 3.3.1 Creando una tabla con DOM.

3.4 Clonando un nodo

Para hacer una copia de un nodo podemos utilizar el método `cloneNode()`. El método `cloneNode()` devuelve un duplicado del nodo actual con todos sus atributos. No se convierte en parte del documento, es decir, no tiene un nodo primario, hasta que se añade a otro nodo. Este método toma sólo un argumento de tipo booleano, verdadero o falso, que no es obligatorio. Si el valor es falso, sólo el nodo y sus atributos son clonados, pero no los nodos secundarios (lo que significa que sus nodos de texto no se clonan). Si se establece un `true` (en profundidad) a continuación, en la

clonación se incluyen todos sus nodos secundarios. Advertencia; si clona un elemento y tiene un atributo Id, se le tiene que dar al nodo clonado su propia identificación, porque el Id debe ser único.

Ejemplo 3.4.1 Clonando Nodos.

```
<html>
<head><title>Clonando Nodos</title>
<style>
  p { font-family: cursive ;
      background-color: yellow;
      border-style:solid;
      border-width:2px;
      border-color:blue;
    }
</style>
<script type="text/javascript">
1   function addPara(mode){
2     var p1 = document.getElementById("para1");
3     var oldPara=p1.firstChild;
4     newPara=oldPara.cloneNode(mode);
5     p1.appendChild(newPara);
    }
</script>
<head>
<body>
6   <p id="para1">
      La noticia es fabulosa, tengo que gritar otra vez! Te has ganado la
      lotería!<br />
    </p>
7   <input type="button" onClick="addPara(true);" value="Hacer Clon" />
</body>
</html>
```

Explicación:

1. El propósito de la función addPara() es clonar el texto de un párrafo.
2. Obtenemos la referencia al párrafo que será clonado, con la instrucción:
var p1 = document.getElementById("para1");
3. firstChild contiene el contenido o el texto del párrafo.
4. Ahora podemos crear un nuevo párrafo, se crea una réplica exacta del párrafo antigua. Se trata de una copia recurrente, por lo que todos los nodos secundarios se pueden copiar. Esto se hace con la línea **newPara=oldPara.cloneNode(mode);**.
5. Para hacer que el clon de una parte del documento, sea anexada al primer párrafo, usamos **p1.appendChild(newPara);**.
6. Este es el párrafo que será clonado. Se da una identificación para hacer referencia a él en la función de la línea **<p id="para1">**.
7. Cuando se pulsa el botón, la función addPara() es llamada con el valor booleano true, que se utiliza para hacer una copia en profundidad del primer párrafo.

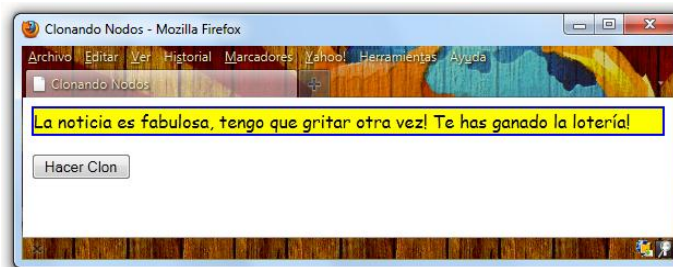


Ilustración 3.4.1 Antes que la clonación se lleve a cabo.

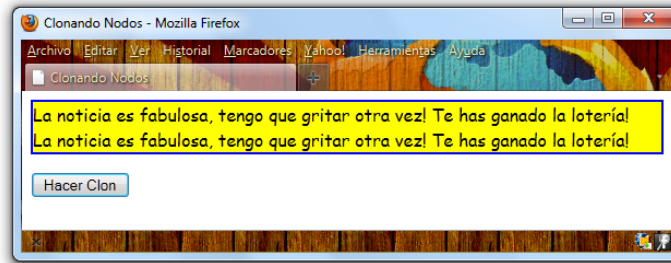


Ilustración 3.4.2 Después de haber pulsado el botón.

3.5 Clonando una tabla con un único Id.

En el siguiente ejemplo hacemos una copia de una tabla, la misma que creamos en el ejemplo 3.3.1. En lugar de copiar todo el código, en este ejemplo mostraremos las medidas necesarias para clonar la tabla y luego darle un identificador único.

Ejemplo 3.5.1 Clonando una tabla con Id único.

```
// Vamos a hacer la copia de la tabla.
// Después de la línea Tcontainer.appendChild(Table);, copiamos el siguiente código.
// De aquí en adelante es el nuevo ejemplo.
1      var tableCopy = document.getElementById("myTable");
2      var newTable = tableCopy.cloneNode(true);
3      newTable.id = "newtable_id";
4      newDiv = document.getElementById("NewTableContainer");
5      newDiv.appendChild(newTable);
        }
    </script>
</head>
<body>
    <div id="TableContainer"></div>
    <br />
6      <div id="NewTableContainer"></div>
</body>
</html>
```

Explicación:

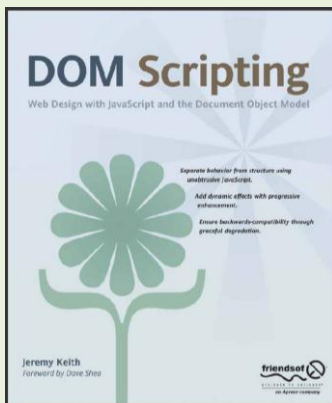
1. El Id de la tabla creada en el ejemplo 3.3.1, es myTable. El método `document.getElementById()` devuelve una referencia a esa tabla. Para eso sirve la línea **`var tableCopy = document.getElementById("myTable");`**.
2. La tabla original creada en el ejemplo 3.3.1, la clonamos con el método `cloneNode()`. Crea una réplica de la tabla original incluyendo sus atributos. **`var newTable = tableCopy.cloneNode(true);`**
3. Debido a que el clon incluye el atributo Id, un nuevo identificador es asignado a la tabla clonada. La regla es que los identificadores de HTML deben ser únicos o el programa podría identificar otro elemento con el mismo Id. **`newTable.id = "newtable_id";`**
4. Esta línea devuelve una referencia a un nuevo div.
`newDiv = document.getElementById("NewTableContainer");`
5. La tabla clonada se anexa al nuevo div, debajo de la tabla original (ver ilustración 3.5.1).
`newDiv.appendChild(newTable);`
6. Este es el contenedor div de la tabla clonada. **`<div id="NewTableContainer"></div>`**



Ilustración 3.5.1 Clonando una tabla con un único Id.

Bibliografía del Capítulo1

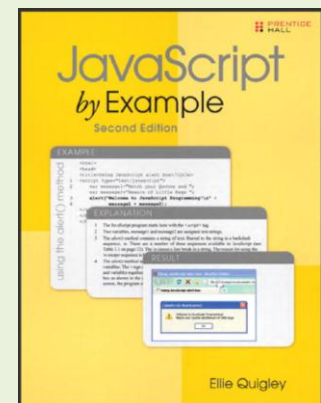
Dom Scripting
Jeremy Keith



Object-Oriented JavaScript
Stoyan Stefanov



JavaScript by Example
Ellie Quigley





Capítulo 4

PROGRAMACIÓN DEL LADO DEL SERVIDOR CON PHP



4 PROGRAMACIÓN DEL LADO DEL SERVIDOR CON PHP.

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas Web dinámicas. PHP es un lenguaje multiplataforma y dispone de gran cantidad de librerías que facilitan el desarrollo de las aplicaciones.

El código generado es mantenible y utiliza objetos, es un lenguaje basado en herramientas con licencia de software libre, es decir que no hay que pagar licencias y se puede ampliar incorporándole nuevas funciones si así lo deseamos.

PHP es un lenguaje de programación del lado del servidor, lo que significa que el código se interpreta del lado del servidor y no del lado del ordenador de los usuarios, el resultado producido es enviado al navegador del visitante en forma de HTML, imagen, documento, etc. Lo que permite que los clientes (usuarios) no accedan al código fuente de PHP sino solo a su resultado HTML.

Es necesario tener acceso a una instalación de trabajo de Apache, PHP y MySQL, de preferencia en el equipo local. Siempre es conveniente poner a prueba a nivel local, tanto para la velocidad como seguridad. Hacer esto en su equipo local disminuye la cantidad de tiempo dedicado a la carga de archivos a un servidor FTP y esperar que las páginas sean recargadas.

Además de los sitios web dinámicos depende de la información almacenada que se pueden modificar rápida y fácilmente, lo que es la principal diferencia entre un sitio dinámico y un sitio HTML estático. Sin embargo, PHP no proporciona una forma sencilla y eficiente para almacenar datos. Aquí es donde una base de datos relacional gestionada entra en juego.

A diferencia de HTML, que es analizado por el navegador cuando se carga una página, PHP es pre-procesado por la máquina que sirve el documento (ésta máquina se conoce como Servidor), todo código PHP que figura en el documento HTML es procesado por el servidor antes de que el documento se envíe al navegador del visitante. Debido a que PHP es un lenguaje del lado del servidor, ejecutar scripts PHP en el equipo local requiere instalar un servidor en el equipo local, por lo que para los ejemplos de este capítulo, será utilizado el servidor Apache corriendo en sus equipos locales.

4.1 Secuencias de comandos del lado del Servidor Versus lado del cliente.

Internet cuenta con dos tipos principales de secuencias de comandos: los del lado del servidor y del lado del cliente. Las secuencias de comandos del lado del cliente están compuestas principalmente de JavaScript, que es responsable de muchas de las características web que vemos como ventanas pop-up, algunas animaciones, y otras características de sitios web como menús desplegables. La razón de esto se llama secuencias de comandos del "cliente" porque el código se ejecuta en la máquina del usuario, después que la página se ha cargado. Usar scripts del lado del cliente te permite realizar cambios en una máquina sin necesidad de cargarla desde el servidor, también facilita la validación del formulario inicial y simplifica la toma de las interfaces para el usuario. Sin embargo, el uso de scripts del lado del cliente requiere que los usuarios tengan activado JavaScript o que su navegador sea compatible con el código que usted ha creado. Esto significa que usted no debe usar secuencias de comandos del lado del cliente para autenticación o el manejo de aspectos sensibles, debido a la capacidad del usuario para modificar o inhabilitar los scripts del lado del cliente.

Los scripts del lado del servidor se realizan en el servidor que aloja el sitio antes que la página se entregue al usuario. Esto significa que cualquier cambio que se haga por un guión, requiere una actualización de la página. Usar scripts del lado del servidor es ideal para la autenticación de usuarios, guardar los cambios en las bases de datos de información, la recuperación de las entradas para la exhibición, y muchas otras tareas. La desventaja de usar scripts del lado del Servidor es principalmente cuando se requiere actualizar la página. Debido a que la secuencia de comandos se procesa antes de que sea entregado al navegador, el usuario no tiene acceso a los mecanismos internos del código. Esto hace que las secuencias de comandos del lado del servidor sea la mejor opción para el manejo de información confidencial.

4.2 Sintaxis básica en PHP

Todas las secuencias de comandos en PHP se escriben en lo que se denominan bloques de código. Estos bloques se pueden incrustar en HTML, en general se definen mediante `<?php` al comienzo y `?>` al final. Todo lo que queda fuera de estos identificadores de bloques será ignorado por el compilador PHP y se devolverá directamente al servidor Web para que lo muestre en la pantalla del cliente.

Este es un pequeño ejemplo de la utilización de la sintaxis PHP, el segundo texto que aparece es generado únicamente con HTML.

```
<?php
    echo '<p>Este texto fue generado con PHP!</p>';
?>
<p>En cambio, este otro texto no.</p>
```

Esto es lo que usted debería poder visualizar en su navegador:

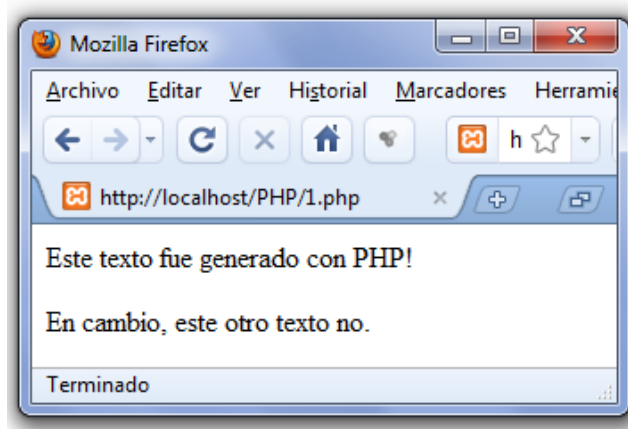


Ilustración 4.2.1 El segundo texto fue generado con HTML, el primero con PHP.

Como podemos ver, el texto dentro de los delimitadores de PHP fue manejado como un script, pero el texto manejado fuera fue representado como código HTML regular. No hay límites para la cantidad de bloques de código PHP que se puede utilizar en una página. Para ilustrar esto, mostramos el siguiente fragmento que es válido:

```
<?php
    echo '<p>Este es un texto simple.</p>';
?>
<p>Una parte de este texto es estático, <?php echo 'Pero este, seguramente no lo es!'; ?></p>
<?php echo '<p>'; ?>
    Este texto está encerrado en etiquetas de párrafo HTML que fue generado por PHP.
<?php echo '</p>'; ?>
```

El fragmento de código anterior, crea la siguiente salida en el navegador:

Este es un texto simple.

Una parte de este texto es estático, Pero este, seguramente no lo es!

Este texto está encerrado en etiquetas de párrafo HTML que fue generado por PHP.

Ilustración 4.2.2 Bloques de código PHP en una misma página.

4.2.1 Etiquetas de HTML como script

Por el bien de la compatibilidad con editores de software como páginas de Microsoft, PHP también soporta el uso de etiquetas `<script>` de HTML. La utilización de etiquetas `<script>` para PHP está un tanto descontinuado porque puede causar confusión con JavaScript. Desde PHP6, estas etiquetas no son soportadas.

```
<script language="php">
    echo 'Este es otro ejemplo';
</script>
```

4.3 Tipos de datos y variables.

El primer tipo de datos que presentaremos es el entero, es el tipo de dato fundamental en PHP y representa valores de números enteros con signos que rebasan los dos mil millones. PHP acepta valores numéricos enteros que utilizan tres bases matemáticas como son el **decimal**, **octal** y **hexadecimal**.

En PHP las variables pueden definirse asignándoles un valor o utilizando la sentencia VAR. Las variables inician siempre con el signo \$ seguido por cualquier combinación de caracteres, siempre que el primer carácter tras el signo \$ sea una letra o subrayado válido.

PHP permite almacenar casi cualquier cosa en una variable, usando uno de los siguientes tipos de datos:

String: Caracteres alfanuméricos, tales como sentencias y nombres.

Integer: Un valor numérico, expresado completamente en números.

Float: Un valor numérico, expresado en números reales o decimales.

Boolean: Evaluaciones verdaderas o falsas (evaluaciones a 1 para verdadero o a 0 (cero) para falso).

Array: Una colección indexada de datos.

Object: Una colección de datos y métodos.

4.3.1 String, sintaxis con comillas simples:

Un string es una serie de caracteres encerrados entre comillas simples (') o dobles ("). Encerrando una cadena en comillas simples, es la manera más simple de crear una cadena en PHP. No se extienden caracteres especiales o variables, pero en cambio estas cadenas son entregadas al navegador como texto simple.

Para ilustrar esto, veamos algunos ejemplos empleando algunos datos diferentes:

```
<?php
// El <br /> agrega una línea en blanco en el buscador para mejor legibilidad
echo 'Esta es una cadena. <br />';
echo 'Esta también es una cadena
Línea de espacio en blanco. <br />';
// Caracteres especiales, tales como un salto de línea con el carácter(\n) ,
// No se extenderá cuando encuentre comillas simples.
echo 'Esta es una cadena \n Con un carácter de salto de línea. <br />';
echo 'Esta es una cadena\'s seguido con un apóstrofe. <br />';
// Un espacio en blanco no necesita ser escapado sino escapando un
// carácter especial.
echo 'Esta es una cadena con una pleca (\) en ella. <br />';
echo 'Esta es una cadena con doble pleca (\\) en ella. <br />';
// Las variables no se excederán en las comillas simples
echo 'Esta $variable no se extenderá. <br />';
?>
```

La salida de este código será así en un navegador:

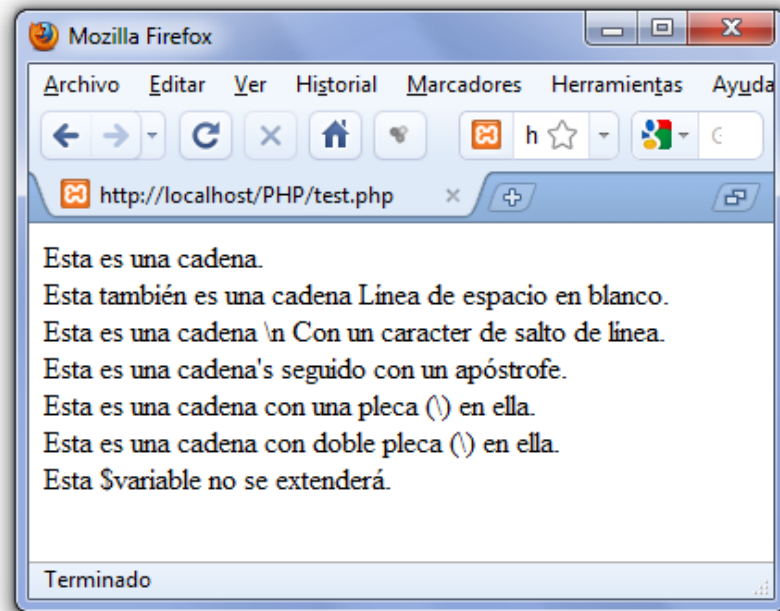


Ilustración 4.3.1.1 El carácter Newline (\n) no aparece en su navegador. Sin embargo, ellos están visibles en el código fuente en la vista de su página, que usted puede ver escogiendo la fuente de vista del menú de su navegador.

4.3.2 String, sintaxis con comillas dobles:

Las cadenas en cascada con doble comillas se comportan semejantes a cadenas en cascada con una sola comilla, la diferencia es que entre comillas dobles se pueden utilizar caracteres especiales incluyendo variables de expansión. Los caracteres especiales como el nuevo carácter de salto de línea (\n) no afectarán el rendimiento del navegador, pero afecta el orden de la línea, y despliega el código fuente. Use la etiqueta (
) para crear una nueva línea en el navegador.

Podremos ver las diferencias de usar comillas dobles o simples en el siguiente código:

```
<?php
    echo "Esta es una cadena. <br />";
    echo "Esta es una cadena
    con espacios
    con múltiples líneas. <br />";
    // Los apostrofes no necesitan ser escapados cuando se usan comillas dobles
    echo "Esta cadena's tiene un apóstrofe. <br />";
    // Las comillas dobles necesitan ser escapadas
    echo "Esta cadena dice, \"Yo necesito escapar!\" <br />";
    // El carácter nueva línea será interpretado
    echo "Estas cadenas tienen \n caracteres \n de nueva línea. <br />";
    // Un espacio en blanco no será impreso si un carácter especial
    // directamente seguido
    echo "Esta cadena contiene un backslash (\). <br />";
    // Las variables serán expandidas si no son escapadas
    $variable = "Palabra";
    echo "Esta cadena usa una $variable. <br />";
    // Una variable puede ser interpretada como texto plano para escapar el
    // signo dólar con un backslash
    echo "Esta cadena escapa la \$variable. <br />";
?>
```

En su navegador, el código anterior debería ser parecido a este:

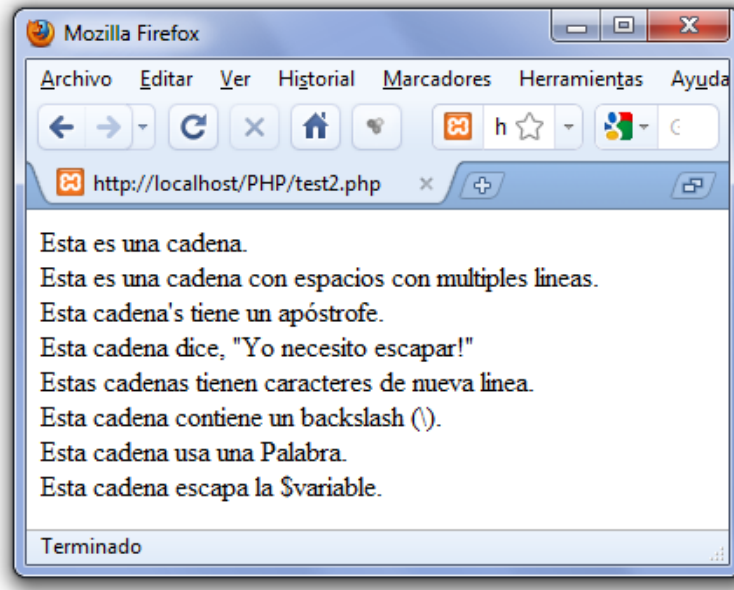


Ilustración 4.3.2.1 Utilización de comillas dobles.

4.3.3 Concatenación de cadenas

Puesto que es necesario en algún momento concatenar cadenas, usted puede lograrlo utilizando el operador de concatenación (.). Para que pueda unir dos cadenas debe poner entre ellas un punto. También se pueden concatenar variables con tal de que ellas no sean de tipo objeto.

En este ejemplo, estamos mostrando la concatenación de cadenas y variables:

```
<?php
    $foo = "Esta es una cadena " . " y esta es otra.";
    echo $foo;
?>

<br>
<?php
    $foo = "Si la patria es pequeña, ";
    $bar = "uno grande la sueña.";
    echo $foo . $bar;
?>
```

Este código producirá en su navegador, una salida como la que se muestra a continuación.

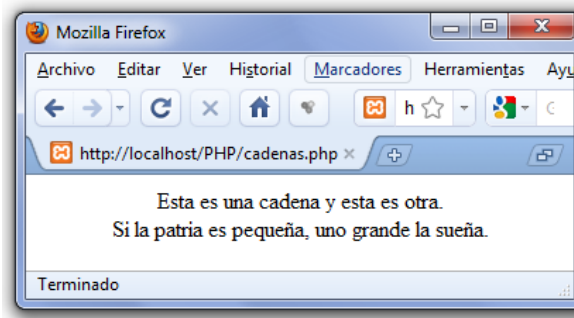


Ilustración 4.3.3.1 Concatenación de cadenas y variables.

4.3.4 Integer (enteros)

Un entero es cualquier número entero positivo o negativo (un número sin un valor decimal). Por ejemplo, los números 1, -27 pero 1.2, 6.3 no lo son.

PHP es un lenguaje flojamente recreado, no es necesario declarar una variable como un entero; sin embargo, si usted lo encuentra necesario, usted puede hacerlo explícitamente, o puede forzar un valor como un entero si utiliza la siguiente sintaxis:

```
$foo = 27; // No usa comillas un número entero siempre significa un entero.  
$bar = (int) "3-peat"; // Evalúa a 3  
$baz = (int) "seven"; // Evalúa a 0  
$bat = (int) "ten 4"; // Evalúa 0
```

4.3.5 Números de punto flotante

Los números con punto flotante (también conocidos como floats o doubles) son los números con valores decimales, o reales. Estos incluyen números como 3.14, 5.33333, y 1.1.

Note que los números de punto flotantes pueden producir resultados inesperados debido al hecho que es imposible representar todos los valores con un número finito de dígitos. Un ejemplo bueno para ilustrar esto es 1/3, que evalúa a un decimal repetidor (0.3333...). Por esta razón, Usted no debe acostumbrar comparar números decimales con igualdades.

4.3.6 Valores Boolean

Un valor Booleano es el tipo más simple de datos; representan la verdad, y pueden contener únicamente uno de dos valores: **Verdadero** o **Falso**. Es importante que note que los valores FALSO (no van entre comillas), los valores Boolean son diferentes de la cadena "FALSO", y lo mismo ocurre para el verdadero. Los valores Booleanos son muy útiles para determinar si una condición existe. Por ejemplo, usted puede usar el condicional if-else si desea realizar las acciones diferentes cuando una condición sea verdadera.

```
if($condición==verdadero)  
{  
    echo 'La condición es verdadera!';  
}  
else  
{  
    echo ' La condición es falsa!';  
}
```

4.3.7 Arreglos o Matrices

En PHP al contrario que otros lenguajes de programación, las matrices son una forma de agrupar cualquier cantidad de variables distintas, independientemente de su tipo en una única variable, técnicamente las matrices representan un mapa ordenado que asigna valores claves a elementos de datos variables.

Los arreglos están entre los tipos de datos más poderosos disponibles en PHP, debido a su habilidad de trazar la información usando llaves para valorar el apareamiento. Esto significa que un arreglo puede almacenar pedazos múltiples de información en una sola variable, todos los cuales estarán indexados por una llave. Por ejemplo; si usted tiene una entrada para almacenar en variables, usted necesitaría hacer lo siguiente si no uso arreglos:

```
<?php  
$entryTitle = "Ejemplo de Título";  
$entryDate = "Abril 13, 2011";
```



```
$entryAuthor = "Ángel";  
$entryBody = "El día de hoy, yo escribí una entrada para el blog."  
?>
```

Esto podría volverse confuso, sobre todo si la entrada necesita ser pasada a través de una función para ser procesada. Usted puede implementar arreglos y simplificar la entrada, de la siguiente manera:

```
<?php  
$entry = array(  
    'title' => 'Ejemplo de Título',  
    'date' => 'Abril 13, 2011',  
    'author' => 'Ángel',  
    'body' => 'El día de hoy, yo escribí una entrada para el blog.'  
);  
?>
```

El poder de este acercamiento consiste en que usted tiene toda esa información ahora guardada en una sola variable, \$entry. Para acceder a cualquier parte de la información, usted necesita agregar la llave al final de la variable.

```
<?php  
echo $entry['título']; // Salida "Ejemplo de Título"  
echo $entry['fecha']; // Salida "Abril 13, 2011"  
echo $entry['autor']; // Salida "Ángel"  
echo $entry['cuerpo']; // Salida "El día de hoy, yo escribí una entrada para el blog."  
?>
```

Los arreglos también pueden indexar información automáticamente usando un índice numérico que inicia en cero (0). Usted puede acceder a los valores del arreglo con el valor indexado que se ha puesto automáticamente usando el índice numérico, sin comillas. Usted puede crear automáticamente un índice numérico para un arreglo omitiendo la llave siempre que usted declare el arreglo.

```
<?php  
$entry = array('Ejemplo de Título', 'Abril 13, 2011', 'Ángel',  
    'El día de hoy, yo escribí una entrada para el blog.');
```

```
echo $entry[0], ' por ', $entry[2];  
?>
```

El código anterior produce una salida en el navegador parecida a ésta:

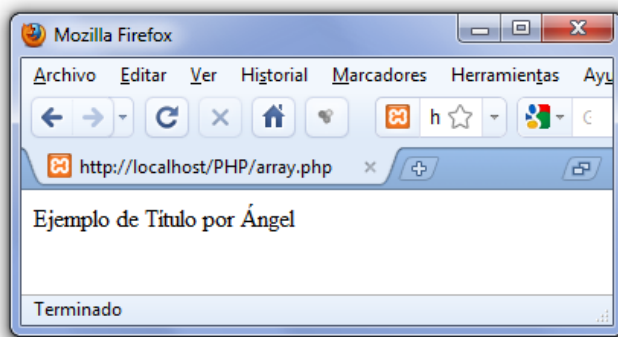


Ilustración 4.3.7.1 Utilización de arreglos.

Una nota importante que debemos tener en cuenta, es que programando los contadores inician en cero (0). Por lo que el primer carácter en una cadena se encuentra en la posición cero (0), no en la posición 1 como usted podría esperar.

Al usar arreglos de cadenas, usted debe tomar un paso adicional para evitar los errores. Para evitar un error, usted debe encerrar el arreglo y su código entre llaves ({}). Esto es conocido como sintaxis compleja, pero no porque sea

complicada para usar; más bien, se llama complejo porque le permite a PHP analizar las declaraciones complejas dentro de una cadena entre comillas.

```
<?php
$persona = array('nombre' => 'Ángel', 'edad' => 25);
echo "El nombre de esta persona es {$persona['nombre']}
y su edad es {$persona['edad']}.";
?>
```

Cuando se trabaja con arreglos multidimensionales, deben usarse corchetes. Dejando las comillas simples fuera de los índices, los arreglos se comportarán inesperadamente.

4.3.8 Arreglos multidimensionales

Otro rasgo de los arreglos, es que permiten anidar dentro de ellos. Esto permite crear un arreglo dentro de otro o crear arreglos multidimensionales.

Los arreglos multidimensionales son excepcionalmente poderosos, sirven para almacenar aún más información que la almacenada en una variable, haciendo los datos inmensos portables, como verá cuando usted inicia a trabajar con bases de datos.

Un arreglo multidimensional de personas podría parecerse a esto:

```
<?php
$persona = array(
    array('nombre' => 'Ángel', 'edad' => 25), // $persona[0]
    array('nombre' => 'Edilberto', 'edad' => 23) // $persona[1]
);
echo "{$persona[0]['nombre']} tiene un hermano de
{$persona[1]['edad']} años de edad.";
?>
```

Este código, producirá una salida como la siguiente:

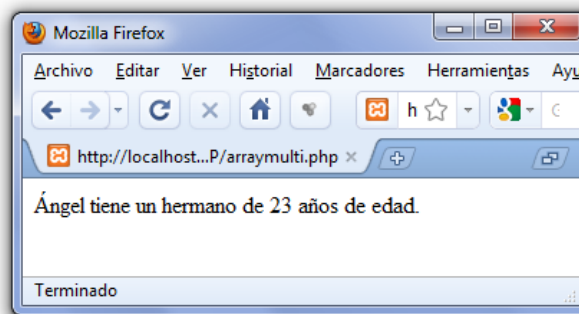


Ilustración 4.3.8.1 Arreglo multidimensional.

Este es otro arreglo multidimensional con cadenas.

```
<?php
$colores = array(
    'frutas' => array('manzanas' => 'roja', 'ciruela' => 'morada'),
    'flores' => array('rosas' => 'roja', 'violetas' => 'azul')
);
// Salida: Una manzana es roja, y una violeta es azul.
echo "Una manzana es {$colores['frutas']['manzanas']}, y una
violeta es {$colores['flores']['violetas']}.";
?>
```

Esta será el resultado en el navegador:

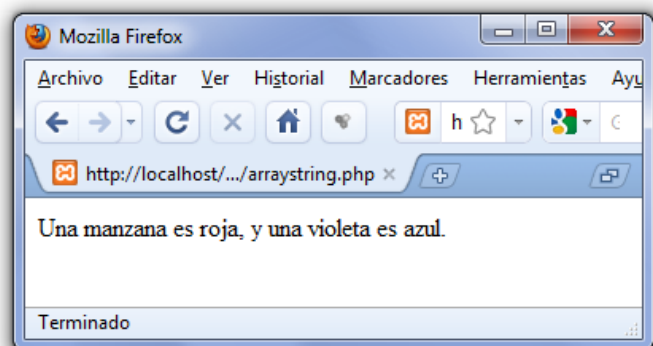


Ilustración 4.3.8.2 Salida de arreglos multidimensionales con cadenas.

4.4 Enviando datos a la salida

Para ver los datos que usted está procesando, necesita una manera de enviarlos a la salida. Esto le permite que despliegue los volúmenes de información de una variable en el navegador de sus usuarios.

Hay varias maneras de enviar una salida, pero el método más común es utilizando los comandos `echo()`, `printf()`, y `sprintf()`. Existen algunas diferencias con respecto al uso de cada uno de ellos, pero el resultado es el mismo en la salida que despliega el navegador.

Es importante que tenga un entendimiento sólido de las opciones con las que puede enviar las salidas al navegador, veremos algunas sentencias que nos permiten hacer esto, y hay propiedades especiales que se asocian a cada uno.

Para la introducción a esta parte del capítulo, iniciando por el prototipo, nombre de la función, argumentos aceptados, y un valor de retorno definido por su tipo de dato. No se preocupe si no está seguro completamente de lo que el código significa, cubriremos toda la información en el transcurso del capítulo.

4.4.1 Sentencia `print()`

La sentencia `print()` es el método más sincero para generar una salida. Su prototipo es así:

`int print (string $arg)`

Esto significa que `print()` acepta un argumento, el cual es una cadena que será la salida del navegador, y retorna un valor entero-`print()`, siempre retornará 1. Este es un ejemplo de la manera en que debe usted implementar este comando.

```
<?php
print("Esto solo es un fragmento.");
?>
```

4.4.2 Sentencia `echo()`

El método más común de generación de salida es probablemente la sentencia `echo()`. Este se diferencia ligeramente del `print()` en que éste puede aceptar múltiples argumentos. Consideremos el siguiente prototipo:

`Void echo (string $arg1 [, string $... ] )`

La sentencia `echo()` acepta uno o más argumentos, separados por comas, y todos los argumentos de salida se muestran en sucesión en el navegador. A diferencia de `print()`, `echo()` no retorna un valor, la palabra reservada `void` nos dice esto. Porque `echo()` también es un lenguaje de construcción, los paréntesis son opcionales y generalmente omitidos. Probemos el siguiente código en un archivo `echo.php`.

```
<?php echo "Hola a ", "todo el mundo!"; ?>
```

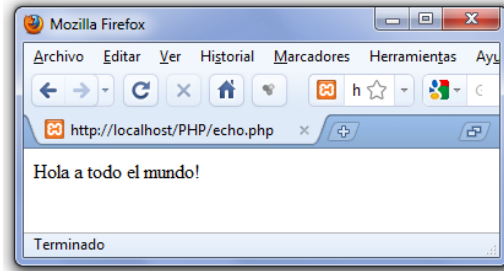


Ilustración 4.4.2.1 Implementación del método `echo`.

Las dos cadenas son agregadas juntas como los argumentos de la sentencia `echo()`, produciendo una cadena final que se pasa al navegador. Lo mismo ocurre cuando trabajas con variables.

```
<?php
    $foo = "Hola a ";
    $bar = "todo el mundo!";
    echo $foo, $bar;
?>
```

La salida de este código, es la misma que se muestra en la ilustración 4.4.2.1

4.4.3 Sentencia `printf()`

La sentencia `printf()`, nos da un poco más de control sobre la salida, permitiéndonos definir el formato de los datos que serán enviados al navegador. Usted podría pensar que esta sentencia es igual a `print()`. Esta última es especial para utilizarla cuando esté tratando con números, primero hechemos un vistazo al prototipo de `printf()`:

```
int printf (string $format [, mixed $args [, mixed $... ] ] )
```

Cuando una función acepta un tipo de dato `mixed`, es porque la función puede aceptar varios tipos de datos, generalmente, todos los tipos de datos excepto los arreglos y los objetos. También, argumentos entre comillas son opcionales en el prototipo de la función.

En este punto, podemos darle formato a una cadena para `printf()`, a lo largo de otros argumentos que encajaran dentro del formato. Esta es una buena oportunidad para verificar que los tipos de datos que están pasando son del prototipo deseado. Veamos el siguiente código:

```
<?php
    printf("PHP es %s!", "importante");
?>
```

La salida del código anterior es este:

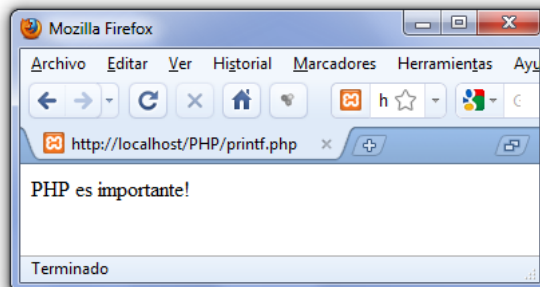


Ilustración 4.4.3.1 Salida empleando la sentencia printf().

En el procesamiento del código, usted puede crear un formato para la cadena ("PHP es %s!") con una conversión específica, la cual inicia con el signo porcentual (%) y a continuación la serie de especificación. En este ejemplo, usted puede asignar un tipo específico a la cadena, con decirle a la función que tipo de argumento debe esperar para el dato.

El uso más práctico de printf() es con números decimales, tales como valores de dólar. Consideremos el siguiente código utilizando echo().

```
<?php
$amt1 = 2.55;
$amt2 = 3.55;
$total = $amt1 + $amt2;
echo 'El costo total es $', $total;
?>
```

Esto es lo que obtenemos cuando corremos el código:

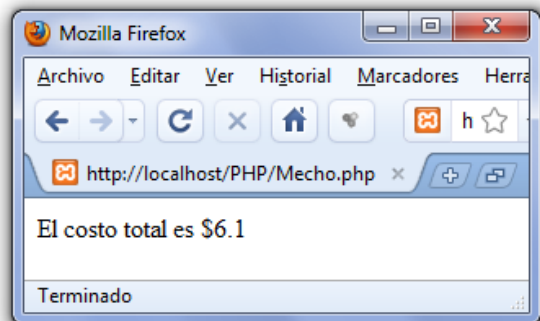


Ilustración 4.4.3.2 Utilizando echo().

En realidad lo que deberíamos obtener es ésta salida:

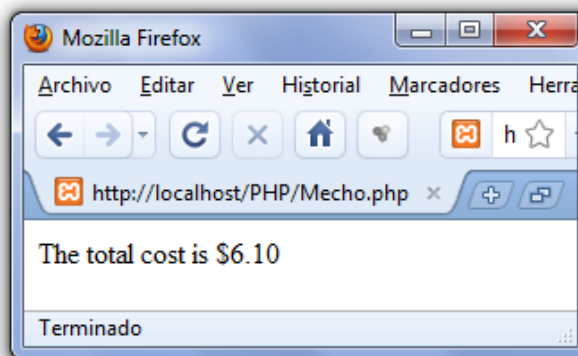


Ilustración 4.4.3.3 Utilizando printf().

La salida debió haber sido 6.10 como lo muestra la ilustración 4.4.3.3, no 6.1 como está en la ilustración 4.4.3.2. Esto por razones obvias no es lo que usted quiere si está tratando de desplegar un precio. Afortunadamente, éste inconveniente lo resolvemos con printf(). Agregando la siguiente línea de código al archivo que estamos trabajando:

```
printf('El costo total es $%.2f', $total);  
en lugar de:  
echo 'El costo total es $', $total;
```

Guardemos el archivo nuevamente y ahora corrámoslo otras ves. Esta vez el navegador nos mostrará el resultado esperado. Véase Ilustración 4.4.3.3

La razón por la que podemos formatear la salida de números en la línea final es porque les estamos especificando que después del punto decimal del número nos muestre dos posiciones, y eso es lo que le decimos a printf() al momento de retornar el valor usando (%.2f).

4.4.4 Sentencia sprintf()

La sentencia sprintf() trabaja de manera parecida a printf(); usted puede darle formato con tipos específicos y un grupo de argumentos que serán insertados dentro de la cadena. La única diferencia es que mientras printf() muestra la salida directamente en el navegador, sprintf() retorna la cadena. Este es su prototipo:

```
String sprintf ( string $format [, mixed $args [, mixed $... ] ] )
```

La única diferencia que puede verse entre las sentencias printf() y sprintf; es la “cadena” procedente de las sentencias. La palabra que precede a una función te dice que tipo de valor devolverá la función. El beneficio de usar sprintf() en nuestros scripts estamos habilitando el formato seleccionado únicamente en la sección de datos, que nos salva de tener que reestructurar la salida entera.

4.5 Operadores

PHP, al igual que otros lenguajes de programación, están provistos de un número de operadores para que usted pueda manipular los datos. Estos operadores están dentro de varias categorías; ahora hablaremos de cómo sacarles provecho a algunos de los siguientes operadores:

7. Operadores aritméticos: El entorno de las funciones matemáticas básicas.
8. Operadores de Asignación: Estos ponen valores a las expresiones.
9. Operadores de Comparación: Estos determinan la similitud entre dos valores.
10. Operadores de control de errores: Estos operadores especiales son para suprimir errores.
11. Operadores de Incremento / decremento: Estos incrementan o decrementan un valor.
12. Operadores lógicos: Estos denotan operaciones lógicas.
13. Operadores de cadenas: Estos manipulan cadenas.

4.5.1 Operadores aritméticos.

Los operadores aritméticos en PHP son solamente funciones que uno usa desde la escuela:

El operador de adición (+) retorna la suma de dos valores.
echo 2 + 2; Salida 4.

El operador de sustracción (-) retorna la diferencia entre dos valores.
echo 4 - 2; salida 2.

El operador multiplicación (*) retorna el producto de dos valores.

echo 4 * 2; salida 8.

El operador de división (/) retorna el cociente de dos valores

echo 8 / 2; salida 4.

El operador de la división (/) retorna un valor decimal si los dos operandos no son enteros y los números serán divisibles, en tales casos el retorno será un numero entero.

El operador modulo (%) retorna el resto de un valor dividido entre otro.

echo 7 % 2; salida 1.

Echo 8 % 2; salida 0.

4.5.2 Operadores de asignación aritmética

PHP proporciona varios operadores de asignación disponibles para que sean empleados con valores de una operación con los valores de una expresión. Esto lo hacemos con el signo igual (=), pero es importante estar entendidos que hay signos que no son iguales, como normalmente se entiende. Veamos los siguientes códigos:

\$a = 5, al leer esta expresión entendemos que “El valor de \$a será 5”.

También hay algunas combinaciones de operadores que nos muestran como declarar y asignar un valor a un operando en un paso sencillo y rápido. Estos operadores combinan operadores aritméticos tales como;

```
<?php
$foo = 2;
$foo += 2; // El nuevo valor es 4
$foo -= 1; // El nuevo valor es 3
$foo *= 4; // El nuevo valor es 12
$foo /= 2; // El nuevo valor es 6
$foo %= 4; // El nuevo valor es 2
?>
```

4.5.3 Operadores de comparación

Usted puede usar operadores de comparación para determinar la similitud entre dos valores. Estos se utilizan especialmente para controlar estructuras. Los operadores de comparación disponibles le permiten determinar si las condiciones siguientes están presentes entre dos valores:

14. (!= or <>): Valores que no son iguales.
15. (!==): Valores que no son idénticos.
16. (<): Valor 1 es menor que el valor 2.
17. (>): Valor 1 es mayor que el valor 2.
18. (<=): Valor 1 es menor o igual al valor 2.
19. (>=): Valor 1 es mayor o igual que el valor 2.

Nota: Igual e idéntico no son la misma cosa. Idénticos es cuando ambos, variables, valores y tipos los son. Normalmente se verifican los valores Booleanos con el operador de la comparación idéntico porque FALSO==0 evalúa para arreglar, mientras FALSO===0 evalúa a falso.

4.5.4 Operadores para controlar errores.

PHP ofrece un operador para controlar errores: el símbolo (@), temporalmente reporta los errores en los niveles de nuestros script; esto impide que los errores se desplieguen si ocurren. Por ejemplo, trate de referenciar un archivo inexistente con `include_once`, esto podría causar un error a lo largo de la línea.

Warning: include_once(fake_file) [function.include-once]: fallo la apertura stream: No existe tal archivo o directorio en /Applications/xampp/xamppfiles/htdocs/simple_blog/test.php en la línea 4

Warning: include_once() [function.include]: Fallo la apertura de 'fake_file' por la inclusion (include_path='.:Applications/xampp/xamppfiles/lib/php') en /Applications/xampp/xamppfiles/htdocs/simple_blog/test.php en la línea 4.

Este es un error bastante delicado, y usted probablemente no quiere que sus usuarios vean algo de esto desplegado en sus pantallas.

```
<?php
    @include_once 'fake_file';
    echo 'Texto para seguir el include.';
?>
```

Poner un símbolo de operador antes de la variable se llama *prepending*. Esta técnica le permite que realice una operación en una variable antes que sea instanciada. Poniendo un operador después de la variable se le llama *postpending*, la técnica instantiates en una variable primero, y entonces ejecuta una operación en ella. El archivo no existe, y el error es generado, pero el símbolo previene del error que se desplegará.

4.5.5 Operadores de incremento / decremento

En algunos scripts, ocasionalmente es necesario agregar o substraer rápidamente un valor. Para agregar uno a un valor, agregamos dos signos (++) antes o después de la variable. Para substraer uno, agregamos dos signos (--).

```
<?php
    $foo = 5;
    ++$foo; // El nuevo valor es 6
    $foo++; // El nuevo valor es 7
    --$foo; // El nuevo valor es 6
    $foo--; // El nuevo valor es 5
    $bar = 4;
    // Echo
    echo ++$bar; // La salida es 5, el nuevo valor es 5
    // Echo
    echo $bar++; // La salida es 5, el nuevo valor es 6
?>
```

4.5.6 Operadores lógicos

Es difícil de cubrir los operadores lógicos en PHP, sin usar el método estructura para ilustrar como trabajan ellos, ahora veremos cómo utilizarlos. Los operadores lógicos le permiten determinar si dos condiciones son verdaderos o no. Esto es útil al usar declaraciones condicionales para dictar lo que pasa en un programa. PHP incluye los siguientes operadores:

AND or &&: Retoran verdadero si ambas expresiones son verdaderas.

OR or ||: Retorna verdadero si al menos una de las expresiones es verdaderas.

XOR: Retorna verdadero si una expresión es verdadera, pero no la otra.

!: Retorna verdadero si la expresión no es verdadera.

Veamos una demostración parcial de cómo trabajan estos operadores:

```
<?php
    $foo = Verdadero;
    $bar = falso;
    // Imprime la sentencia si $foo y $bar son verdaderos
    if($foo && $bar) {
        echo 'Ambos $foo y $bar son verdaderos. <br />';
    }
    // Imprime la sentencia si $foo o $bar es verdadero
    if($foo || $bar) {
        echo 'Por lo menos una de las variables es verdadera. <br />';
    }
    // Imprime la sentencia si $foo o $bar son verdaderos, pero no ambos
    if($foo xor $bar) {
        echo 'Una variable es verdadera, y una es falsa. <br />';
    }
    // Imprime la sentencia si $bar no es verdadera
    if(!$bar) {
        echo '$bar es falsa. <br />';
    }
?>
```

Este código produce la siguiente salida:

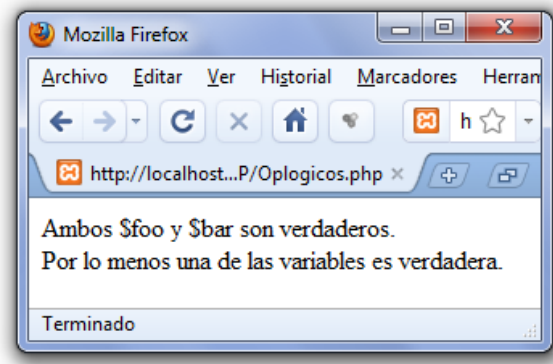


Ilustración 4.5.6.1 Operadores lógicos.

4.5.7 Operadores de cadenas

Hay dos operadores habilitados para cadenas en PHP: el operador de concatenación (.) y el operador de asignación (.=). Los operadores de concatenación combinan dos cadenas dentro de una, uniendo el extremo de la cadena a la izquierda del operador al principio de la cadena a la derecha del operador.

El operador de la asignación encadenado agrega una cadena al extremo de una variable existente:

```
<?php
    $foo = "Hola";
    $bar = $foo . " mundo! <br />";
    echo $bar; // Salida: Hola mundo!
    $bar .= " y de nuevo!";
    echo $bar; // Salida: Hola mundo! y de nuevo!
?>
```

4.6 Estructura de control

Para agregar poder y conveniencia a sus scripts, PHP soporta un número de sentencias condicionales, bucles, y otras estructuras de control, esto nos permite manipular los datos a lo largo de nuestro código.

20. if	30. return
21. Else	31. require
22. elseif/else if	32. include
23. while	33. require_once
24. do-while	34. include_once
25. for	35. goto
26. foreach	
27. break	
28. continue	
29. switch	
30. return	

Puesto que una parte de estas sentencias condicionales ya se han estudiado en el capítulo de JavaScript, únicamente tomaremos las sentencias que no se han estudiado en esta monografía.

4.6.1 Continue

La sentencia continue trabaja similar al break, con una diferencia esencial; termina solamente con la iteración actual. Después de la sentencia continue, se vuelve a iniciar con la evaluación de la condición. Esto es útil en casos donde usted quiere solo realizar las acciones en los datos en la vuelta en que se encuentra obedeciendo cierto criterio, de tal forma que se producen solo esos valores:

```
<?php
for($i=0; $i<=10; ++$i) {
    /*
     * Si el módulo de $i y 2 no es cero (que evalúa
     * a falso), nosotros continuamos
     */
    if($i%2) {
        continue;
    }
    echo $i, " ";
}
?>
```

Al correr la aplicación obtenemos la siguiente salida:

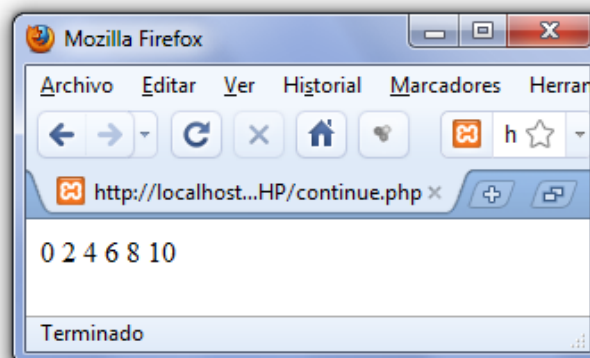


Ilustración 4.6.1.1 Implementación de la sentencia continue.

4.6.2 Include, include_once, require, y require_once

Un gran rasgo proporcionado por PHP es la habilidad de cargar una escritura de un archivo externo; esto hace mucho más fácil de organizar su código en los proyectos más grandes. PHP proporciona cuatro estructuras que usted puede utilizar para cargar un script externo include, include_once, require, y require_once.

El manual de PHP recomienda que los diseñadores usen include_once y require_once porque éstas estructuras verifican si el archivo ya ha estado cargado antes de cualquier carga, esto ahorra los recursos y puede aumentar la ejecución de sus aplicaciones.

Ahora echaremos un vistazo a un ejercicio que ilustra el poder de cargar scripts externos. Crearemos para efecto de ilustrar este ejemplo, dos archivos, uno se llamara extras.php y el otro se llamara test.php:

En el archivo extras.php agregue el siguiente código:

```
<?php
    $foo = "verde";
    $bar = "rojo";
?>
```

Ahora al archivo test.php agregaremos el siguiente código:

```
<?php
    include_once 'extras.php';
    echo 'Variable $foo tiene un valor de ', $foo, "<br />\n";
    echo 'Variable $bar tiene un valor de ', $bar, "<br />\n";
?>
```

Guarde los cambios en ambos archivos y ahora acceda desde el navegador a la página test.php, verá el siguiente resultado:

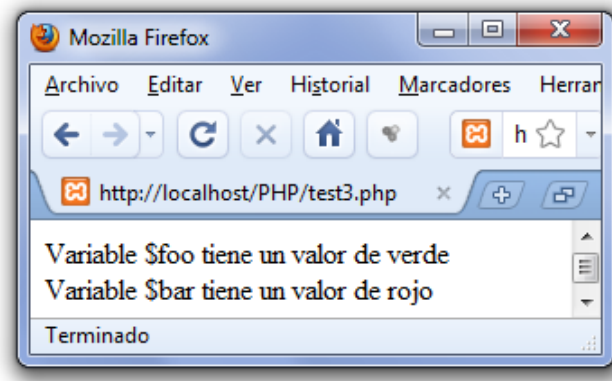


Ilustración 4.6.2.1 Salida del archivo test.php, utilizando la inclusión del archivo extras.php.

Para incluir el archivo extras.php se uso include_once, usted puede habilitar el acceso a información almacenada en un archivo. Esto provee un uso especial cuando usted está incluyendo archivos que están en otras áreas de su sitio, en lugar de exigir copiar y pegar esas funciones en cada archivo. La adopción de este enfoque reduce el tamaño de sus aplicaciones y pueden desempeñar un papel en la optimización de la aplicación.

Este pequeño ejemplo siguiente, ilustra cómo usar include_once puede reducir la carga en el servidor; empecemos agregando este código al archivo extras.php:

```
<?php
    $var += 1;
?>
```

Lo que sigue es agregar este código al archivo test.php:

```
<?php
    $var = 0;
    include 'extras.php';
    echo $var, "<br />";
    include 'extras.php';
    echo $var, "<br />";
?>
```

Este código genera el siguiente resultado cuando se carga en un navegador:

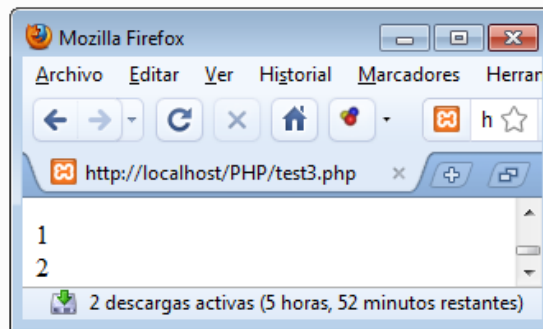


Ilustración 4.6.2.2 Uso de la sentencia include.

El archivo es cargado solo una vez, el script se ejecuta sólo una vez. Esto reduce la carga en el servidor, que a su vez reduce el tiempo de ejecución de los scripts.

4.6.3 Goto

PHP5.3.0 introdujo la instrucción goto, que le permite saltar a una nueva sección de código:

```
<?php
    if ($i==7) {
        goto foo;
    }
    echo "Esto será un salto si \$i es igual a 7.";
foo:
    echo "Esto se debe imprimir.";
?>
```

Goto es una adición controvertida al lenguaje, sobre todo porque muchos desarrolladores consideran que han tenido efectos negativos sobre la legibilidad del código. También tenga en cuenta que XAMPP ejecuta PHP 5.2 por defecto, por lo que goto no funcionará en el entorno de prueba por defecto.

4.6.4 Funciones definidas por el usuario

Tal vez la característica más poderosa de PHP es la capacidad de definir y ejecutar las funciones dentro de su código. Una función es un bloque de código con nombre que se declara dentro de las secuencias de comandos que puede llamar después. Las funciones pueden aceptar cualquier número de argumentos y pueden devolver un valor utilizando la instrucción return.

El formato básico de una función requiere primero identificar la función con la palabra reservada function frente a una cadena que sirve como nombre de la función. Esta cadena puede contener cualquier carácter, alfanuméricos y guiones, pero no debe comenzar con un número. Usted puede adjuntar cualquier argumento que desee entre

paréntesis después del nombre de la función. Tenga en cuenta que todavía debe incluir los paréntesis, incluso si la función no requiere que se pase ningún argumento.

Las palabras reservadas son términos especiales que no se pueden utilizar para los nombres de funciones. Estos incluyen la palabra `function`, los nombres de estructura de control, y varios otros términos que se observaran a medida que surjan.

Comience por la declaración de su primera función:

```
<?php
function Saludo()
{
    echo "Hola amigos míos!";
}
// Ejecución de la función
Saludo();
?>
```

La función produce la siguiente salida cuando se llama:



Ilustración 4.6.4.1 Funciones definidas por el usuario

Para agregar argumentos, se colocan entre paréntesis las variables de la declaración de la función, separados por comas. Puede utilizar estos argumentos en la función para determinar el valor devuelto por la misma:

```
<?php
function encontrar($nombre)
{
    echo "Hola, mi nombre es $nombre. Encantado de conocerte! <br />";
}
encontrar("Ángel");
?>
```

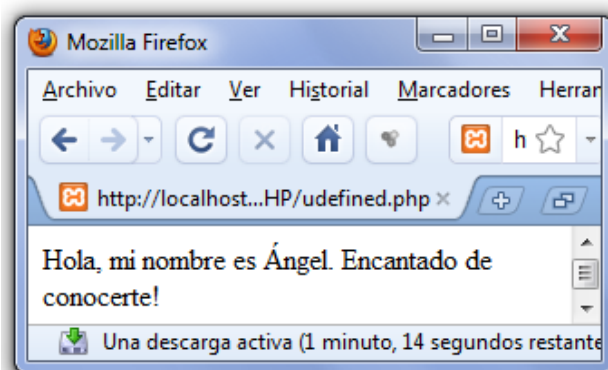


Ilustración 4.6.4.2 Salida del código anterior.

4.6.5 Devolviendo valores desde una función.

La mayoría de las veces, no va a querer el resultado de una función llamada inmediatamente. Para almacenar los resultado en una variable, se utiliza la sentencia return. Agregue el código siguiente a un archivo con extensión .php:

```
<?php
/*
 * Basado en el tiempo pasado a la función en el ejército (24 horas)
 * El tiempo, devuelve un saludo
 */
function saludar($tiempo)
{
    if($tiempo<12)
    {
        return "Buenos días!";
    }
    else if($tiempo<18)
    {
        return "Buenas tardes!";
    }
    else
    {
        return "Buenas noches!";
    }
}
$saludando = saludar(14);
echo "$saludando Como está usted?";
?>
```

El resultado de saludar() en la variable \$saludo, que después puede utilizar para mostrar el tiempo de saludo para el usuario. Cuando se establece 14 (14:00) como parámetro y ejecutar este script en su navegador, se obtiene la siguiente salida:

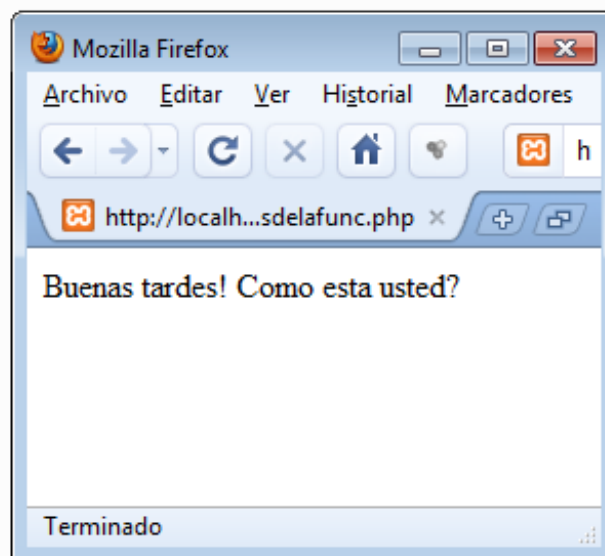


Ilustración 4.6.5.1 Retorno de valores desde una función.

4.7 Pasando información con PHP

Ahora que usted se siente cómodo con los conceptos básicos de PHP, usted está listo para saber más sobre cómo la información se mueve de página en página. Usted necesita esto para casi todos los aspectos del blog, de escribir nuevos mensajes y permitir a los usuarios hacer sus comentarios.

Aprenderemos:

- ¿Qué son las matrices superglobales en PHP y cómo usarlas?
- ¿Cómo enviar datos mediante formularios HTML usando el método POST?
- ¿Cómo enviar datos en la dirección URL usando el método GET?
- ¿Cómo almacenar datos en SESIONES para su uso en varias páginas?
- ¿Cómo crear y almacenar cookies para el retorno de los usuarios?

Pasar datos es lo que separa a las páginas web dinámicas a partir de las estáticas, mediante la personalización de una experiencia basada en las opciones de usuario, usted es capaz de añadir un nuevo nivel de valor a un sitio web.

4.7.1 Matrices Superglobales

PHP ofrece varios tipos de matrices superglobales a los desarrolladores, cada uno con un propósito útil y diferente. Una matriz superglobal, es una variable especial que siempre está disponible en las secuencias de comandos, sin importar el alcance actual de la secuencia de comandos. PHP incluye varias matrices superglobales:

- `$GLOBALS`: Variables disponibles en el ámbito global.
- `$_SERVER`: Información sobre el servidor.
- `$_GET`: Datos que se pasan con el método HTTP GET.
- `$_POST`: Datos que se pasan con el método HTTP POST.
- `$_REQUEST`: Datos que se pasan a través de una solicitud HTTP.
- `$_FILES`: Datos pasados por una entrada de archivo HTML.
- `$_SESSION`: Especifican los datos actuales de las sesiones de los usuarios.
- `$_COOKIE`: Datos almacenados en el navegador del usuario en una cookie.

4.7.2 Ámbito de aplicación de variables.

En la programación, el alcance se refiere al contexto en el que declara una variable, la mayoría de las variables en PHP tienen un único ámbito: global. El uso de este alcance significa que una variable está disponible en la secuencia de comandos que se declara, es así como en cualquier secuencia de comandos que se incluye después de la variable es declarada o de cualquier secuencia de comandos que incluye el archivo en que la variable se declara.

Por ejemplo, pruebe a abrir un nuevo archivo `test.php` y experimente el alcance de las variables:

```
<?php
    $foo = "algún valor";
    include_once 'extras.php'; // $foo esta variable está en extras.php
    $bar = "otro valor"; // $bar esta variable no está en extras.php
    echo "test.php: Foo es $foo, y bar es $bar. <br />";
?>
```

Ahora abra `extras.php` e introduzca el siguiente código:

```
<?php
echo "extras.php: Foo es $foo, y bar is $bar. <br />";
?>
```

Cuando usted cargue en su navegador la página, esta será la salida:

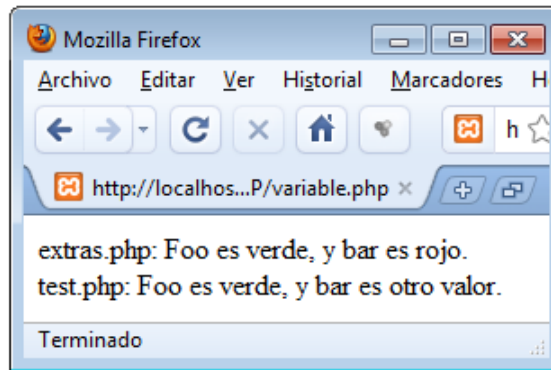


Ilustración 4.7.2.1 Ámbito de aplicación de variables.

Cuando `error_reporting (E_ALL)` está activo, se mostrará un aviso, lo que le permite saber que `$ bar` es una variable no definida. El informe de errores es una buena manera de asegurarse de que todas las bases están cubiertas en los scripts. Con el desarrollo de aplicaciones, para mejores prácticas debe usar `error_reporting (E_ALL)`, lo que significa que todos los errores y avisos se mostraran. No queremos que nuestros usuarios vean los códigos de errores desagradables en los scripts desde su navegador.

El ámbito de aplicación cambiará un poco cuando empiece a usar funciones, porque las variables declaradas dentro de una función tienen ámbito local, lo que significa que están disponibles sólo dentro de la función que las declara. Además, las variables declaradas en el ámbito global sólo están disponibles si se declaran explícitamente como globales dentro de la función.

Ahora está listo para ver cómo `error_reporting (E_ALL)` afecta a las secuencias de comandos. En `test.php`, escribir el siguiente código:

```
<?php
    error_reporting(E_ALL);
    $foo = "Estoy fuera de la función!";
    function test()
    {
        return $foo;
    }
    echo test(); // El aviso que emite $foo es indefinido
?>
```

La variable `$foo` es indefinida si se ejecuta en la función del archivo `Test()`, por lo que emite un aviso. Usted puede aclarar esto porque se declara `$foo` como global dentro de su función `test()`, lo que significa que `$foo` en el ámbito local se referirá a la variable `$foo` en el ámbito global:

```
<?php
    $foo = "Estoy fuera de la función!";
    function test()
    {
        global $foo; // Declaración de la variable $foo como global
        return $foo;
    }
    echo test();
?>
```


Visitar test.php para ver que se ejecuta test(), se puede ver el resultado en esta salida:

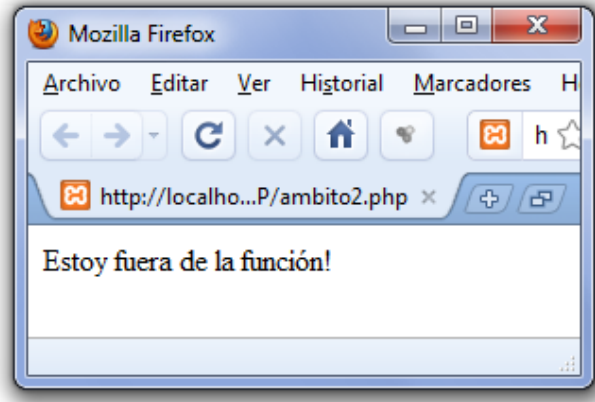


Ilustración 4.7.2.2 Salida de la función test.php.

Una variable declarada dentro de una función no está disponible fuera de esa función a menos que se especifique como valor de retorno de la función. Por ejemplo, considere el siguiente código:

```
<?php
error_reporting(E_ALL);
function test()
{
    $foo = "Declarada dentro de una función. <br />";
    $bar = "También declarado dentro de una función. <br />";
    return $bar;
}
$baz = test();
/*
 * Los anuncios que se emiten foo $ y $ bar no están definidos
 */
echo $foo, $bar, $baz;
?>
```

Este código muestra los siguientes resultados:

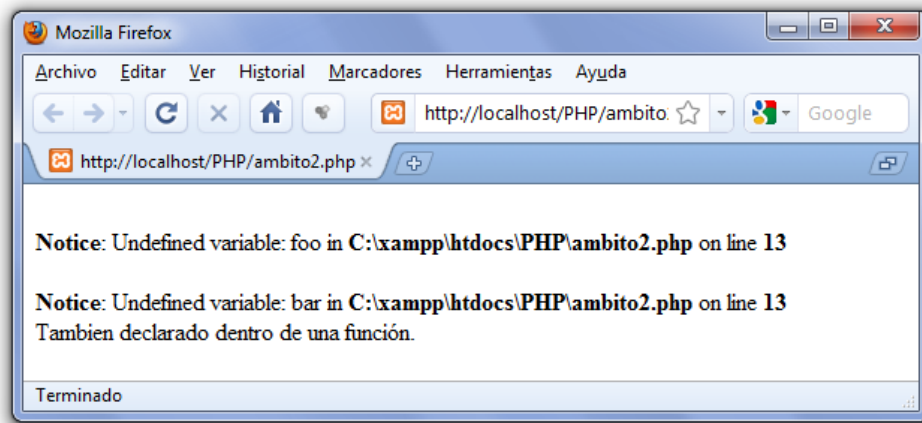


Ilustración 4.7.2.3 Variables declaradas dentro de una función.

Tienes que declarar dos variables dentro de una función y devolver ambos y, seguidamente, se utiliza una matriz y la list() para acceder a los valores fácilmente.

La ejecución de este código produce la salida deseada:

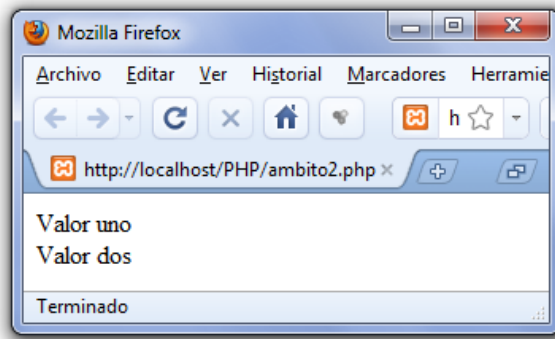


Ilustración 4.7.2.4 Retornando una lista

4.7.3 \$GLOBALS

PHP proporciona otra opción para acceder a las variables en el ámbito global: la matriz \$GLOBALS superglobales. Todas las variables en el ámbito global se cargan en el array \$GLOBALS, lo que le permite acceder a ellos utilizando el nombre de variable como la clave de la matriz.

Puede probar esta matriz en test.php:

```
<?php
    $foo = "Algunos valores.";
    function test()
    {
        echo $GLOBALS['foo'];
    }
    test();
?>
```

Este código genera el siguiente resultado:

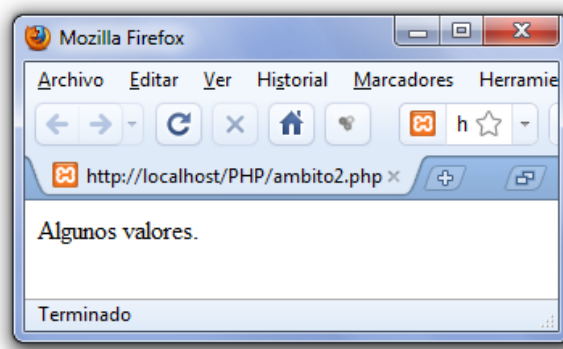


Ilustración 4.7.3.1 Variables globales.

Por lo general, una buena práctica es evitar el uso de variables globales siempre que sea posible. El método preferido de acceso a variables globales dentro de las funciones es pasarlos como argumentos. Esto hace las secuencias de comandos más fáciles de leer, y simplifica el mantenimiento a largo plazo.

4.7.4 \$_SERVER

Los superglobales \$_SERVER almacenan información sobre el servidor y el script actual. También tiene características que le permiten acceder a la dirección IP de un visitante del sitio, qué sitio se refiere al visitante a este

script, y muchas otras piezas de información útil. Puede ver el manual de PHP para obtener más información acerca de esta característica.

Una de las piezas de información más útil disponible en las superglobales es `$_SERVER`, el nombre del lugar de acogida, que se almacena en `HTTP_HOST`. El nombre del sitio de acogida es útil porque le permite crear una plantilla sencilla que se puede utilizar a través de diferentes proyectos, sin necesidad de cambiar su código.

Por ejemplo, puede utilizar el siguiente fragmento de código para dar la bienvenida a un visitante a su sitio:

```
<?php
    echo "<h1> Bienvenido a $_SERVER[HTTP_HOST]! </h1>";
?>
```

La ejecución de este código en `test.php` produce esta salida:

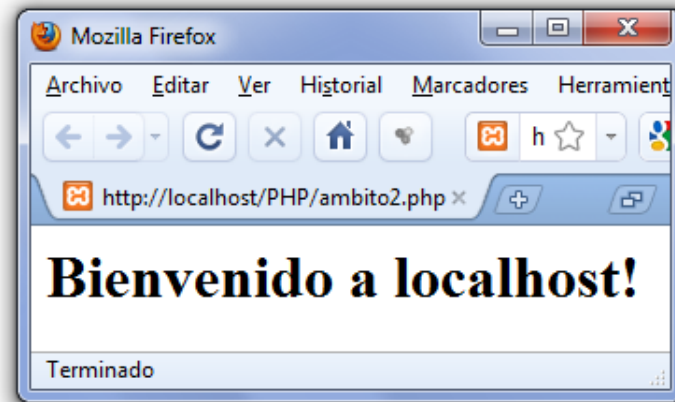


Ilustración 4.7.4.1 Utilizando `$_SERVER`

A medida que continúe adelante con la creación de un blog simple, usted descubrirá algunas aplicaciones para `HTTP_HOST` en sus programas.

Observe el uso de `print_r()` en la parte inferior de la secuencia de comandos. Esta es una excelente forma de depuración de código, especialmente las matrices, ya que genera una visualización "legible" de los contenidos de una variable. Usted puede ver esto en el trabajo cuando se carga el script de prueba en `test.php`:

```
<?php
/*
 * Tenga en cuenta que las etiquetas <pre> y caracteres de nueva línea (\n) se utilizan
 * en aras de la legibilidad
 */
// Ruta al archivo actual (es decir, / simple_blog / test.php)
echo $_SERVER['PHP_SELF'], "\n\n";

// La información sobre el navegador del usuario
echo $_SERVER['HTTP_USER_AGENT'], "\n\n";

// Dirección de la página que se refiere al usuario (si hay)
echo $_SERVER['HTTP_REFERER'], "\n\n";

// Dirección IP desde la que el usuario está viendo la secuencia de comandos
echo $_SERVER['REMOTE_ADDR'], "\n\n";

//exportación Humanos de lectura del contenido de $_SERVER
print_r($_SERVER);

?>
```

Cargando test.php en su navegador de salida muestra algo como lo siguiente:

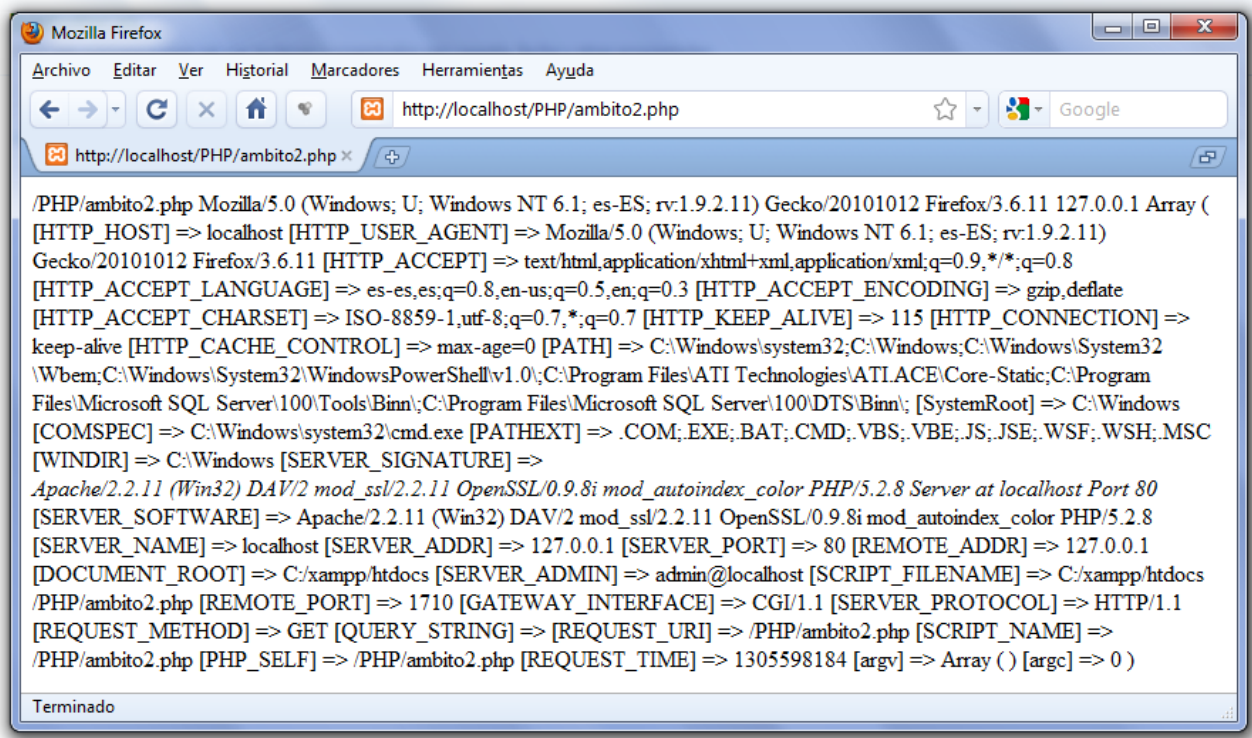


Ilustración 4.7.4.2 Utilización de \$_SERVER

4.7.5 \$_GET

Uno de los dos métodos más comunes de transmisión de datos entre páginas es el método GET. En GET los datos se pasan a través de una cadena de consulta en la URL, que se ve algo como esto:

<http://example.com?var1=somevalue&var2=othervalue>

La cadena de consulta se inicia con un signo de interrogación (?) y, a continuación los nombres de una variable. La siguiente parte de la cadena de consulta es un signo de igual (=), seguido de un valor. Usted puede añadir más variables añadiendo un signo (&) al final del primer valor, a continuación, declarar otra variable, seguido por un signo igual y un valor. Tenga en cuenta que el no incluir los valores de la cadena de consulta entre comillas, pueden causar problemas cuando se utilizan los valores con espacios y / o caracteres especiales.

4.7.6 Codificación de URL

Afortunadamente, PHP incluye el urlencode(), el cual puede utilizar para obtener los valores de lista para ser aprobada en una dirección URL. Esta función se utiliza en conjunto con urldecode(), las dos funciones permiten pasar valores complejos a través de la URL.

4.7.7 Acceso a variables de URL

Se utiliza la matriz \$_GET superglobales para acceder a las variables de la cadena de consulta. Puede acceder a las variables en esta matriz con el nombre de la variable como la clave de la matriz. Por ejemplo, puede utilizar este código para acceso a las variables de la URL de ejemplo al principio de esta sección:

```
<?php
echo "var1: ", $_GET['var1'], "<br />";
echo "var2: ", $_GET['var2'], "<br />";
?>
```

La ejecución de este código genera el siguiente resultado

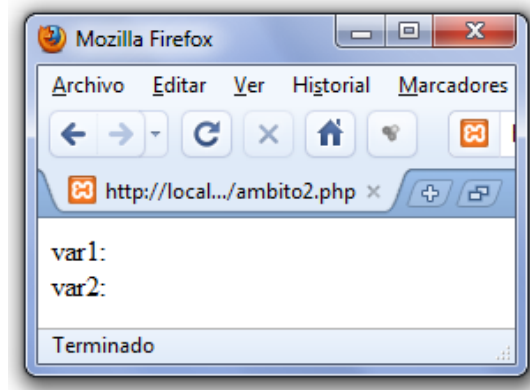


Ilustración 4.7.7.1 Variables de URL.

El superglobal `$_GET` le permite determinar la información que se muestra en una página, dependiendo de los valores que se pasan en la URL. Por ejemplo, puede utilizar este código si desea permitir a un usuario una breve descripción de su sitio o su información de contacto:

```
<ul id="menu">
<li> <a href="test.php">Casa</a> </li>
<li> <a href="test.php?page=about">Acerca de nosotros</a> </li>
<li> <a href="test.php?page=contact">Contáctenos</a> </li>
</ul>

<?php
/*
 * Medida de seguridad básica para garantizar que
 * la variable página ha sido aprobado, lo que le permite
 * evitar daño muy básico utilizando htmlentities ()
 */
if(isset($_GET['page'])) {
    $page = htmlentities($_GET['page']);
} else {
    $page = NULL;
}
switch($page) {
    case 'about':
        echo "
        <h1> Acerca de nosotros </h1>
        <p> Somos desarrolladores rockin 'web! </p>";
        break;
    case 'contáctenos':
        echo "
        <h1> Nuestros contactos </h1>
        <p> Envíanos un email a
        <a href='mailto:info@example.com'>
        info@example.com
        </a>
        </p>";
        break;
}
/*
 * Crear una página por defecto en caso de que ninguna variable sea pasada
```

```
*/
default:
echo "
<h1> Seleccione una página! </h1>
<p>
Elija una página de arriba
para aprender más sobre nosotros!
</p>";
break;
}
?>
```

Cargando la página muestra una pantalla similar a lo que se ve en la ilustración 4.7.7.2

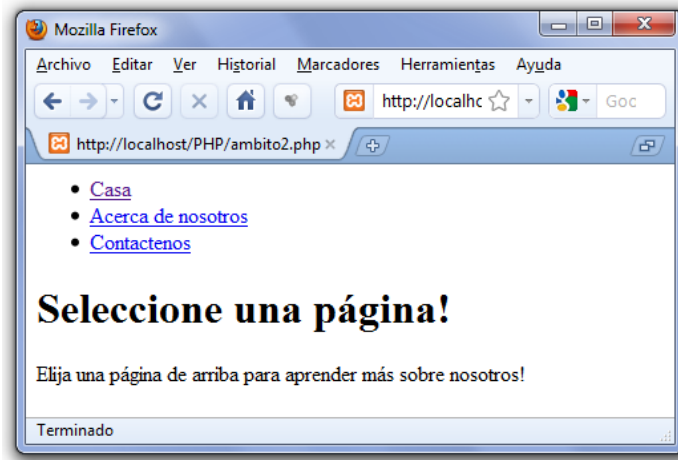


Ilustración 4.7.7.2 Iniciando un blog.

Las variables `$_GET` son útiles, también permiten a los usuarios maliciosos pasar información directamente en su guión. Debido a esto, siempre se debe desinfectar la entrada. También debe evitar el envío de información sensible a través de una solicitud GET. Las precauciones básicas de seguridad serán cubiertos más adelante.

4.7.8 `$_POST`

PHP provee un segundo método para enviar datos: **POST**. Presentación de la mayoría de los formularios web (como un contacto formulario o registro en el sitio) exige la utilización del método POST para enviar la información del formulario para su procesamiento.

Estructuralmente, las solicitudes POST son muy similares a las solicitudes GET, y acceder a sus valores se hace de la misma manera: se utiliza el superglobales `$_POST` y añadir el nombre de la variable como la clave de la matriz. En la mayoría de los casos, un valor POST se pasa de un formulario web que se genera en HTML. Por ejemplo, hacer una búsqueda en este formulario de inscripción:

```
<form action="test.php" method="post">
  <input type="text" name="username" />
  <input type="text" name="email" />
  <input type="submit" value="Registrar!" />
</form>
```

El atributo nombre que le dice a la matriz las claves a utilizar para acceder a la información presentada con el formulario. Por ejemplo, se utiliza la variable `['username']` los `$_POST` en la secuencia de comandos para tener acceso al campo "nombre de usuario".

Abra el archivo Test.php para construir un guión simple de manipulación de formulario.

```
<?php
/*
 * Comprueba si el formulario se ha enviado
 */
if($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Muestra la información presentada
    echo "Gracias por registrarse! <br />",
    "Username: ", htmlentities($_POST['username']), "<br />",
    "Email: ", htmlentities($_POST['email']), "<br />";
} else {
    // Si el formulario no se presentó, muestra el formulario
?>

<form action="test.php" method="post">
<label for="username">Nombre de Usuario:</label>
<input type="text" name="username" />
<label for="email">Email:</label>
<input type="text" name="email" />
<input type="submit" value="Register!" />
</form>
<?php } // Fin de la sentencia else ?>
```

Cuando http://localhost/simple_blog/test.php carga, el script comprueba si los datos han sido presentados a través del método POST. Si es así, el script da gracias al usuario por el registro y muestra el nombre de usuario seleccionado y dirección de correo electrónico proporcionada. Si no se presenta a través de POST, se muestra el script el formulario de inscripción.

4.7.9 \$_REQUEST

El superglobal `$_REQUEST` es una matriz que contiene los contenidos de la `$_GET`, `$_POST`, y `$_COOKIE`. Si modifica el formulario en `test.php` a `someterse a test.php? presentar = true` con un nombre de usuario y la introducción de texto por correo electrónico, puede acceder a todos los datos presentados con el siguiente código:

```
<?php
// Comprueba si el formulario se ha enviado
if($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Salida de los contenidos de $_REQUEST
    foreach($_REQUEST as $key => $val) {
        echo $key, " : ", $val, "<br />";
    }
} else {
    // Si el formulario no se presentó, muestra el formulario HTML
    ?>
    <form action="request.php?submit=true" method="post">
        <label for="username">Nombre de Usuario:</label>
        <input type="text" name="username" />
        <label for="email">Correo Electrónico:</label>
        <input type="text" name="email" />
        <input type="submit" name="submit" value="Registro!" />
    </form>
<?php } // Fin de la declaración else ?>
```

La producción de este script se ve algo como esto:

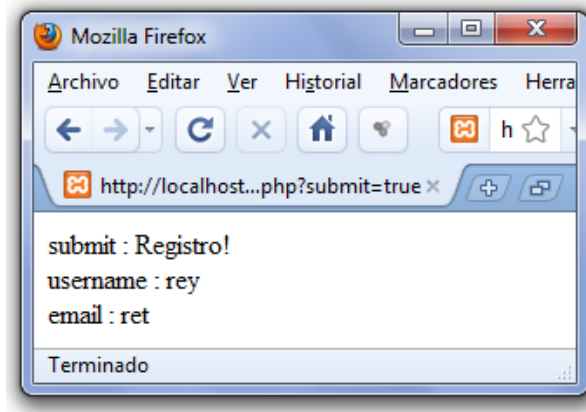


Ilustración 4.7.9.1 Salida de la sentencia \$_REQUEST.

La prioridad de los valores es importante en este caso. La declaración de un índice de \$_COOKIE, \$_POST, y \$_GET con el mismo nombre podría confundir a los nuevos desarrolladores. \$_REQUEST causa que las \$_COOKIE reemplacen los \$_POST y \$_GET. Si \$_COOKIE no contiene el nombre de la variable, \$_POST anulará \$_GET. Por ejemplo, si establece el atributo de acción a test.php? variable = nombre de usuario, usted no verá ningún cambio en el valor nombre de usuario porque \$_POST anulará \$_GET, incluso si el campo nombre de usuario se deja en blanco.

4.7.10 \$_FILES

Otra característica de los formularios HTML es la capacidad de permitir a los usuarios subir archivos. En una aplicación como en un blog, tiene que ser capaz de aceptar imágenes e incluir en sus entradas. Con el fin de acceder y subirla a un archivo, es necesario utilizar el superglobales \$_FILES.

\$_FILES funciona un poco diferente a los \$_POST y \$_GET en que cada archivo crea una matriz de elementos relacionados que proporcionan información sobre el archivo subido. La información proporcionada incluye:

nombre: El nombre del archivo

Tipo: El tipo de archivo (por ejemplo, la imagen / .jpeg)

tmp_name: La ubicación temporal del archivo subido

Error: un código de error correspondiente al tipo de error (0 si no hay errores encontrados)

Tamaño: El tamaño del archivo en bytes (por ejemplo, 9347012)

Puede almacenar cada archivo subido como una matriz multidimensional en la super \$_FILES, que accede utilizando primero el nombre del campo como la clave de serie y, a continuación el nombre del valor del campo deseado (por ejemplo, \$_FILES['nombre']['upload1']).

Puede experimentar con \$_FILES mediante la creación de un archivo test.php en el campo de carga. Recuerde agregar enctype = "multipart / form-data" a la etiqueta <form> para permitir la subida de archivos:

```
<?php
// Comprueba si el formulario se ha enviado
if($_SERVER['REQUEST_METHOD'] == 'POST') {
// Comprueba si un archivo se ha subido sin errores
    if(isset($_FILES['foto']))
        && is_uploaded_file($_FILES['foto']['tmp_name'])
        && $_FILES['foto']['error'] == UPLOAD_ERR_OK {
            // Muestra el contenido de $_FILES
            foreach($_FILES['foto'] as $key => $value) {
                echo "$key : $value <br />";
            }
        }
    } else {
```



```
        echo "No existe el fichero subido!";
    }
} else {
    // Si el formulario no se presentó, muestra el formulario HTML
?>
<form action="files.php" method="post"
enctype="multipart/form-data">
<label for="photo">Foto del Usuario:</label>
<input type="file" name="foto" />
<input type="submit" value="Subir foto" />
</form>
<?php } // Fin de la sentencia else ?>
```

Al ejecutar este script en test.php muestra la pantalla mostrada en la Figura 4.7.10.1.

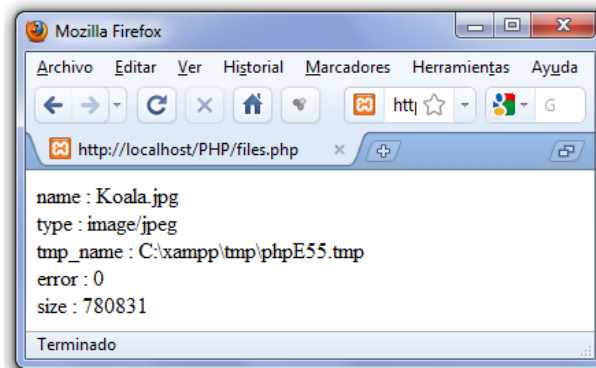


Ilustración 4.7.10.1 \$_FILE.

El uso de un archivo subido es un poco más complicado. Sin embargo, usted puede agregar este script para test.php para ver un ejemplo muy básico de la utilización de archivos subidos:

```
<?php
// Comprueba si el formulario se ha enviado
if($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Comprueba si el archivo fue cargado sin errores
    if(isset($_FILES['foto'])
        && is_uploaded_file($_FILES['foto']['tmp_name'])
        && $_FILES['foto']['error'] == UPLOAD_ERR_OK) {
        // Comprueba si el archivo es una imagen JPG
        if($_FILES['foto']['type'] == 'image/jpeg') {
            $tmp_img = $_FILES['foto']['tmp_name'];
            // Crea un recurso de imagen
            $image = imagecreatefromjpeg($tmp_img);
            // Indica al navegador el tipo de archivo
            header('Content-Type: image/jpeg');
            // Salidas el archivo en el navegador
            imagejpeg($image, "", 90);
            // Libera la memoria utilizada para el archivo
            imagedestroy($image);
        } else {
            echo "El archivo subido no era una imagen JPG.";
        }
    } else {
        echo "No hay foto subida!";
    }
} else {
    // Si el formulario no se presentó, muestra el formulario HTML
?>
<form action="files.php" method="post"
    enctype="multipart/form-data">
```

```
<label for="foto">Foto del Usuario:</label>
<input type="file" name="foto" />
<input type="submit" value="Subir una foto" />

</form>
<?php } // Fin declaración else
?>
```

Una vez que un usuario selecciona un archivo JPG y hace clic en el botón "Subir fotos", el script implementa una serie de acciones:

1. Se comprueba que el formulario se ha enviado mediante el método POST:
 - a. Si no, se muestra la forma y la transformación posterior no se realiza.
 - b. En caso afirmativo, se procede al Paso 2.
2. Comprueba si un archivo se ha subido y que el archivo no tiene errores:
 - a. Si ambas condiciones no se cumplen, el guión despliega un mensaje de error.
 - b. Si se cumplen ambas condiciones, se procede al paso 3.
3. Se verifica que el archivo subido es una imagen JPG:
 - a. Si no, la secuencia de comandos genera un mensaje de error.
 - b. En caso afirmativo, se procede al paso 4.
4. Utiliza `imagecreatefromjpeg()` para crear un recurso de imagen del archivo temporal.
5. Se envía un encabezado Content-Type al navegador, por lo que la imagen se controla adecuadamente.
6. Genera la imagen utilizando `imagejpeg()`.
7. Utiliza `imagedestroy()` para liberar la memoria consumida por el recurso de imagen.

Después de que el archivo se carga, la secuencia de comandos muestra la imagen en el navegador (vea la Figura 4.7.10.2).



Ilustración 4.7.10.2 Subiendo una foto a la web.

4.7.11 \$_SESSION

Cuando trabajamos con sesiones en PHP estamos ofreciendo la capacidad de guardar variables (incluyendo matrices y clases). Entre ejecución de secuencias de comandos y llamarlas posteriormente. Para que funcione este sistema, el servidor Web debe de ser capaz de identificar un explorador Web y aquí es donde las cookies representan su papel. Al contrario que el uso de cookies a un usuario, las sesiones no pueden guardar en realidad ninguna información importante en la máquina del cliente. Las sesiones funcionan sobre el concepto de que cada explorador cliente individual tiene un “justificante” (denominado id de sesión) que después se presenta al servidor Web durante todas las solicitudes.

En caso de que usted necesite almacenar un valor para toda la visita de un usuario, el superglobales \$_SESSION proporciona una solución práctica y fácil. Cuando una variable \$_SESSION se declara, permanece en la memoria hasta que esté expresamente. Sin esto, las sesiones (el valor predeterminado de tiempo de espera es de 180 minutos), o el navegador se cierran.

Un uso común de las variables \$_SESSION es almacenar el estado de conexión de un usuario. Usted puede construir una nueva cuenta para crear una sesión de un usuario que ya ha registrado.

Su script realizara las siguientes tareas:

1. Mostrar un formulario de registro.
2. Da gracias al usuario por el registro después de haberse registrado.
3. Dígle al usuario que ya se ha registrado en las cargas de la página consecuente.

Debe iniciar sesión antes de los datos para poder utilizar las variables de sesión; debe hacerlo llamando a session_start() en la parte superior de la secuencia de comandos.

Nota: Si usted llama a session_start() más de una vez, PHP genera un aviso y las llamadas adicionales a session_start() serán ignoradas. Además, usted debe llamar a session_start() al principio de cualquier secuencia de comandos que se utilice, antes de enviar cualquier salida al navegador. De lo contrario, también generará un aviso y evita que el período de sesiones se inicie. Por último, el script que inicia la sesión debe ser lo primero en el archivo, así que asegúrese de colocar la apertura <?php en la línea 1 del archivo.

Puede implementar una variable \$_SESSION insertando el siguiente código en test.php:

```
<?php
// Inicializar los datos de sesión
session_start();
/*
 * Si el usuario ya está registrado, mostrar un
 * Mensaje * haciéndoles saber.
 */
if(isset($_SESSION['Nombre-Usuario'])) {
    echo "Usted ya está registrado como $_SESSION[Unombre]";
}
// Comprueba si el formulario se ha enviado
else if($_SERVER['REQUEST_METHOD'] == 'POST') {
    /*
     * Si tanto el nombre de usuario y los campos de correo electrónico se llenan,
     * guardar el nombre de usuario en una variable de sesión y
     * La salida de un mensaje de agradecimiento al navegador. Para
     * eliminar los espacios en blanco iniciales y finales, se utiliza la
     * función trim().
     */
    if(!empty(trim($_POST['Nombre-Usuario']))
    && !empty(trim($_POST['email']))) {
        // El almacenamiento escapó $_POST los valores de las variables
        $Unombre = htmlentities($_POST['Nombre-Usuario']);
        $email = htmlentities($_POST['email']);
        $_SESSION['Nombre-Usuario'] = $Unombre;
        echo "Gracias por registrarse! <br />"
    }
}
```

```
"Nombre de Usuario: $Unombre <br />",  
"Email: $email <br />";  
}  
/*  
 * Si el usuario no rellena los dos campos, en la pantalla aparece  
 * un mensaje haciéndoles saber que ambos campos son  
 * requeridos para el registro.  
 */  
else {  
    echo "Por favor, rellenar los dos campos! <br />";  
}  
}  
// Si el formulario no se presentó, muestra el formulario HTML  
else {  
    ?>  
    <form action="files.php?Nombre-Usuario=overwritten" method="post">  
    <label for="Nombre-Usuario">Nombre del Usuario:</label>  
    <input type="text" name="Nombre-Usuario" />  
    <label for="email">Email:</label>  
    <input type="text" name="email" />  
    <input type="submit" value="Registro!" />  
    </form>  
    <?php } // Fin de la declaración else  
    ?>
```

Usted debe ver a su formulario de registro después de cargar test.php en su navegador. y la secuencia de comandos mostrará el mensaje siguiente después de rellenar los campos:

Al recargar la página debe aparecer el siguiente mensaje:

Utiliza unset () para destruir una variable \$ _SESSION. Extracción de la variable de sesión nombre de usuario y que muestra el formulario de inscripción de nuevo es tan sencillo como insertar el siguiente código:

```
unset($_SESSION['username']);
```

4.7.12 Usando session_destroy()

PHP proporciona una función llamada session_destroy(), cuyo nombre puede ser algo engañoso. Cuando se llama a esta función, una sesión será destruida la próxima vez que se carga la página. Sin embargo, las variables de sesión siguen estando disponibles con el mismo guión. Inserte este código en test.php para ver este comportamiento en acción:

```
<?php  
error_reporting(E_ALL);  
session_start();  
// Crear una variable de sesión  
$_SESSION['prueba'] = "A value.";  
// Destruye la sesión  
session_destroy();  
// Intento de salida de la variable de sesión (de salida:.. Valor)  
echo $_SESSION['prueba'];  
// Desactivar la variable específica  
unset($_SESSION['prueba']);
```

```
// Intento de salida de la variable de sesión (genera un aviso)
echo $_SESSION['prueba'];
?>
```

La carga de este script en el navegador genera el siguiente resultado:

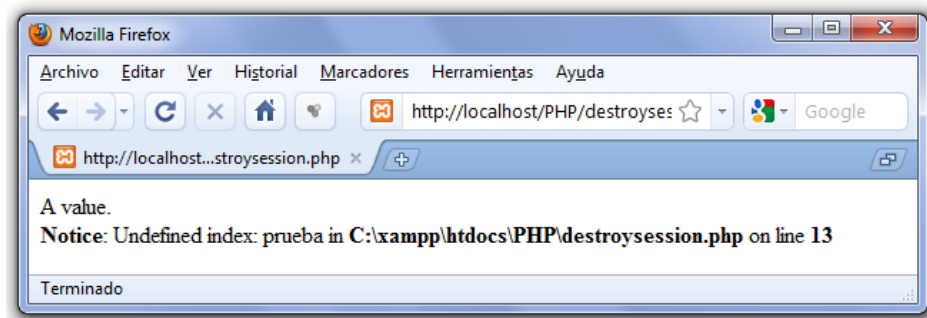


Ilustración 4.7.12.1 Destruyendo una variable de tipo SESSION.

Este comportamiento puede ser confuso, por lo que debe especificarse `unset()` sobre las variables de sesión que necesitan destruir, antes del final de la secuencia de comandos para evitar un comportamiento inesperado.

4.7.13 \$_COOKIE

Al contrario que nuestros programas favoritos que podemos ejecutar en el escritorio, como un procesador de textos, PHP no puede recordar una solicitud pasada. La única información que tiene disponible PHP cuando se ejecuta una secuencia de comando es la información proporcionada durante la solicitud HTTP. ¿Y si quisiéramos crear un sitio Web que recordara el nombre de todos los visitantes y los utilizara para generar contenido personalizado desde nuestras secuencias de comandos PHP? ¿Cómo podríamos distinguir entre un visitante y otro en nuestras secuencias de comandos PHP? Este problema se soluciona de forma elocuente, utilizando las **cookies**.

Cuando visitamos un sitio Web, el servidor nos proporciona una cookie que nos identifica. La próxima vez que visitemos el sitio Web (o incluso otra página de dicho sitio Web), devolveremos la cookie de nuevo al servidor, que la utilizará para determinar quiénes somos.

Las cookies se comportan de manera similar a las sesiones, pero estas le permiten almacenar información en la máquina de un usuario por período más largo de tiempo. La información almacenada en las cookies permanece disponible incluso después de que el usuario cierra su navegador, suponiendo que la fecha de caducidad indicada en la cookie se establece lo suficientemente lejos en el futuro.

Usted puede utilizar cookies para hacer repetir a un usuario visitas más agradables mediante la retención correspondiente, no sensibles a datos. Esto puede incluir la configuración que personalizan la experiencia del usuario en su sitio, lleve a cabo tareas repetitivas de forma automática, o permitir a un usuario permanecer conectado y evitar el paso adicional de registro en cada visita.

En PHP, se utiliza la super-global `$_COOKIE` para acceder a los valores de cookie. Sin embargo, el establecimiento de una cookie requiere el uso de la función `setcookie()`. Abrir `test.php` y establecer de nuevo una cookie que contiene el nombre de usuario y darle la bienvenida cada vez que se carga la página.

Nota: Este fragmento utiliza la función `time()`. El valor devuelto por `time()` es el número de segundos desde la época Unix¹, que es comúnmente utilizado en la programación para determinar las fechas y los intervalos. “La época Unix fue la medianoche el horario universal coordinado (UTC) del 1 de enero de 1970. Sirve como una vara de medir para determinar el tiempo cuando se trata de sistemas UNIX”.

```
<?php
/*
 * Si el usuario regresa a esta página
 * después de registrarse previamente, utilice la
```

```
* "cookie" para darles la bienvenida de nuevo.
*/
if(isset($_COOKIE['Nombre-Usuario'])) {
    echo "Bienvenido Nuevamente, ",
    htmlentities($_COOKIE['Nombre-Usuario']),
    "! <br />";
}
/*
* Si el usuario es nuevo y presenta el
* formulario de registro, una cookie con
* su nombre de usuario le mostrar un agradecimiento
* y el mensaje.
*/
else if($_SERVER['REQUEST_METHOD']=='POST'
&& !empty($_POST['Nombre-Usuario'])) {
    // Desinfectar la entrada y almacenar en una variable
    $nombre = htmlentities($_POST['Nombre-Usuario']);
    // Establecer una cookie que expira en una semana
    $expires = time()+7*24*60*60;
    setcookie('Nombre-Usuario', $nombre, $expires, '/');
    // Emite un mensaje de agradecimiento
    echo "Gracias por registrarse, $nombre! <br />";
}
/*
* Si el usuario no se ha registrado anteriormente
* o rellenado el formulario de inscripción, se muestran los
* Formulario de inscripción.
*/
else {
    ?>
    <form method="post">
    <label for="Nombre-Usuario">Nombre de Usuario:</label>
    <input type="text" name="Nombre-Usuario" />
    <input type="submit" value="Registro" />
    </form>
    <?php } // Fin de la sentencia else
    ?>
```

Como en los ejemplos anteriores, un usuario podrá ver el formulario de inscripción la primera vez que navega a test.php. Después de llenar y enviar el formulario, verá el mensaje siguiente, asumiendo que entró en el nombre de usuario, "el mismo nombre inicial":

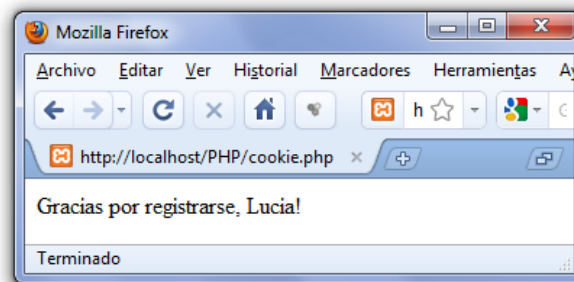


Ilustración 4.7.13.1 Cuando el usuario carga la página y se registra.

Las visitas subsiguientes cargarán la variable \$_COOKIE y desplegará este mensaje en su pantalla:

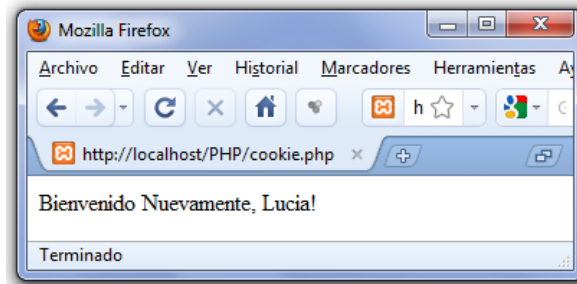


Ilustración 4.7.13.2 Cuando el usuario accede nuevamente a la página.

Los navegadores les permiten a los usuarios rehusarse a las cookies, es buena idea no usarlas en las partes vitales de sus aplicaciones. Cuando pruebe una nueva aplicación que utilice cookies, debería habilitar el uso de cookies en su navegador y asegurarse que los usuarios pueden beneficiarse y utilizar su aplicación sin necesidad de cookies.

4.8 PHP Orientado a Objetos.

Vamos a aclarar las cosas desde el principio: PHP no es un lenguaje orientado a objetos, pero tiene amplias funciones de programación orientada a objetos. Estos se sometieron a revisión exhaustiva y la mejora en PHP 5 fue lanzada en julio de 2004, PHP 5 y la programación orientada a objetos (POO), el modelo sigue siendo esencialmente sin cambios en PHP 6. El propósito de este documento es ayudarle a aprovechar esas características para que sea más fácil de reutilizar el código en una variedad de situaciones. Supongo que usted está familiarizado con los conceptos básicos de PHP, como las variables, matrices y funciones. Si no es así, este material no es para usted al menos no todavía. Le sugiero que comience por el principio es decir, por lo más básico.

Las técnicas y los códigos utilizados requieren PHP 5 y PHP 6. No vamos a trabajar con PHP 4.

Las técnicas que enseñaremos en esta parte están destinadas a mejorar la forma de trabajar con PHP, no reemplazar todo lo que has aprendido hasta ahora. Sin embargo, si usted decide profundizar en programación orientada a objetos, que establecen una base sólida para estudios posteriores. Encontrarás los conocimientos en este libro PHP-Object-Oriented Solutions., si se va a utilizar un framework PHP, como el Zend Framework ([www.zend.com / es / comunidad / marco](http://www.zend.com/es/comunidad/marco)). Aunque trabajar con marcos puede tomar una gran parte del trabajo escribiendo código, sin un conocimiento práctico de programación orientada a objetos, se le pierde por completo.

¿Para que se usa la programación orientada a objetos?

La respuesta es para romper el código largo, de procedimiento en unidades discretas de la programación lógica. En muchos sentidos, esto es similar a la creación de funciones personalizadas. Sin embargo, la programación orientada a objetos toma las cosas un paso más allá mediante la eliminación de todas las funciones desde el script principal, y agrupándolos en unidades especializadas llamadas clases. El código dentro de la clase hace todo el trabajo sucio, la manipulación real de los datos-dejando el script principal como un conjunto de instrucciones de alto nivel. Para tomar un ejemplo común que será familiar para los desarrolladores de PHP, antes de utilizar la entrada de un formulario en línea, usted necesita asegurarse que no contiene nada que pueda ser utilizado para sabotear la base de datos.

El enfoque de un procedimiento se ve en el proyecto específico, y escribe el código a la medida, por lo general una serie de instrucciones condicionales diseñado para comprobar cada campo de entrada, a su vez. Por ejemplo, este tipo de código se utiliza comúnmente para asegurarse de que un nombre de usuario cumple con la cantidad de caracteres mínimos requeridos:

```
if (strlen($username) < 6 || strlen($username) > 12) {  
    $error['username'] = 'El nombre de usuario debería estar entre 6 y 12 caracteres';  
}
```


En la programación orientada a objetos se ve la programación de una manera más genérica. En lugar de preguntar "¿Cómo puedo validar este formulario?" el enfoque orientado a objetos es preguntar "¿Cómo puedo validar cualquier formulario?" Lo hace mediante la identificación de tareas comunes y la creación de funciones genéricas (o métodos, como son llamados en Programación Orientada a Objetos) para manejarlos. La comprobación de la longitud del texto es una tarea, que tiene sentido tener un método que comprueba la longitud de cualquier campo de entrada y genera automáticamente el mensaje de error. La definición del método se encuentra escondida en el interior del archivo de la clase, dejando algo como esto en el script principal:

```
$val->checkTextLength('username', 6, 12);
```

En esta etapa, no se preocupe por lo que el código parece o cómo funciona (estos objetos tienen un enfoque de validación de entrada). No te preocupes por los términos, la clase y el método, se describirá más adelante.

El enfoque adoptando la POO tiene dos ventajas:

Reutilización de código: Romper las tareas complejas en módulos genéricos que hacen mucho más fácil de reutilizar el código. Los archivos de clase son normalmente independientes de la secuencia de comandos principal, para que puedan desplegarse rápidamente en diferentes proyectos.

Mantenimiento más fácil y fiabilidad: La concentración en las tareas del género significa que cada método definido en una clase normalmente se maneja en una sola tarea. Esto hace que sea más fácil identificar y eliminar errores. La naturaleza modular de código almacenado fuera del guion principal significa que, si surge un problema, lo arreglamos en un solo lugar. Una vez que una clase ha sido ensayada y probada, se puede tratar como una caja negra, y se usan para producir resultados consistentes.

Esto hace que el desarrollo de proyectos complejos en equipos mucho más fáciles. Los desarrolladores individuales no necesitan preocuparse de lo que sucede dentro de una unidad en particular, todo lo que importa es que produce el resultado esperado.

4.8.1 Usando Clases y Objetos

El bloque de construcción fundamental de todo el código orientado a objetos se llama clase. Esto es simplemente una colección de variables y funciones relacionadas, todo ello envuelto en un par de llaves y etiquetados con el nombre de la clase. En programación orientada a objetos, una variable asociada a una clase se refiere a una propiedad, y una función que se llama método. Si usted ha construido una biblioteca de sus propias funciones personalizadas de PHP para su reutilización en diferentes proyectos, la creación de una clase le resultará muy familiar.

No todas las variables en una clase son las propiedades. Una propiedad es un tipo especial de variable que puede utilizarse en cualquier lugar dentro de una clase y, a veces fuera también.

Por lo tanto, si desea validar la entrada de usuario, puede crear una clase llamada Validador (por convención, los nombres de las clases siempre empiezan con letra capital) para agrupar una serie de métodos (en otras palabras, las funciones) para realizar pruebas como las siguientes:

- ¿Es este un número?
- ¿Está dentro de un rango específico?
- ¿Es esto una dirección válida de correo electrónico?
- ¿Este texto tiene un contenido potencialmente dañino?

Cuando se desea utilizar cualquiera de las propiedades de la clase o métodos, es necesario crear una instancia de la clase utilizando la palabra clave new como esto:

```
$val = new Pos_Validator();
```


Esto crea un objeto llamado \$ val. Como puede ver, se parece a una simple variable de PHP, Entonces, ¿qué es un objeto? En este caso particular, es principalmente un dispositivo que le da acceso a todos los métodos definidos en la clase Pos_Validator. Sin \$ val, que no tienen manera de utilizarlos.

Echemos un rápido vistazo a un ejemplo de la clase Pos_Validator en acción. Como he dicho antes, es necesario crear una instancia de la clase para poder usarla. Esto se conoce como instancia de un objeto. De la misma manera como usted pasa argumentos a las funciones, también puede pasar argumentos a un objeto en el momento de la creación de instancias. La clase Pos_Validator acepta dos argumentos, ambos opcionales. Los argumentos opcionales primero son una serie de campos de formulario requerido. Así, la secuencia de comandos para crear instancias del objeto Pos_Validator para validar un formulario sencillo podría tener este aspecto:

```
// Crear una matriz de campos de formulario requerido
$required = array('edad', 'nombre', 'comentarios');

// Instancia del objeto de validación, y pasarle el array $required
$val = new Pos_Validator($required);
```

La manera de acceder a las propiedades de un objeto y métodos con el operador -> (un guión seguido por un signo mayor de-muestra, sin espacio entre ellas). Incluso si usted no sabe nada sobre programación orientada a objetos, no debe tomarle mucho tiempo saber lo que el código hace (tratar de adivinar, y después lee el siguiente párrafo para ver si estaban en lo cierto):

```
// Utilizar métodos de la clase para validar campos individuales
$val->isInt('edad');
$val->removeTags('nombre', 0, 0, 1);
$val->checkTextLength('comentarios', 5, 500);
$val->removeTags('comentarios', 0, 0, 1);
$val->isEmail('email');

// validar la entrada y recibe los mensajes de error
$filtered = $val->validateInput();
$missing = $val->getMissing();
$errors = $val->getErrors();
```

El objeto \$ val comienza comprobando si la edad es un número entero. A continuación, elimina las etiquetas HTML el campo de nombre, comprueba que el campo de comentarios contiene entre 5 y 500 caracteres, y tiras de todas las etiquetas de él antes de comprobar que el campo de correo electrónico contiene una dirección de correo electrónico bien escrita. Las tres últimas líneas son validar la entrada, y obtener los nombres de los desaparecidos campos y detalles de los errores.

Otra ventaja es que los objetos son independientes unos de otros, aunque sean casos de la misma clase. Se pueden crear dos instancias independientes de la clase Pos_Validator para validar la entrada del usuario, tanto las matrices \$_POST y \$_GET. Debido a que los objetos están separados, usted puede identificar en un mensaje de error que ha venido y tomar las medidas oportunas. Cada objeto se comporta como una caja negra, manteniendo los datos pasados a cada uno totalmente independiente de la otra. La analogía caja negra también se aplica a uno de los principales conceptos detrás de Programación orientada a objetos: **encapsulación**.

4.8.2 Protección de la integridad de datos con la encapsulación

La idea de la encapsulación es asegurar que cada parte de una solicitud es independiente y no interfiere con ningún otro, excepto de una manera claramente definida. La programación orientada a objetos se usa hasta en tareas complejas en los componentes de sus partes, por lo que es necesario garantizar que el cambio del valor de una propiedad no desencadene una cadena de efectos no deseados a través de otras partes de la aplicación. Al definir una propiedad en una clase, se debe especificar si es público, privado o protegido. Las propiedades públicas deben ser accesibles a todas las partes de un script PHP, tanto dentro y fuera de la definición de la clase, y sus valores pueden ser cambiados en la misma forma que cualquier variable. Las propiedades privadas y protegidas, por el contrario, ocultan las secuencias de comandos externas, por lo que no se pueden cambiar arbitrariamente.

Los métodos también pueden ser públicos, protegidos o privados. Dado que los métodos permiten a los objetos hacer cosas, como validar los datos proporcionados, con frecuencia ellos deben ser públicos. Sin embargo, los métodos protegidos y privados son útiles para ocultar el funcionamiento interno de una clase por parte del usuario final.

```
protected $_inputType;
```

El valor de `$_inputType` se establece internamente por la clase en el momento de crear instancias del objeto. Sin intentar cambiarlo directamente, PHP genera un error fatal, con lo que todo se para. Aunque esto pueda parecer un inconveniente, esto preserva la integridad de su código para evitar que un atacante engañe a una rutina de validación para manejar las variables con el tipo equivocado de la fuente. Mientras una clase esté bien diseñada, la encapsulación evita los valores de las propiedades importantes de ser cambiadas excepto siguiendo las normas establecidas por la clase.

La encapsulación es una gran ventaja para el usuario final, supone una importante responsabilidad en el promotor para asegurar que los cambios internos en el código no producen cambios inesperados en la producción. Si un método se espera que regrese una cadena, no de repente devolver una matriz. La caja negra debe trabajar de forma coherente. De lo contrario, todo el código dependiente podría verse afectado, derrotando el propósito entero de programación orientada a objetos.

4.8.3 Polimorfismo

A pesar del sonido oscuro de su nombre, el polimorfismo es un concepto relativamente simple. Se aplica tanto a los métodos, las propiedades y a los medios con el mismo nombre en diferentes contextos. Si eso no lo hace más claro, un ejemplo del mundo real debe ayudar. La palabra "polimorfismo" proviene de dos palabras griegas que significan "muchas formas". La cabeza de un ser humano tiene una forma muy diferente a la cabeza de un caballo, pero su función es básicamente la misma (comer, respirar, ver, y así sucesivamente), así que usamos la misma palabra sin confusión. Programación Orientada a Objetos se aplica a la programación por lo que le permite dar el mismo nombre a los métodos y propiedades que juegan un papel similar en las diferentes clases.

En el contexto de un sitio web, puede haber diferentes clases para interactuar con bases de datos en MySQL y SQLite. Aunque el código necesario para conectar a cada base de datos y ejecutar consultas es diferente, los conceptos de conexión y ejecutar consultas son comunes a ambos, por lo que tiene sentido dar las dos clases de métodos, `connect ()` y `query ()` y una propiedad `_result $`. Un Objeto MySQL usará automáticamente el código encapsulado en su caja negra, y SQLite un objeto que haga lo mismo. Sin embargo, gracias al polimorfismo, tanto en clases de uso de métodos y propiedades con nombres comunes.

Contraste esto con la necesidad en la programación de procedimiento a utilizar las diferentes funciones, tales como `MySQL_connect ()` y `sqlite_open ()`. Si desea cambiar la base de datos de su sitio web, es necesario cambiar todas las líneas de código de base de datos. Con el enfoque orientado a objetos, los únicos cambios que usted necesita hacer son los detalles de la conexión y la creación de instancias para un objeto MySQL en lugar de un objeto de SQLite, o viceversa. Mientras su SQL es la base de datos neutral, el resto del código debería funcionar sin problemas.

4.8.4 Herencia

Dado que una clase es simplemente una colección de funciones y variables relacionadas, una forma de añadir nueva funcionalidad es modificar el código. En las primeras etapas de desarrollo, es por lo general el enfoque correcto, el objetivo fundamental de la programación orientada a objetos es la reutilización y la fiabilidad. Así como usted ha heredado algunas características de sus padres, y desarrolló nuevas, las suyas propias, una clase hija o subclase de programación orientada a objetos pueden heredar todas las características de los padres (o superclase), adaptar algunas de ellas, y añadir otros nuevos. Mientras que los seres humanos tienen dos padres, en PHP una clase hija sólo puede tener uno padre. (Algunos lenguajes orientado a objetos, como Python y C ++, permiten a una clase herencia de más de un padre, pero PHP sólo admite herencia simple.)

Elegir las herramientas adecuadas para trabajar con PHP:



El código PHP se escribe y se almacena en el servidor Web en texto sin formato. El servidor web compila Scripts PHP en código de bytes en tiempo de ejecución, por lo que no hay necesidad de ninguna herramienta especial o un compilador. Todo lo que necesitas es un editor de texto para escribir los scripts PHP y un servidor habilitado para ejecutarlos. Puesto que usted necesita estar familiarizado con lo básico de PHP, se supone que ya tienen acceso a un servidor web. Aunque se puede probar el código en un servidor web remoto, que encontrarás en un entorno de prueba local es mucho más eficiente.

Si no desea configurar un entorno de prueba, te sugiero que intente XAMPP para Linux o Windows ([www.apachefriends.org / es / xampp.html](http://www.apachefriends.org/es/xampp.html)) o MAMP para Mac OS X ([www.mamp.info / es / mamp.html](http://www.mamp.info/es/mamp.html)). Ambos son fáciles de instalar y tiene una buena reputación. Alternativamente, es posible que desee invertir en un entorno de desarrollo integrado de PHP especializados (IDE), como Zend Studio for Eclipse (www.zend.com/en/products/studio/) o PhpED ([www.nusphere.com / productos / phped.htm](http://www.nusphere.com/productos/phped.htm)). Ambos se han incorporado en servidores de PHP para las pruebas. Ver anexo Instalaciones.

4.8.5 Uso de PHP Zend Framework Codificación estándar

He elegido este conjunto particular de convenciones de codificación, ya que viene de la fuente. Zend es la empresa fundada por los creadores del motor de la base de PHP, Zeev Suraski y Andi Gutmans, que siguen siendo principales contribuyentes a PHP. El estándar fue desarrollado para el Zend Framework, una amplia biblioteca de clases avanzadas de PHP que usted debe estudiar después de terminar de estudiar este capítulo. El énfasis está en hacer que su código sea fácil de leer y entender. Los detalles completos están en <http://framework.zend.com/manual/en/codificación-standard.html>, así que voy a ir sólo a través de los puntos principales.

Convenciones de nombres:

Utilice nombres de clase que se asignan a la estructura de directorios en los que se almacena la clase anteponiendo el nombre de clase con el nombre de cada directorio seguido de un subrayado. Todas las clases en este capítulo están en el directorio Pos, por lo que la clase en Validator.php se llama Pos_Validator.

Mayúscula sólo la primera letra de cada palabra en los nombres de archivos y directorios.

Utilice nombres descriptivos para los métodos y propiedades. Deben ser lo más detallado como sea posible para aumentar la comprensibilidad. Los nombres deben comenzar con una minúscula, pero cuando estén constituidos por más de una palabra, cada posteriores la palabra debe comenzar con una letra mayúscula (caso de camello).

Empiece los nombres de las propiedades privadas y protegidas con un guión bajo (privada y bienes protegidos).

Utilice mayúsculas sólo para las constantes, y separar cada palabra con un subrayado.

Estilo de codificación:

Utilice siempre la apertura total de etiquetas PHP (<? php).

Omitir el cierre de etiquetas PHP (??) en los archivos que contienen sólo código PHP. El cierre de etiquetas PHP es opcional, siempre y cuando nada más (por ejemplo, HTML) viene después del código PHP. Dejando de lado la etiqueta de cierre tiene la ventaja de evitar los espacios en blanco no deseados provocando la "headers already sent" error cuando se utiliza incluye (<http://docs.php.net/manual/en/language.basic-syntax.instructionseparation.php>).

Guión del código de cuatro espacios (que han seguido este convenio en los archivos de descarga, pero yo sólo lo he utilizado dos espacios debido a las restricciones de la página impresa).

Siempre que sea posible, limitar las líneas hasta un máximo de 80 caracteres. El impreso de la página me permite sólo 72, por lo que he utilizado una flecha como ésta ➡ para indicar el código escrito en la misma línea.

Siempre use comillas simples para las cadenas, a menos que contengan las variables que son procesadas u otros signos de cotización.

Use una combinación de comillas simples y dobles en lugar de escapes.

Ponga un espacio a ambos lados del operador de concatenación (.) para mejorar la legibilidad.

Inserte un espacio al final después de las comas para mejorar la legibilidad.

Quiebren matrices asociativas en varias líneas, y el uso de los espacios en blanco de relleno para alinear las claves y valores.

Ponga las llaves de apertura y cierre de clases y funciones en líneas separadas.

Ponga la llave de apertura de una sentencia condicional en la misma línea como la condición, pero la llave de cierre en una línea propia.

Utilice el formato PHPDoc comentarios.

4.8.6 Creación de clases y objetos

PHP tiene muchas clases incorporadas, algunas de las cuales vamos a utilizar más adelante en el capítulo, como DateTime, XmlWriter y XmlReader. Sin embargo, el enfoque en esta parte es la construcción de sus propias clases. Una vez que una clase se ha definido, se utiliza exactamente de la misma manera que cualquier de las incorporadas en PHP.

Una clase normalmente contiene propiedades y métodos. No hay reglas sobre la orden en que deben aparecer dentro de la clase, pero la convención normal es declarar todas las propiedades en primer lugar, seguido de los métodos. A lo largo del documento, voy a utilizar ejemplos basados en una aplicación imaginaria de comercio electrónico que vende una pequeña gama de productos.

Así, así es como se define una clase llamada producto:

```
class Producto
{
    // propiedades definidas
    // métodos definidos
}
```

Eso es en realidad una clase válida. No hace nada, ya que contiene sólo los comentarios, pero es perfectamente válido.

No hay reglas en donde debe definir las clases, pero es considerada la mejor práctica para definir cada clase en un archivo propio, ya que hace más fácil volver a implementar las clases en diferentes proyectos. A diferencia de algunos lenguajes, tales como ActionScript, no hay ninguna restricción en llamar al archivo, pero hace la vida más fácil si se utiliza el mismo nombre que la clase. Así, el Producto clase debe ser definida en product.php. Ya que el producto es probable un nombre de clase común, se recomienda para darle un prefijo de tres o cuatro caracteres para evitar conflictos de nombres.

La clase no sirve de nada sin ningún tipo de propiedades o métodos. Estas son las mismas variables y funciones, pero antes de añadirlos, es necesario comprender modificadores de visibilidad.

Como expliqué anteriormente, la encapsulación es un concepto clave en la programación orientada a objetos, por lo que se necesita para controlar la visibilidad de las propiedades de una clase y los métodos. La visibilidad determina si una propiedad o método se puede acceder directamente a cualquier parte de una secuencia de comandos que utiliza una clase, o si sigue siendo la residencia de la clase. Este es el principio de la caja negra. Para mantener la fiabilidad motor de un coche bien afinado, un mecánico no requiere, jugar con el calendario de los tapones. Igualmente, si una clase es la de producir resultados fiables, ciertas partes de su funcionamiento interno deben ser escondidos de la

vista. Para ello, antes de la propiedad o método de definición con una de las siguientes palabras: público, protegido, o privado. Tabla 4.8.6.1 explica el significado de cada palabra clave.

Modificador	Significado
Público	Esto significa que la propiedad o el método se puede acceder por cualquier parte de una secuencia de comandos dentro y fuera de la definición de la clase. Todos los métodos son considerados como públicos, a menos que vaya precedido por un modificador diferente.
Protegido	Esto impide el acceso externo a una propiedad o método, pero permite el acceso a nivel interno como a los padres y las clases hijas.
Privado	Propiedades privadas y los métodos privados sólo se puede acceder dentro de la clase donde se definen.

Tabla 4.8.6.1 PHP modificadores y visibilidad (Control de acceso).

Aunque se puede omitir el modificador de visibilidad de un método, en cuyo caso se pondrá por defecto a público, no se considera una buena práctica.

Experimentando con la visibilidad:

Este ejercicio muestra lo que sucede cuando se intenta acceder a una propiedad protegida fuera de la definición de clase y se muestra cómo crear métodos getter y setter.

1. Crear un sitio de PHP en la raíz de su servidor. El nombre no es importante, pero yo lo he llamado Soluciones.
2. Crear una carpeta llamada C2, y crear un archivo llamado product.php en la carpeta C2.
3. Inserte el siguiente código en product.php:

```
<?php
class Ch2_product
{
    // Las propiedades se definen aquí
    protected $_type = 'Libro';
    // Los métodos se definen aquí
}
```

Esto define la clase Ch2_product con una propiedad protegida llamada \$_type. Para mayor comodidad, he dado a la propiedad \$_type un valor por defecto, pero el establecimiento de un valor por defecto no es necesario. El valor de una propiedad se puede establecer de maneras diferentes.

Como se señaló anteriormente, el Zend Framework de PHP codificación estándar omite el cierre de etiquetas PHP para evitar problemas con los espacios en blanco no deseados. En los códigos siguientes, también omitiremos la etiqueta de apertura para ahorrar espacio, pero siempre se debe usar, ya que sólo la etiqueta de cierre es opcional.

4. En la raíz del sitio, cree una carpeta llamada ch2_exercises. Dentro de la carpeta nueva, crear un archivo llamado product_test.php, e introduzca el siguiente código:

```
// incluye el archivo de clase
require_once '../Ch2/Product.php';
```

Para utilizar una clase, se debe incluir en su guión, pero por el momento he utilizado require_once. Al incluir un archivo de clase de esta manera, el motor de PHP define la clase lista para uso. Al igual que las funciones, las clases se pueden definir sólo una vez en una secuencia de comandos, así que es una buena idea usar require_once, para evitar desencadenar un error fatal.

5. A continuación, es necesario crear una instancia de una clase. Agregue el código siguiente para `product_test.php`:

```
// Crear una instancia de la clase Ch2_Product
$product = new Ch2_Product();
```

La sintaxis es muy simple. Puede utilizar la nueva palabra clave en frente del nombre de clase y luego un par de paréntesis vacíos después del nombre de la clase. Los paréntesis no son estrictamente necesarios en este caso, pero es una buena práctica usarlos. Como se verá más adelante, puede pasar argumentos a una clase de la misma manera que con una función. La instancia de la clase se almacena en una variable ordinaria, en este caso `$_Producto`. `$_Producto` es ahora un objeto.

6. Vamos a tratar de mostrar el valor de la propiedad del objeto `$_type`. Para acceder a un objeto de propiedades o métodos, se utiliza el operador `->` (un guión seguido del signo mayor-que sin espacio entre ellos).

Agregue el código siguiente para `product_test.php`:

```
// Mostrar la propiedad $_type
echo $product->_type;
```

Cuando accede a las propiedades con el operador `->`, se le cae el signo de dólar (\$) por lo que al inicio del nombre de la propiedad con `$product ->` se refiere a la propiedad `$_type` del objeto `$product`.

7. Guarde ambos archivos, y cargue `product_test.php` en su navegador. Usted debe ver un resultado similar a la Ilustración 4.8.6.2.

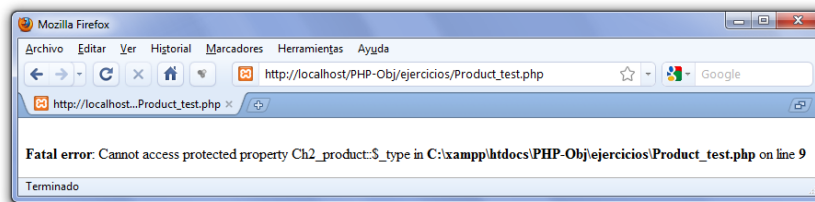


Ilustración 4.8.6.1 El intento de acceder a una propiedad protegida resultados en un error fatal.

Esta es la visibilidad y la encapsulación en el trabajo. La propiedad `$_type` está tan bien protegida por lo que al intento de acceso se dispara un error fatal.

8. Volver a `product_test.php`, y tratar de cambiar el valor de `$_type` como éste:

```
$product->_type = 'DVD';
```

El resultado es el mismo: un error fatal. Usted no puede presentar por pantalla `$_type` con `echo` ni cambiar su valor. La propiedad está protegida en el interior de la clase y no puede ser accedida por un script externo.

9. Para acceder a una propiedad protegida o privada, es necesario crear los métodos `getter` y `setter` dentro del archivo de la clase. Ambos son muy simples, y el uso de una variable especial llamada `$This`, que hace referencia al objeto actual. Agregue el código siguiente a la clase `Ch2_Product`:

```
// Métodos definidos aquí
public function getProductType()
{
    return $this->_type;
}
public function setProductType($type)
{
    $this->_type = $type;
}
```

Ambos métodos tienen que ser accesible desde fuera de la clase, por lo que su visibilidad se establece en público. Aunque la programación orientada a objetos se refiere a ellos como los métodos, que son, de hecho, las funciones y se utiliza la palabra clave en función de la misma manera como en el código de procedimiento. Tenga en cuenta que he utilizado nombres detallados y descriptivos. Esto no sólo hace que sea más fácil entender lo que hacen, que tenía que evitar el uso de `getType()`, que es el nombre de una función integrada de PHP.

La función `getter`, `getProductType()`, devuelve `$this->_type`, es decir, el `$_type` la propiedad del objeto actual.

La función `setter`, `setProductType()`, toma un único argumento, \$ tipo, y asigna su valor para `$this->_type`. Observe el uso de `$type` (sin un guión bajo). Es común la práctica de dar un argumento del mismo nombre menos el guión-como la propiedad a la que se asigna su valor. Esto hace el código fácil de entender, pero tenga cuidado de no mezclar los dos.

10. Para mostrar el valor de `$_type`, cambiar el código en el paso 6 por esto:

```
echo $product->getProductType();
```

Si ejecuta `product_test.php` ahora, debería ver libro que aparece en pantalla.

11. Para cambiar el valor de `$_type`, modificar el código en el paso anterior como éste:

```
$product->setProductType('DVD');  
echo $product->getProductType();
```

Ejecute el script de nuevo, y usted debería ver en la pantalla de DVD. Así, a pesar de los `$_type` se protege la propiedad, puede cambiar su valor por defecto del libro mediante el uso de `setProductType()`.

Pasando Valores a un constructor.

Este ejercicio se basa en el anterior mediante la adición de un constructor que toma dos argumentos: uno para establecer el tipo de producto y el otro para establecer su título.

1. Dado que el tipo de producto se ajustará al pasar un argumento al constructor, necesita eliminar el valor predeterminado de `$_type` (Libro). También crear una nueva propiedad protegida denominada `$_title`. Las propiedades en la parte superior de la clase `Ch2_Product` deben tener ahora el siguiente aspecto:

```
// Propiedades que se definen aquí  
protected $_type;  
protected $_title;
```

2. El constructor debe tener dos argumentos, uno para cada una de las propiedades, y asignarlos a las propiedades del objeto protegido. Agregue el código siguiente inmediatamente después de las propiedades (y recordar que hay dos guiones antes del "constructor"):

```
// constructor  
public function __construct($type, $title)  
{  
    $this->_type = $type;  
    $this->_title = $title;  
}
```

3. Dado que el valor de `$_type` se establece por el constructor, ya no necesita el método `setter`. Por lo tanto, eliminar y agregar un método `getter` para recuperar el valor de la propiedad `$_title`. La clase modificada deberá ser de esta manera:


```
<?php
class Ch2_product
{
    // propiedades que se definen aquí
    protected $_type;
    protected $_title;

    // constructor
    public function __construct($type, $title)
    {
        $this->_type = $type;
        $this->_title = $title;
    }

    // métodos definidos aquí
    public function getProductType()
    {
        return $this->_type;
    }
    public function getTitle()
    {
        return $this->_title;
    }
}
?>
```

4. Usted instancia el objeto de la misma manera que antes, usando la palabra clave new. Sin embargo, el constructor de la clase ahora espera dos argumentos. Para mantener el código sencillo, no he añadido ningún código para comprobar el tipo de datos que se pasan como argumentos, pero vamos a suponer que los dos están destinados a ser cadenas. Prueba la clase revisando lo que se ha modificado al código en product_test.php de la siguiente manera:

```
$product1 = new Ch2_Product('Libro', 'PHP Object-Oriented Solutions');
$product2 = new Ch2_Product('DVD', 'Atonement');
echo '<p>$product1 es un ' . $product1->getProductType();
echo ' llamado ' . $product1->getTitle() . "'</p>';
echo '<p>$product2 es un ' . $product2->getProductType();
echo ' llamado ' . $product2->getTitle() . "'</p>';
```

Cuando se ejecuta product_test.php, debería ver algo similar a la Ilustración 4.8.6.3.

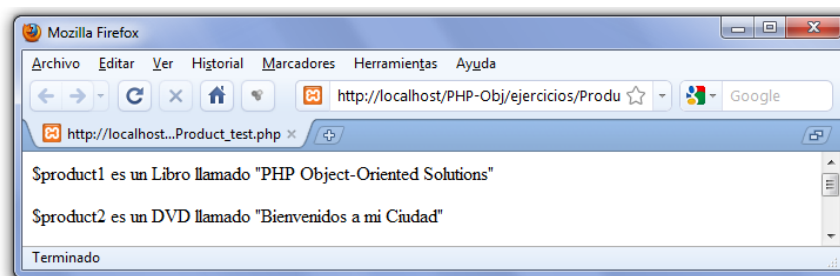


Ilustración 4.8.6.2 Los objetos tienen ahora los diferentes tipos de productos y los títulos asignados por el constructor.

Es cierto que este no es el resultado más emocionante del mundo, pero es el objetivo en esta etapa para entender la sintaxis básica de PHP OOP sin introducir ningún tipo de distracciones.

El código en el paso final del ejercicio, será creando dos objetos independientes: uno un libro, el otro un DVD. A primera vista, esto podría no parecer diferente de la creación de dos matrices asociativas de la siguiente manera:

```
$product1 = array('type' => 'Book',  
                 'title' => 'PHP Object-Oriented Solutions');  
$product2 = array('type' => 'DVD',  
                 'title' => 'Atonement');
```

Sin embargo, las diferencias son enormes:

Los valores de un elemento de la matriz se puede cambiar en cualquier momento. Protegidas y privadas, las propiedades de un objeto no se puede cambiar a menos que la clase proporcione un método para hacerlo.

Usted no tiene control sobre el tipo de datos almacenados en una matriz, pero en una cadena puede hacer cambios arbitrariamente a un número o un recurso de base de datos. El código dentro de un constructor de la clase o el método setter puede rechazar cualquier dato que no se ajuste a sus especificaciones (por ejemplo, podría utilizar `is_string()` para asegurarse si valor es una cadena antes de asignar a la propiedad).

A diferencia de los elementos de la matriz, los objetos pueden tener métodos capaces de procesar datos. Hasta ahora, los únicos métodos que hemos visto son muy simples, pero veremos cada vez más su complejidad a medida que avancemos en el estudio.

El ejercicio anterior utiliza el constructor para asignar valores pasados desde el exterior a las propiedades del objeto, pero se puede utilizar el constructor para hacer una programación inicial para todos los objetos de la misma clase. Así, por ejemplo, si todos los productos son hechos por el mismo fabricante, podría utilizar el constructor para asignar el valor predeterminado a una propiedad protegida llamada `$ _manufacturer`. Si desea que todos los objetos que comparten el mismo valor, no hay necesidad de pasarle al constructor como argumento.

El uso de un constructor ha hecho la clase `Ch2_Product` más flexible, pero introduce también un problema. Aunque es posible crear objetos para representar los diferentes tipos de productos, libros y DVD's tienen características diferentes. Los libros tienen páginas, pero los DVD no. Los DVD tienen una duración o el tiempo de juego, pero los libros no lo tienen. Usted puede evitar esto mediante la creación de una propiedad llamada `$ _length` y el uso de sentencias condicionales para determinar si esto se refiere a los números de páginas o al tiempo de juego, dependiendo del tipo de producto. Se podría, pero es desordenado. Una solución mucho mejor es utilizar la herencia y crear subclases especializadas.

Creando clases hijas para los libros y DVD's.

En este ejercicio se crean dos clases hijas de `Ch2_Product` y muestra lo que sucede cuando añadimos una nueva propiedad a una clase hija. Continuar trabajando con `product.php` y `product_test.php` del ejercicio anterior.

1. Crear un nuevo archivo llamado `Libro.php` en la carpeta `Ch2`, e introduzca el siguiente código:

```
<?php  
require_once'Product.php';  
class Ch2_Libro extends Ch2_Product  
{  
    //La definición de la clase va aquí  
}  
?>
```

Esto incluye la clase padre con `require_once` y utiliza la palabra clave `extends` para indicar que la clase `Ch2_Libro` es un hijo de `Ch2_Product`. No es necesario añadir nada entre las llaves. Tal como está, esta es una definición completa de la nueva clase, que hereda todas las propiedades y métodos de su padre.

2. Crear un nuevo archivo llamado DVD.php en la carpeta C2, e introduzca el siguiente código:

```
<?php

require_once'Product.php';
class Ch2_DVD extends Ch2_Product
{
    //La definición de la clase va aquí
}

?>
```

3. Modificar el código en product_test.php así:

```
<?php

//Para las clases hijas Libro y DVD
require_once '../Ch2/Libro.php';
require_once '../Ch2/DVD.php';
$producto1 = new Ch2_Libro('Libro', 'PHP Object-Oriented Solutions');
$producto2 = new Ch2_DVD('DVD', 'Atonement');
echo '<p>$producto1 es un ' . $producto1->getProductType();
echo ' llamado ' . $producto1->getTitle() . "</p>";
echo '<p>$producto2 es un ' . $producto2->getProductType();
echo ' llamado ' . $producto2->getTitle() . "</p>";

?>
```

En lugar de incluir Ch2_Producto, esto incluye los dos archivos de las clases hijas y crea cada una de las clases Ch2_Libro y Ch2_DVD.

4. Ejecute Product_test.php. La salida debe ser exactamente la misma que antes, es decir, la salida debe ser igual a la ilustración siguiente.

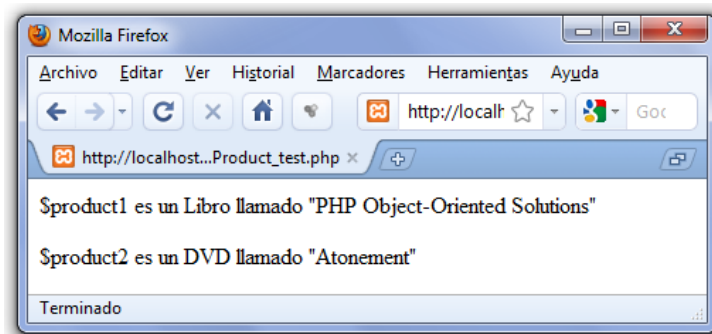


Ilustración 4.8.6.3 Utilizando clases hijas.

5. Por el momento, las clases hijas son idénticas a sus clases padre. Así que vamos a añadir una nueva propiedad denominada \$ _pageCount a la clase Ch2_Libro que establezca su valor al pasar un tercer argumento al constructor. Hacer la nueva propiedad protegida, y añadir un método getter para recuperar su valor:

```
class Ch2_Book extends Ch2_Product
{
    protected $ _pageCount;
```

```
public function __construct($type, $title, $pageCount)
{
    $this->_pageCount = $pageCount;
}
public function getPageCount()
{
    return $this->_pageCount;
}
}
```

6. Para comprobar que todo funciona como se esperaba, modificar `product_test.php` así:

```
<?php

require_once '../Ch2/Libro.php';
$libro = new Ch2_Libro('Libro', 'PHP Object-Oriented Solutions', 300);
echo '<p>' . $libro->getTitle() . ' tiene ' . $libro->getPageCount() . ' páginas'</p>';

?>
```

Al ejecutar la secuencia de comandos, debería ver el resultado que se muestra en la ilustración siguiente.

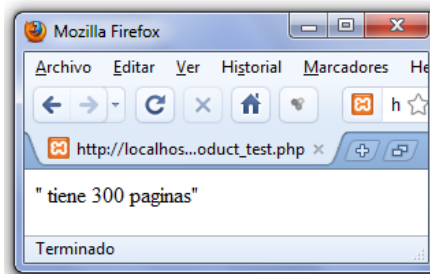


Ilustración 4.8.6.4 Utilizando la propiedad `$_pageCount`.

La secuencia de comandos no genera ninguna advertencia, por lo que no hay errores de sintaxis. El método `getTitle()` de la clase padre ha sido heredada por la clase hija, por lo que es legítimo usarlo. Sin embargo, no se ha producido ninguna salida. "Por supuesto que no tiene porque no se ha asignado el valor de `$_title` a la propiedad del objeto `$_title`".

Una clase hija hereda todas las propiedades de su padre y sus métodos. Por lo que también estaría justificado en preguntarse por qué no utilizar el constructor `Ch2_Producto`. Después de todo, lo hizo cuando probó las clases hija en el paso 4. La respuesta es que se trata de polimorfismo en acción: una clase hija hereda los métodos de su padre y propiedades, pero también puede definirlos o anularlos. Ya no estás creando un objeto `Ch2_Producto`, pero si un objeto `Ch2_Libro`.

Al definir un constructor en la clase hija, lo ha invalidado el constructor de los padres. La clase hija hace sólo lo que su propio constructor le dice que haga. Acepta tres argumentos pero no hace nada con los dos primeros (`$_type`, y `$_title`). Para asignar el valor de estos argumentos a las propiedades `$_type` y `$_title`, usted tiene que hacerlo de forma explícita en el constructor de la clase hija o encontrar alguna manera de invocar el constructor de los padres.

4.8.6.1 Acceso a los métodos y propiedades de una clase padre

La posibilidad de reemplazar los métodos de una clase principal o propiedades es muy conveniente, pero no son momentos en los que quieres acceder a ellos en una clase hija. Para ello, es necesario utilizar el operador de resolución de ámbito.

El operador de resolución de alcance es un par de dos puntos (::). El nombre de la clase va al lado izquierdo del operador, y el nombre del método o la propiedad va a la derecha de esta manera:

```
ClassName::methodOrPropertyName
```

PHP no se opone si pones un espacio en blanco a ambos lados del operador de resolución de ámbito, pero la convención normal es escribir todo como una sola entidad.

El operador de resolución de alcance tiene los siguientes usos:

Da acceso a propiedades o métodos de una clase padre.

Se utiliza para llamar a los métodos estáticos y propiedades de una clase.

Da acceso a las constantes de clase.

Es la forma convencional de hacer referencia a un método de clase o de los bienes en la documentación. Por ejemplo, en lugar de referirse a "la getPageCount () de la Clase Ch2_Libro, "esto se reduce a Ch2_Libro::getPageCount().

Debido a que Ch2_Libro reemplaza el constructor Ch2_Product, usted necesita llamar al constructor del padre dentro del constructor del hijo. Uso de la sintaxis ha señalado anteriormente, puede llamar al constructor Ch2_Product así:

```
Ch2_Product :: _Construct($type, $title);
```

Sin embargo, PHP dispone de dos palabras clave para el trabajo práctico con el operador de resolución de ámbito, a saber:

- **Parent:** Esto se refiere a los padres o de cualquier antepasado de la clase actual.
- **Self:** Esto se refiere a la clase actual. Aunque esto suene lo mismo que \$this, que se reunió antes, sí se refiere a la clase en general, mientras que \$ this hace referencia a los la instancia actual de la clase (es decir, un objeto creado a partir de la clase).

Así, para llamar al constructor de los padres de la clase Ch2_Libro, puede modificar el código anterior de esta manera:

```
parent::__construct($type, $title);
```

La ventaja de las palabras clave parent y self, es que no es necesario cambiarlos, si cambia el nombre de cualquiera de las categorías a que se refieren. El siguiente ejercicio muestra cómo llamar al constructor padre en un método reemplazado.

Llamando al constructor de los padres:

Este breve ejercicio modifica el código en del ejercicio anterior para llamar al constructor padre Cuando se reemplaza en una clase hija. Se debe continuar trabajando con los mismos archivos.

1. Modificar el constructor en Libro.php a este aspecto:

```
<?php
    public function __construct($type, $title, $pageCount)
    {
        parent::__construct($type, $title);
        $this->_pageCount = $pageCount;
    }
?>
```

Esto llama al constructor de la clase dominante (Ch2_Product) y le pasa \$type y \$title como argumentos. El método constructor padre asigna estos valores a la _type \$ y \$ propiedades _title, respectivamente. Dado que ambas propiedades se heredan, se convierten en propiedades de la clase hija. El valor de la propiedad \$_pageCount se asigna de la misma manera que antes.

2. Ejecutar `Product_test.php`. Esta vez debería ver el resultado de la muestra en la Ilustración. (La única diferencia es que llama a la versión actualizada de la clase `Ch2_Libro`.)

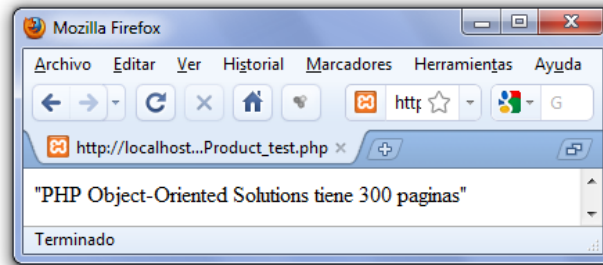


Ilustración 4.8.6.1.1 El título del libro aparece ahora desplegado correctamente en el navegador.

No me molestaré en hacer los mismos cambios en la clase `Ch2_DVD`, porque es probable que se haya dado cuenta que no tiene sentido establecer la propiedad `$_type` en el constructor cuando tiene clases separadas para los distintos tipos de productos. Sin embargo, se trataba de una forma práctica de demostrar el uso de los padres con el operador de resolución de ámbito.

Al establecer la propiedad `$_type` en el constructor no sólo es redundante, sino que se rompe el principio de encapsulación, al permitir un valor arbitrario asignado a una propiedad que debe fijarse. Un libro es un libro es un libro. . . En lugar de llamar al constructor padre para establecer su valor, voy a codificar la propiedad `$_type` en el constructor del hijo. Como podrá ver más adelante en este capítulo, el constructor de los padres ya no es necesario. Para aquellos de ustedes que ya saben los fundamentos de la programación orientada a objetos, tengo la intención de convertir la clase `Ch2_Product` en una clase abstracta.

Creando constructores separados para los libros y DVD's.

Cuando añadió originalmente la propiedad `$_pageCount` a `Ch2_Libro`, el constructor del hijo hizo caso omiso de uno de los padres sin establecer la propiedad `$_type` y `$_title`. Ahora vamos a darle ese derecho. Continuar trabajando con los mismos archivos.

1. La propiedad `$_type` de un objeto `Ch2_Libro` será siempre de libros, así que usted puede codificar el valor en el constructor, pero la propiedad `$_title` tiene que ser fijado pasar un argumento al constructor. Esto significa que usted necesita pasar sólo dos argumentos para el constructor `Ch2_Libro` (para el título y el número de páginas). Modifíquese la clase como ésta:

```
public function __construct($title, $pageCount)
{
    $this->_title = $title;
    $this->_pageCount = $pageCount;
    $this->_type = 'libro';
}
```

Observe que las propiedades `$_title` y `$_type` no se declaran en la parte superior de la clase hija, sin embargo, se hace referencia en el constructor como `$this->_title` y `$this->_type`. Esto se debe a que la clase hijo hereda las propiedades de las clases padre, y los trata como propio.

2. Al crear un objeto `Ch2_Libro`, que ahora pasa sólo dos argumentos: `title` y la `pageCount`. Modificar el código en `product_test.php` como este:

```
<?php
require_once '../Ch2/Libro.php';
$libro = new Ch2_Libro('PHP Object-Oriented Solutions', 300);
echo '<p>' . $libro->getProductType() . ' ' . $libro->getTitle() . ' ' . $libro->getPageCount() . ' páginas</p>';
?>
```

El código adicional en la orden de echo utiliza getProductType \$libro-> () para mostrar el valor de \$ _type.

3. Ejecutar Product_test.php. Usted debe ver el resultado se muestra en la ilustración.

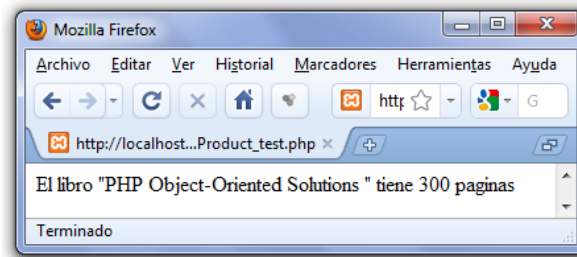


Ilustración 4.8.6.1.2 El tipo de producto fue agregado a la salida del navegador.

4. En lugar de un recuento de páginas, la clase Ch2_DVD necesita una propiedad para guardar el tiempo de juego y un método getter para recuperar su valor. El código es muy similar a la versión revisada de Ch2_Libro, por lo que no requiere ninguna explicación. La lista completa es así:

El método constructor en cada una de estas clases reescrito en las hijas hace todo lo que necesita, por lo que ya no utiliza la palabra clave de los padres para llamar al constructor de los padres. Decidir si para llamar al constructor de los padres depende totalmente en el diseño de la clase hija. Digamos que tiene una clase con dos propiedades, \$ _a y \$ _b, que son fijados para pasar dos argumentos al constructor de la siguiente manera:

```
class ChildClass extends ParentClass
{
    protected $_c;
    public function __construct($a, $b, $c)
    {
        parent::__construct($a, $b);
        $this->_c = $c;
    }
}
```

En este ejemplo deliberadamente simple, llamar al constructor padre guarda sólo una línea de código, sino una clase real puede tener un constructor que es mucho más larga. Sin embargo, es importante contar no cuánto ahorro al escribir, el principio de la herencia. Cualquier cambio que realice en el constructor de los padres se aplicará automáticamente a la clase hija. Si se reemplaza el constructor de los padres por completo, como lo he hecho en Ch2_Libro y Ch2_DVD, la relación entre padres e hijos es mucho más tenue. Sin embargo, usted verá que yo he hecho esto deliberadamente como un preludio a la conversión Ch2_Product en una clase abstracta. Las clases abstractas son un avanzado concepto de programación orientada a objetos.

Hay momentos en los que no quieren que suceda. La principal razón por la que podría querer evitar que algo sea reemplazado, ya que hacerlo violaría la lógica de la aplicación o tal vez un principio de negocios. Por ejemplo, puede crear un método que utilice una fórmula fija para calcular la comisión obtenida por una persona de ventas. Si permite que el método se invalide, alguien podría crear una clase hija y el doble de la comisión. Agradable para la gente de las ventas, no tan agradable para la integridad de su clase.

Creando clases y métodos abstractos

Si usted trabajó a través de los ejercicios desde el comienzo de este capítulo, es posible que se pregunte si la clase Ch2_Product es realmente necesaria. Como resultado de la herencia, tiene clases especializadas para los libros y DVDs. Si otros productos se añaden a la gama, puede crear nuevas clases hijas y probablemente nunca crear un objeto Ch2_Product en la suya propia. De hecho, para mantener el control, es una buena idea para evitar que los productos genéricos sean creados. Sin embargo, sin la clase padre, sería necesario declarar común las propiedades y métodos en cada clase hija. Al hacer la clase padre resumen, usted obtiene todos los beneficios de la herencia, y evitar que la clase padre sea una instancia.

Para hacer una clase abstracta, agregue la palabra clave `abstract` en frente de la clase y el nombre de la clase en la definición de clase. Así, la clase `Ch2_Product` revisada se parece a esto:

```
abstract class Ch2_Product
{
    // propiedades definidas aquí
    protected $_type;
    protected $_title;
    // métodos definidos aquí
    public function getProductType()
    {
        return $this->_type;
    }
    public function getTitle()
    {
        return $this->_title;
    }
}
```

Las únicas diferencias son la adición de la palabra clave `abstract` en la primera línea y el la eliminación del constructor. Yo podría haber dejado el constructor, si todas las clases hijas se acostumbran a la misma, pero no lo hacen. Todo lo que queda son las propiedades y métodos comunes.

Sin embargo, digamos que usted desea que todas las clases hijas pongan en práctica un método concreto, pero sabemos que los detalles del método serán diferentes para cada clase. En lugar de definir un método de la clase abstracta y fundamental en cada una de las clases hija, se puede definir un método abstracto. Un método abstracto simplemente los nombres de los métodos y establece su visibilidad, pero deja los detalles a la clase hija. En nuestra tienda en línea teóricamente, es posible que desee crear un método para mostrar los detalles de cada producto. Debido a que los detalles de cada tipo de productos son diferentes, la aplicación será diferente.

El siguiente listado muestra la Clase `Ch2_Product` con la adición de un método abstracto llamado `display()`:

```
<?php
    abstract class Ch2_product
    {
        // propiedades que se definen aquí
        protected $_type;
        protected $_title;
        // métodos definidos aquí
        public function getProductType()
        {
            return $this->_type;
        }
        public function getTitle()
        {
            return $this->_title;
        }
        abstract public function display();
    }
?>
```

Declarar un método abstracto como éste obliga a todas las clases hijas a definir los detalles de los métodos. De lo contrario, provoca un error fatal. Así que, en esencia, este dice que todas las clases en la Jerarquía `Ch2_Product` deben tener un método `display()`, pero reconoce que la aplicación será diferente para cada clase. Para la clase `Ch2_Libro`, que podría tener este aspecto:

```
        public function display()
        {
            echo "<p>Libro: $this->_title ($this->_pageCount pages)</p>";
        }
```

Para Ch2_DVD, podría tener este aspecto:

```
public function display()
{
    echo "<p>DVD: $this->_title ($this->_duracion)</p>";
}
```

A continuación, puede crear instancias de ambas clases y el uso de display() para la producción de datos adecuados para cada tipo de objeto en product_test.php.

```
<?php

require_once '../Ch2/Libro.php';
require_once '../Ch2/DVD.php';
$libro = new Ch2_Libro('PHP Object-Oriented Solutions', 300);
$movie = new Ch2_DVD('Atonement', '2 hr 10 min');
$libro->display();
$movie->display();

?>
```

La salida de la pantalla que se muestra en la ilustración:

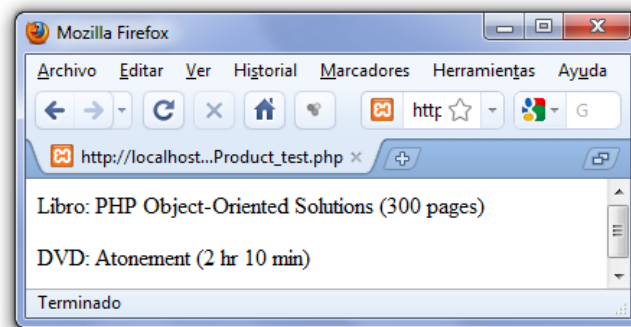


Ilustración 4.8.6.1.3 Para declarar un método abstracto, cada hijo es forzado a implementar su propia versión.

Una clase abstracta no necesita contener métodos abstractos, pero tan pronto como se declara un método abstracto en una clase, toda la clase debe ser declarada abstracta.

Limpiar con un método destructor.

Normalmente, PHP maneja la administración de memoria a través de la recolección de basura-la automática la eliminación de las variables y objetos cuando ya no son necesarios por un guión. Sin embargo, hay ocasiones en que es posible que desee eliminar un recurso, como una base de datos, cuando un objeto ya no es necesario. Así como PHP llama al método `__construct()` cuando se crea un objeto nuevo, busca el método mágico `__destruct()` inmediatamente antes de extraer un objeto de la memoria y se ejecuta el destructor de forma automática si se ha definido.

Un destructor al igual que cualquier otro método se define:

```
public function __destruct()
{
    // código para limpiar los recursos antes de eliminar el objeto.
}
```

Un destructor no puede tener ningún argumento. También es aconsejable evitar las llamadas a otros objetos destructores, ya que no hay manera de imponer el orden en que los objetos se borran, y el otro objeto podría haber sido eliminado en el momento en que el destructor intenta de acceder a ellos.

4.9 Uso de Filtros para validar la entrada de usuarios con PHP

Las funciones de filtro se activan por defecto en la versión de PHP 5.2 y superiores. Puedes verificar que su servidor compatible con ellas ejecutando `phpinfo()` para mostrar los detalles de configuración, a como se muestran en la ilustración 4.9.1. Si de entrada La validación y Filtro aparece como habilitado, ya está listo para usarla.

Si, por cualquier razón, las funciones de filtro no están permitidas, se pueden instalar por separado de la extensión de PHP Biblioteca Comunitaria (PECL). El alojamiento compartido, esto debe ser hecho por el administrador del servidor.

filter

Input Validation and Filtering	enabled	
Revision	\$Revision: 1.52.2.44 \$	

Directive	Local Value	Master Value
filter.default	unsafe_raw	unsafe_raw
filter.default_flags	no value	no value

Ilustración 4.9.1 Confirmación de que las funciones y filtros están habilitadas.

Si usted está administrando su propio servidor, escriba lo siguiente en la línea de comandos de Linux o Unix:

```
pecl install filter
```

En Windows, `php_filter.dll` se descarga de http://pecl4win.php.net/ext.php/php_filter.dll, y guárdelo en la carpeta `ext` con los otros archivos PHP de DLL. Asegúrese de que el archivo coincide con la versión de PHP que se está ejecutando. Después de la instalación desde PECL, reiniciar el servidor web.

No debería haber ninguna necesidad de instalar las funciones de filtro de PECL en PHP 5.2 o superior, ya que son parte del núcleo de PHP instaladas de forma predeterminada.

Las funciones de filtro de PHP adoptan un enfoque poco convencional a la validación. En lugar de funciones separadas para poner a prueba los tipos de datos específicos o formatos, hay sólo siete funciones de filtro, sólo cuatro de los cuales realmente no hacen ningún tipo de filtrado o validación. Usted controla lo que está aceptado o rechazado para pasar una matriz de constantes PHP como argumentos de la función. Hay 50 constantes asociadas a las funciones de filtro, que los hace muy versátiles.

Aunque el propósito de este capítulo es ocultar la complejidad, es necesario comprender cómo funcionan las funciones de filtro con el fin de ser capaces de construir una clase que crea una interfaz más simple. Las siguientes secciones contienen una gran cantidad de material de referencia para orientar a través del laberinto.

Vamos a empezar mirando a las funciones. Tabla 4.9.1 enumera cada función con una breve descripción de su propósito. Los cuatro últimos son los que hacen el filtrado.

Funciones	Descripción
<code>filter_has_var()</code>	Esto comprueba si una variable existe en uno de las entradas de las matrices superglobales. Tiene dos argumentos: una de las constantes que figuran en la tabla 4.9.2 y el nombre de la variable que está buscando.
<code>filter()</code> _id	Esto devuelve el valor numérico de la constante de PHP por un nombre de filtro. Se toma un argumento: una cadena que contiene los nombres de filtro que devuelve <code>filter_list()</code> (ver tabla 4.9.2).
<code>filter_list()</code>	Esto devuelve una matriz que contiene los nombres de filtros de apoyo (ver tabla 4.9.2).
<code>filter_input()</code>	Utilice esta función para aplicar un filtro a una sola variable de una amplia superglobales. Toma hasta cuatro argumentos: una de las constantes que figuran en la tabla 4.9.2, el nombre de la variable, el filtro que se aplicará, y las opciones que se aplicará al filtro. Los dos últimos argumentos son opcionales. Si no hay ningún filtro se especifica, PHP usa el valor predeterminado.

Filter_input_array()	Esto filtra el contenido de un array superglobal de acuerdo a los criterios definidos por el usuario. Se necesitan dos argumentos: una de las constantes que figuran en la Tabla 4.9.2y una matriz multidimensional que define la forma en que cada elemento de la matriz superglobales se filtra. El segundo argumento es opcional, si no es suministrada, por defecto se aplica el filtro.
filter_var()	Esta se ocupa de una sola variable que viene de una fuente interna, como resultado de la base de datos o un cálculo realizado por el script (en otras palabras, en cualquier parte, excepto a partir de una matriz superglobales). Ocupa tres argumentos: el nombre de la variable, el filtro que debe aplicarse, y las opciones. Los dos últimos argumentos son opcionales. Si no hay ningún filtro se especifica por defecto.
filter_var_array()	Esto es idéntico a filter_input_array (), salvo que se filtra una serie de datos internos, en lugar de datos externos de una matriz superglobales. El primer argumento es el nombre de la matriz interna.

Tabla 4.9.1 Funciones para filtrar en PHP.

Tres funciones, filter_has_var(), filter_input(), y filter_input_array() son diseñadas para trabajar con matrices superglobales, como \$_POST y \$_GET. En lugar de utilizar las variables superglobales, es necesario hacer referencia a la matriz por su equivalente filtro constante. Las constantes y sus equivalentes se muestran en la Tabla 4.9.2. En el momento de escribir estas líneas, las constantes no se han aplicado todavía.

Constantes	Equivalentes Superglobales	Notas
input_COOKIE	Variables \$ COOKIE	
input_ENV	Variables \$ ENV	
input_GET	Variables \$ GET	
input_POST	Variables \$ POST	
input_REQUEST	Variables \$ REQUEST	No implementada todavía
input_SERVER	Variables \$ SERVER	
input_SESSION	Variables \$ SESSION	No implementada todavía

Tabla 4.9.2 Superglobales constantes utilizadas por funciones de filtro.

Aunque nuestro interés principal está en las funciones que hacen el filtrado, echemos una rápida búsqueda a las otros tres:

filter_has_var(): Esta función ofrece una forma alternativa para averiguar si una variable existe en una entrada. Se toma como argumentos una de las constantes que figuran en la tabla 4.9.2 y el nombre de la variable. Filter_has_var() se usa para comprobar si el array \$_POST contiene una variable componente denominada de esta manera:

```
if (filter_has_var(input_POST, 'component')) {  
    // Hacer algo  
}
```

Esto hace exactamente lo mismo que array_key_exists ('componente', \$_POST). Ambos prueban únicamente si la variable existe, no comprobar si tiene un valor. En la práctica, también puede utilizar isset (\$ ['componente'] \$_POST) para comprobar si una variable \$_POST existe. Sin embargo, hay una diferencia sutil en que, a diferencia de los otros dos isset () devuelve false si el valor que se está probando es nulo. Dado que las variables \$_POST siempre se transmiten en forma de cadenas, la única manera que el valor puede ser nulo es si es reiniciado después de haber sido recibido por el guión. Una cadena vacía devuelve verdadero.

filter_list(): Esto produce una matriz que contiene los nombres de los filtros compatibles con el servidor actual.

```
print_r(filter_list());
```

El objetivo principal de estos nombres es en combinación con `filter_id()`, se describe a continuación.

filter_id(): Este toma como un argumento de los nombres que aparecen por `filter_list()`, y devuelve un entero que representa uno de los filtros de apoyo. Por ejemplo, echo `filter_id('int')`; muestra 257. Usted puede ver los valores generados por `filter_id()` para cada filtro mediante la ejecución de `filter_id.php` en la carpeta `ch4_exercises`.

Probablemente esté pensando, "¿Qué diablos es el valor en eso?"

El valor práctico más importante para `filter_id()` es como un sustituto de las constantes de PHP utilizada por las funciones de filtro principal. En lugar de utilizar `FILTER_VALIDATE_INT` como argumento al filtrar la entrada, puede utilizar `filter_id('int')`. Voy a explicar por qué esto es útil más adelante en el capítulo en la "Prueba de otras constantes de filtro" del ejercicio.

4.9.1 Configuración de las opciones del filtro

La clave para utilizar las cuatro restantes funciones de filtro se encuentra en el establecimiento de las opciones correctas para cada filtro. Al ver cómo funcionan, usted comprenderá ¿por qué este capítulo acerca de ellos envuelve una interfaz más fácil de usar. En el momento de este escrito, PHP tiene soporte para 18 filtros, que pueden clasificarse de la siguiente manera:

validation filters: Estas comprueban que la entrada es de un tipo de datos en particular, como entero o booleano, o que se ajusta a un formato específico, como la dirección de correo electrónico. Si la prueba de validación se realiza correctamente, el filtro normalmente devuelve la entrada sin cambios. Si la prueba falla, el filtro devuelve falso.

Sanitizing filters: Esta tira de caracteres específicos de salida de entrada y retorna la versión desinfectada. Por ejemplo, usted puede utilizar para evitar que alguien entre en un comentario en una página del blog:

```
<a href="#" onclick="javascript:window.location='http://example.com/dosomethingbad.php'">Freebies galore!</a>
```

El filtro puede eliminar las etiquetas `<a>` completamente o convertir la apertura y cierre de corchetes en entidades HTML para evitar el acoplamiento y la secuencia de comandos de trabajo.

¡Tienes que ser muy cuidadoso al usar filtros de desinfección con los números, como el decimal, el punto se elimina en determinadas circunstancias, cambiando por completo el valor del número.

Custom filter: La devolución de llamada al filtro acepta una función definida por el usuario y los filtros de entrada de acuerdo a sus criterios.

Cada filtro se identifica por una constante de PHP. La mayoría de los filtros tienen opciones, banderas, o ambos. Estos le permiten ajustar con precisión el funcionamiento de un filtro. La mayoría de las banderas se pueden utilizar sólo con los filtros relacionados, pero los indicadores que se indican en la Tabla 4.9.1.1 se pueden utilizar con cualquier filtro. Los filtros de validación se enumeradas en la tabla 4.9.1.2.

Bandera	Descripción
<code>FILTER_REQUIRE_ARRAY</code>	Este rechaza cualquier valor que no es una matriz. Cuando este indicador está establecido, el filtro se aplica a todos los elementos de la matriz. También trabaja en matrices multidimensionales, la aplicación del filtro de forma recursiva para cada nivel.
<code>FILTER_REQUIRE_SCALAR</code>	Este rechaza cualquier valor que no es escalar. En otras palabras, el valor debe ser uno de los siguientes tipos de datos: Booleano, entero, el número de punto flotante, de recursos (por ejemplo, como un enlace de base de datos o identificador de archivo), o una cadena.

Tabla 4.9.1.1 Las banderas que se pueden utilizar con cualquier filtro.

Filtro	Opciones	Bandera	Descripción
FILTER_VALIDATE_BOOLEAN		FILTER_NULL_ON_FAILURE	Cuando se utiliza sin una bandera, devuelve true de '1', 'true', 'on', y 'sí'; devuelve false en caso contrario. Si la Bandera FILTER_NULL_ON_FAILURE se establece, false sólo para '0', 'false', 'off', 'no', y una cadena vacía; NULL se devuelve para todos los valores no booleano.
FILTER_VALIDATE_EMAIL			Comprueba si un valor se ajusta al formato de correo electrónico.
FILTER_VALIDATE_FLOAT	decimal	FILTER_FLAG_ALLOW_THOUSAND	Comprueba si un número de punto flotante o entero, devuelve false para cualquier otro tipo de dato. La opción permite que el decimal use una coma como separador decimal. El ajuste de la bandera acepta números que contiene un separador de miles (coma es el defecto, pero es el período utiliza cuando el decimal se coloca en '.'). El valor devuelto siempre es despojado del Separador de miles, por medio de un punto decimal.
FILTER_VALIDATE_INT	min_range max_range	FILTER_FLAG_ALLOW_OCTAL FILTER_FLAG_ALLOW_HEX	Revisa los enteros y devuelve falso para cualquier otro tipo de dato. Específicamente el mínimo y máximo valor aceptable en una matriz asociativa utilizando min_range y max_range (usted puede fijar solamente uno o ambos). La bandera permite números octales y hexadecimales. Rechaza números con puntos decimales, siempre que la fracción sea cero (0). Ejemplo 10.0.
FILTER_VALIDATE_IP		FILTER_FLAG_IPV4 FILTER_FLAG_IPV6 FILTER_FLAG_NO_PRIV_RANGE FILTER_FLAG_NO_RES_RANGE	Controla que los valores sean una dirección IP. La bandera sigue las especificaciones en IPv4 o IPv6, o no rangos provenientes privados.
FILTER_VALIDATE_REGEXP	regexp		Valida un valor contra una expresión regular compatible. La de valor se devuelve, no sólo en la parte que coincida con la expresión normal.
FILTER_VALIDATE_URL		FILTER_FLAG_SCHEME_REQUIRED FILTER_FLAG_HOST_REQUIRED FILTER_FLAG_PATH_REQUIRED FILTER_FLAG_QUERY_REQUIRED	Comprueba si un valor se ajusta al formato de una dirección URL, opcionalmente con componentes necesarios como se especifica por las banderas.

Tabla 4.9.1.2 Filtros para validación.

Comprobación de una variable que contiene un número

El siguiente ejercicio muestra cómo utilizar `filter_var ()` para probar si una variable contiene un número.

1. Crear un archivo llamado `filter_var.php` en la carpeta `ch4_exercises`, e introduzca el siguiente código:

```
<?php
$var = 10;
$filtered = filter_var($var, FILTER_VALIDATE_INT);
var_dump($filtered);
?>
```

Esto asigna un número entero a `$var`. Esto es pasado al `filter_var ()` con el filtro constante para un número entero, `FILTER_VALIDATE_INT`, y el resultado se almacena en `$filtered`. La última línea pasa a `$filter` para `var_dump()` con el fin de mostrar el tipo de datos y el valor devuelto por `filter_var ()`.

2. Probar el script en el navegador, y usted debería ver el mismo resultado que en la Ilustración 4.9.1.2.

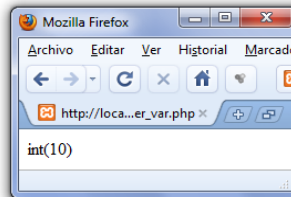


Ilustración 4.9.1.1 Usando `var_dump ()` muestra tanto el tipo de datos y su valor.

Esto confirma que `$var` tiene, de hecho, un número entero, y muestra su valor como 10.

3. Cambie la primera línea para incluir el número de citas como esta:

```
$var = '10';
```

4. Guarde la página y pruebe de nuevo. El resultado debe ser el mismo. A pesar de que `$var` es ahora una cadena, PHP interpreta una cadena que contiene sólo un número entero como un entero.
5. Cambie el valor en la primera línea a un número de punto flotante en una cadena como esta:

```
$var = '10.5';
```

6. Guarde la página y pruebe de nuevo. En ese momento deberá ver el resultado como se muestra en la Ilustración 4.9.1.3.

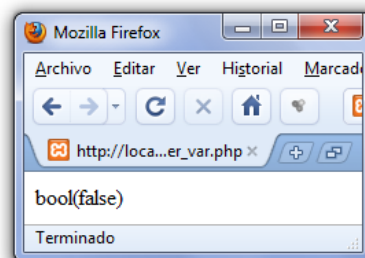


Ilustración 4.9.1.2 La función de filtro devuelve falso cuando la variable no pasa la prueba.

La variable no pasó la prueba, por lo que `filter_var()` devuelve el valor booleano falso. Para comprobar porque el número ya no es un número entero, y porque no es una cadena, quite las comillas y pruebe de nuevo la página. El resultado debe ser el mismo.

7. Cambie el filtro constante de prueba para un número de punto flotante como este:

```
$filtered = filter_var($var, FILTER_VALIDATE_FLOAT);
```

8. Pruebe la página de nuevo. Esta vez usted debe ver la confirmación de que \$var es una variable numérica de punto flotante, junto con su valor, como se muestra en la Ilustración 4.9.1.4.

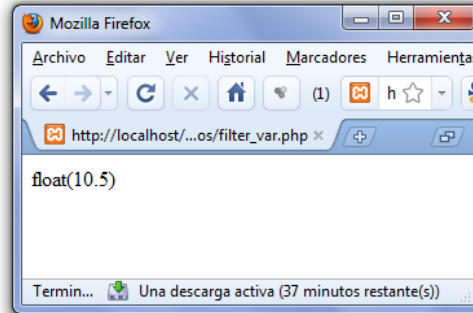


Ilustración 4.9.1.3 La confirmación de que \$ var es un número de punto flotante.

9. Cambiar el valor de \$var de nuevo a 10, y pruebe la página de nuevo. El resultado debe ser similar a la Ilustración 4.9.1.4.



Ilustración 4.9.1.4 El filtro para números decimales con punto flotante también acepta números enteros.

A pesar de que \$var es un número entero, es aceptada por el filtro. Por lo tanto, si desea comprobar para cualquier número, utilice FILTER_VALIDATE_FLOAT. Utilice sólo FILTER_VALIDATE_INT cuando es vital que el valor deber ser un número entero y nada más.

10. Por último, cambie el filtro permanente con una llamada a filter_id () como este:

```
$filtered = filter_var($var, filter_id('float'));
```

11. Prueba la página. El resultado debe ser idéntico al anterior. Esto simplemente demuestra que puede utilizar filter_id () con los nombres de los filtros si los encuentran más fácil de recordar.

Las únicas diferencias cuando se utiliza filter_input () es que es necesario especificar la fuente, y puesto que la variable proviene de una matriz, el nombre de la variable debe ser una cadena. Para adaptar la secuencia de comandos en el paso 1 del ejercicio anterior para filtrar ['var'] \$_POST, el cambio es de esta manera:

```
<?php
    $var = 10;
    $filtered = filter_var(input_POST, 'var', FILTER_VALIDATE_INT);
    var_dump($filtered);
?>
```

4.10 Trabajando con Bases de Datos.

Los sitios web modernos son muy poderosos, y gran parte de ese poder se deriva de su capacidad para almacenar información. El almacenamiento de información permite a los desarrolladores crear interacciones altamente personalizables entre su software y sus usuarios, que van desde blogs basados en entradas los sistemas de comentarios para alta potencia como aplicaciones bancarias que se encargan de las transacciones importantes de manera segura.

En esta parte cubriremos lo básico de MySQL, una potente base de datos de código abierto.

Para este ejemplo, deberá crear las siguientes tablas en un gestor de bases de datos, estamos almacenando información sobre los artistas musicales (véanse los cuadros 4.8.1.1 y 4.8.1.2).

artista_id	artista_nombre
1	Erick Andrés Ramírez.
2	Alexander Sáenz

Tabla 4.10.1 Tabla de artistas.

album_id	artista_id	album_nombre
1	1	Para Mercedes, Por Siempre.
2	1	Banco de Sueños EP3.
3	2	Acompáñame a estar Solo.
4	2	El tiempo pasado.

Tabla 4.10.2 Tabla de Álbum de artistas.

La primera tabla, artistas, incluye dos campos. El primer campo, artist_id, almacena un único valor numérico que identifica a los artistas. La segunda columna, artist_name, almacena el nombre del artista.

La segunda tabla, álbum, almacena un único identificador para cada álbum en la columna y album_id el nombre del álbum en el que lo creó! columna ALBUM_NOMBRE. En la tabla álbum incluye una tercera columna, artist_id, que se refiere a los artistas y las tablas de álbumes. Esta columna almacena el identificador único del artista que corresponde al artista que grabó el álbum.

No se profundiza más en aspectos de SQL, porque el punto principal de este capítulo no es aprender a utilizarlo, sino como utilizar PHP con bases de datos, en el anexo ---, están algunas propuestas sobre los gestores que usted puede utilizar y las bases de datos que puede administrar con ellos.

4.11 Abriendo una conexión

Se necesita un método mediante el cual los scripts PHP se puedan conectar a MySQL con el fin de interactuar con las bases de datos. Puede establecer esta conexión en cualquiera de varios enfoques:

- Extensión MySQL PHP.
- Extensión de PHP MySQLi.
- Objetos de datos PHP (PDO).

4.11.1 Extensión MySQL de PHP

La extensión de MySQL es la extensión original proporcionados por PHP que permite a los desarrolladores crear aplicaciones PHP que interactúan con bases de datos MySQL. La extensión de MySQL utiliza una interfaz de procedimiento, lo que significa que cada acción es una función (véase el ejemplo de código que sigue). Usted puede

utilizar la tabla `artist` descrita anteriormente como base para escribir un script PHP que recupere todos los nombres de los artistas. `Test.php`.

Abra el archivo con eclipse e introduzca el siguiente código:

```
<?php
// Abrir una conexión MySQL
$link = MySQL_connect('localhost', 'root', '');
if(!$link) {
    die('Connection failed: ' . MySQL_error());
}

// Selecciona la base de datos para trabajar
$db = MySQL_select_db('test');
if(!$db) {
    die('Selected database unavailable: ' . MySQL_error());
}

// Crear y ejecutar una consulta MySQL
$sql = "select artist_name FROM artist";
$result = MySQL_query($sql);

// Bucle a través de los datos devueltos y salida
while($row = MySQL_fetch_array($result)) {
    printf("Artista: %s<br />", $row['artist_name']);
}

// Liberar la memoria asociada con la consulta
MySQL_free_result($result);

// Cerrar la conexión
MySQL_close($link);
?>
```

Al navegar al sitio `http://localhost/simple_blog/test.php` en su navegador obtiene el siguiente resultado:

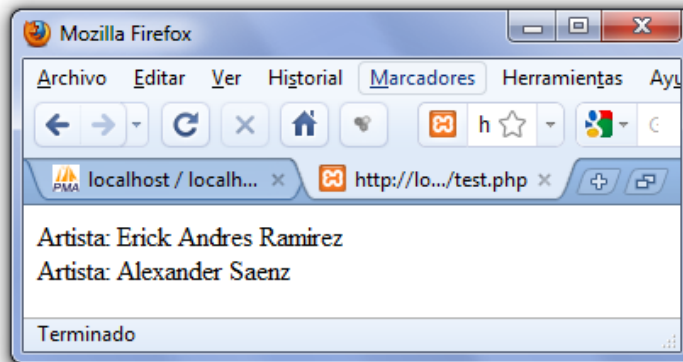


Ilustración 4.11.1.1 Salida de la consulta hecha desde PHP a la Base de Datos test.

Como el ejemplo anterior ilustra de inmediato, cada paso en el proceso tiene una función asignado a la misma:

`MySQL_connect()`: Acepta el host, nombre de usuario y una contraseña para una conexión MySQL. Debe llamar a esta función antes de cualquier interacción con la base de datos pueda llevarse a cabo.

`die()`: Este es un alias para el comando `exit()`. Se detiene la ejecución de una secuencia de comandos después de mostrar un mensaje opcional (pasado como argumento de la función).



MySQL_error(): Si ocurre un error en la base de datos MySQL, esta función muestra que el error, lo que es útil para la depuración.

MySQL_select_db(): Selecciona la base de datos con la que el script interactúa.

MySQL_query(): Ejecuta una consulta a la base de datos. Esta consulta puede crear, modificar, retornar o eliminar filas de la tabla, así como realizar muchas otras tareas.

MySQL_fetch_array(): Convierte el recurso MySQL devuelto por MySQL_query() en una matriz.

MySQL_free_result(): Libera la memoria utilizada por MySQL_query() para maximizar el rendimiento de escritura.

MySQL_close(): Cierra la conexión abierta por MySQL_connect().

La extensión de MySQL no es compatible con comandos preparados, por lo que es susceptible a la inyección de SQL, un potencialmente y devastador problema de seguridad en aplicaciones web. Los usuarios malintencionados pueden usar la inyección de SQL para extraer información sensible a partir de una base de datos, o incluso ir tan lejos como para borrar toda la información en una base de datos dada.

Usted puede minimizar este riesgo por desinfección de toda la información que desea insertar en el base de datos. La extensión de MySQL proporciona una función para escapar datos llamada MySQL_real_escape_string(), que se escapa (inserta una barra invertida antes) los caracteres especiales. Funciones adicional para la desinfección de datos, como htmlentities() y strip_tags(), están disponibles. Sin embargo, algunos riesgos existen, incluso si implementa estas medidas de seguridad.

4.11.2 La extensión MySQLi

El manual recomienda que los desarrolladores de MySQL usando MySQL 4.1.3 o posterior el uso de la extensión MySQLi. Hay muchas ventajas a usar MySQLi respecto a la original extensión de MySQL, incluyendo MySQLi de:

- Apoyo a los métodos de programación orientados a objetos y de procedimiento.
- Soporte para múltiples sentencias.
- Mejora de las capacidades de depuración.
- Soporte para comandos preparados.

4.11.2.1 Uso de Sentencias preparadas

El MySQLi y extensiones DOP proporcionan una característica muy útil en declaraciones preparadas. En pocas palabras, las declaraciones preparadas permiten separar los datos utilizados en un comando SQL desde el propio comando. Si usted no puede separar estas, un usuario malintencionado podría alterar sus comandos. Usando una declaración preparada significa que todos los datos presentados están totalmente escapados, que elimina la posibilidad de inyección de código SQL. Puede leer más sobre este tema en Harrison Fisk artículo en declaraciones preparadas en <http://dev.mysql.com/tech-resources/articles/4.1/preparedstatements.html>.

Una declaración preparada funciona de manera similar a una declaración regular de MySQL, excepto que utiliza un marcador de posición (un signo de interrogación [?]) para representar los datos. Usted puede hacer el mejor uso de sentencias preparadas donde el uso de la entrada del usuario es una consulta.

Por ejemplo, si usted tiene un formulario en su sitio que le pregunta el color favorito del usuario, usted podría hacer uso de la entrada en una consulta de MySQL a través de la superglobales \$_POST:

```
$sql = "select info FROM colors WHERE color = '$_POST[fav_color]'";
```

Sin embargo, usted no está realizando ningún saneamiento de esta entrada, por lo que un usuario malintencionado podría potencialmente explotar el formulario o dañar a su sitio mediante la inyección de SQL. Para evitar esto, puede volver a escribir la anterior declaración como una declaración preparada:

```
$sql = "select info FROM colors WHERE color = ?";
```

El signo de interrogación actúa como un marcador de posición, y le dice a MySQL que cualquier cosa que pasa a esta consulta se utilizará tan sólo como un parámetro para la instrucción actual. Esto evita a un usuario malintencionado engañar a MySQL para que dé información o dañe la base de datos.

Para utilizar MySQLi, de establecer una conexión con una interfaz orientada a objetos, así como debatir los pros y los contras de programación orientada a objetos frente a la programación de procedimiento.

La principal diferencia entre programación orientada a objetos y el código de procedimiento es que un objeto puede almacenar información, usted se libera de tener que pasar variables explícitamente de una función a otra.

Nota: MySQLi también proporciona una interfaz de procedimiento para los desarrolladores. Véase la entrada manual de PHP en MySQLi para más de la información.

Para familiarizarse con MySQLi, puede volver a escribir el ejemplo anterior utilizando MySQLi. Modificar Test.php que contiene lo siguiente:

```
<?php
// Abrir una conexión MySQL
$link = new MySQLi('localhost', 'root', '', 'test');
if(!$link) {
    die('Fallo la conexión: ' . $link->error());
}
// Crear y ejecutar una consulta MySQL
$sql = "select artist_name FROM artist";
$result = $link->query($sql);
// Bucle a través de los datos devueltos y salida
while($row = $result->fetch_assoc()) {
    printf("Artista: %s<br />", $row['artist_name']);
}
// Liberar la memoria asociada con la consulta
$result->close();
// Cerrar la conexión
$link->close();
?>
```

Navegar a http://localhost/simple_blog/test.php en su navegador obtiene el siguiente resultado:

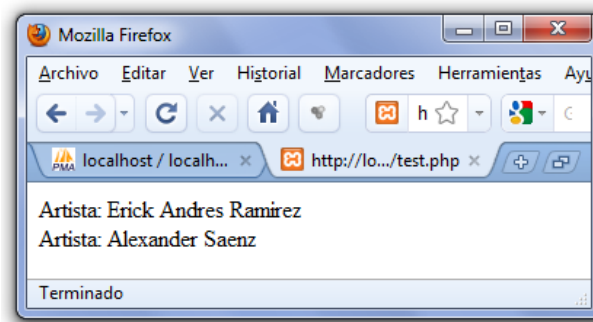


Ilustración 4.11.2.1.1 Utilizando MySQLi.

MySQLi funciona de manera similar a la extensión de MySQL, con una excepción fundamental: en lugar de proporcionar funciones individuales, los desarrolladores que utilizan MySQLi tienen acceso a métodos o funciones contenidas en el objeto MySQLi. En el fragmento de código anterior, usted instancia su objeto MySQLi en la variable `$link` y establece una conexión con el host, nombre de usuario, contraseña y un nombre de base de datos.

Para ejecutar una consulta, se llama al método `query()` y pasa la variable que contiene la declaración MySQL. Usted llama a un método en programación orientada a objetos utilizando la variable que contiene el objeto, seguido por una flecha (`->`) y el nombre del método al que desea llamar. Por ejemplo, esta línea del ejemplo de código ilustra cómo llamar a un método de programación orientada a objetos:

```
$result = $link->query($sql);
```

El método `query()` devuelve un objeto `MySQLi_result`, que tiene métodos que le permiten acceso a la información devuelta por la consulta.

Para acceder a cada entrada devuelta al final, se establece un bucle que utiliza el resultado de llamar a esta línea:

```
$result->fetch_assoc();
```

A continuación, destruir los datos devueltos mediante una llamada al método `cerrar()` en el objeto `$result`. Además, cerrar la conexión MySQLi mediante una llamada al método `close()` en `$link`, también.

Uso de Sentencias preparadas con MySQLi

Lo que realmente diferencia MySQLi aparte de la extensión de MySQL es su capacidad de usar comandos preparados. Si quiere permitir a un usuario seleccionar un artista que quiere ver álbumes, puede crear un formulario que se ve algo como esto:

```
<form method="post">
  <label for="artist">Seleccione un Artista:</label>
  <select name="artist">
    <option value="1">Alexander Sanez</option>
    <option value="2"></option>
  </select>
  <input type="submit" />
</form>
```

Cuando el usuario selecciona un artista, la identificación única del artista se pasa a la secuencia de comandos de procesamiento en el Variable ['artist'] `$_POST` (reparar el `$_POST`), que le permite cambiar su consulta basada en la entrada del usuario.

En `test.php`, puede crear un script rápido que muestra los nombres de álbum basado en la entrada del usuario:

```
<?php
if($_SERVER['REQUEST_METHOD']=='POST')
{
  // Abrir una conexión MySQL
  $link = new MySQLi('localhost', 'root', '', 'test');
  if(!$link) {
    die('Connection failed: ' . $mysqli->error());
  }
  // Crear y ejecutar una consulta MySQL
  $sql = "select album_name FROM albums WHERE artist_id=?";
  if($stmt = $link->prepare($sql))
  {
    $stmt->bind_param('i', $_POST['artist']);
    $stmt->execute();
    $stmt->bind_result($album);
    while($stmt->fetch()) {
      printf("Album: %s<br />", $album);
    }
    $stmt->close();
  }
  // Cerrar la conexión
```

```
$link->close();
}
else {
?>
<form method="post">
<label for="artist">Seleccione un Artista:</label>
<select name="artist">
<option value="1">Erick Andres Ramirez</option>
<option value="2">Feist</option>
</select>
<input type="submit" />
</form>
<?php } // Fin del else
?>
```

Cuando un usuario envía el formulario, un nuevo objeto MySQLi se crea, y una consulta se crea con un marcador para el `artist_id` en la cláusula `WHERE`. A continuación, puede llamar a la función `prepare()` en su Objeto MySQLi (`$link->prepare($sql)`) y pasar la consulta como un parámetro.

Con su estado de preparado (`$stmt`), tiene que decirle a MySQL cómo manejar la entrada del usuario e insertarlo en la consulta. Esto se conoce como parámetros vinculantes a la consulta, y llamando a la `bind_param()` en la recién creada `$stmt`, que es un objeto `MySQLi_stmt`.

Enlazar parámetros requiere un par de pasos: comenzar por pasar el tipo del parámetro, a continuación, pasar el valor del parámetro.

MySQLi admite cuatro tipos de datos:

- i: Integer (cualquier valor de número entero).
- s: String (cualquier combinación de caracteres).
- d: Double (cualquier número de punto flotante).
- b: Blob (los datos se envían en paquetes que se utiliza para almacenar imágenes u otros datos binarios).

Estás pasando la identificación del artista, por lo que establece el tipo de parámetro a `i`, a continuación, pasar el valor de `['artist'] $_POST`.

Con los parámetros vinculados, puede ejecutar la sentencia mediante el método `execute()`. Después de ejecutar la consulta, es necesario especificar las variables que contienen los resultados devueltos, lo que se logra con el método `bind_result()`. Para cada columna que ha solicitado, es necesario proporcionar una variable para contenerla. En este ejemplo, usted necesita almacenar el nombre del álbum, que se logra mediante el suministro de la variable `$album`.

La secuencia de comandos ahora sabe dónde almacenar los valores devueltos, por lo que puede crear un bucle para ejecutar al mismo tiempo resultados que aún existen (devuelto por el método `fetch()`). Dentro del bucle, la producción de cada nombre de álbum. Por último, se debe destruir el conjunto de resultados y cerrar la conexión mediante una llamada al método `close()` en tanto su `MySQLi_stmt` y objetos `MySQLi`, liberan la memoria utilizada por la consulta. Si carga el script navegando a `http://localhost/simple_blog/test.php` y seleccione Erick Andres Ramirez de la lista, verá el siguiente resultado:

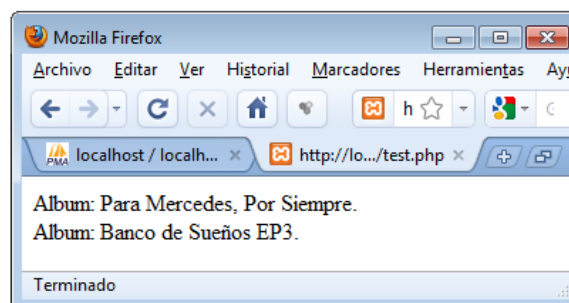


Ilustración 4.11.2.1.2 Esta es la salida si se seleccionó a Erick Andres Ramirez.

4.11.3 Objetos de datos PHP (PDO)

PHP Data Objects, o DOP, es similar a MySQLi en que trata de un enfoque orientado a objetos para la manipulación de consultas que soportan comandos preparados. La principal diferencia entre MySQLi y DOP es que DOP es una abstracción de base de datos de acceso en capa. Esto significa que la DOP soporta múltiples idiomas de bases de datos y proporciona un conjunto uniforme de métodos para el manejo de la mayoría de las interacciones de bases de datos.

Esta es una gran ventaja para aplicaciones que necesitan soporte de múltiples tipos de bases de datos, tales como PostgreSQL, Firebird, u Oracle. El cambio de tipo de base de datos de una a otra por lo general requiere escribir sólo una pequeña cantidad de código, que permite a los desarrolladores cambiar los controladores existentes de DOP y continuar con la rutina.

La desventaja de DOP es que algunas de las características avanzadas de MySQL no están disponibles, tales como apoyo a varias instrucciones. Otro problema potencial cuando se usa DOP es que se basa en la programación orientada a objetos características de PHP5, lo que significa que los servidores que ejecutan PHP4 no será capaz de ejecutar secuencias de comandos con denominación de origen. Esto puede convertirse en un problema menor con el tiempo porque algunos servidores no tienen acceso a PHP5, sin embargo, sigue siendo algo que tienes que tener en cuenta a la hora de elegir su método de acceso de base de datos.

4.11.3.1 Ejemplo de reescritura de su DOP

Usted puede utilizar DOP para volver a escribir su declaración preparada. En test.php, modificar el código de la siguiente manera:

```
<?php
    if($_SERVER['REQUEST_METHOD']=='POST')
    {
        // Abrir una conexión MySQL
        $dbinfo = 'MySQL:host=localhost;dbname=test';
        $user = 'root';
        $pass = '';
        $link = new PDO($dbinfo, $user, $pass);
        // Crear y ejecutar una consulta MySQL
        $sql = "select album_name
        FROM albums
        WHERE artist_id=?";
        $stmt = $link->prepare($sql);
        if($stmt->execute(array($_POST['artist'])))
        {
            while($row = $stmt->fetch()) {
                printf("Album: %s<br />", $row['album_name']);
            }
            $stmt->closeCursor();
        }
    }
    else {
        ?>
        <form method="post">
        <label for="artist">Seleccionet un Artista:</label>
        <select name="artist">
        <option value="1">Erick Andrés Ramírez</option>
        <option value="2">Alexander Saenz</option>
        </select>
        <input type="submit" />
        </form>
        <?php } // Fin del else
    ?>
```

El primer paso, es abrir la conexión de base de datos, es un poco diferente de los otros dos métodos que ha aprendido sobre la marcha. Esta diferencia surge del hecho de que DOP puede soportar múltiples tipos de bases de datos, lo que significa que es necesario especificar un controlador para crear el tipo de conexión.

En primer lugar, se crea una variable llamada `$dbinfo` que le dice a la DOP que inicie con el controlador de MySQL para el host `localhost` anfitrión y la base de datos `test`. A continuación, se crean dos variables más, la `$user` y `$pass`, para contener su nombre de usuario y contraseña de la base de datos.

Después de su conexión, se forma la consulta con un marcador de posición, pasar al método `prepare()`, y luego pasar la consulta que se ha preparado. Esto devuelve un objeto PDO Statement que se guarda en la variable `$stmt`. A continuación, se llama al método `execute()` con una matriz que contiene el identificador de artista suministrado por el usuario, `['artist'] $_POST`. Esto es equivalente a llamar tanto `bind_param()` y `execute()` con la extensión MySQLi.

Después que la sentencia se ha ejecutado, se establece un bucle que se ejecuta mientras los resultados todavía existen. Cada resultado se envía al navegador, y libera la memoria utilizando el método `closeCursor()`.

Al ejecutar este script de carga `http://localhost/simple_blog/test.php` dependiendo del artista que el usuario escoja, produce lo siguiente:

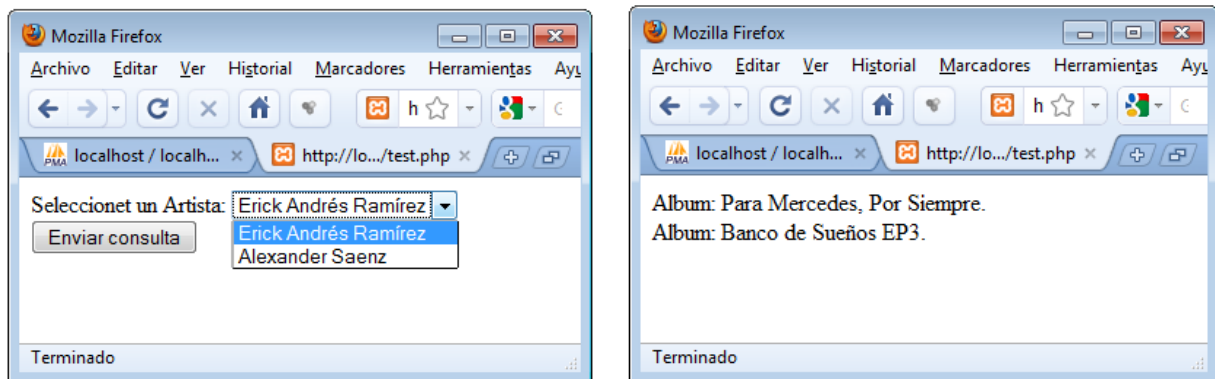


Ilustración 4.11.3.1.1 Al escoger al artista Erick Andres Ramirez, produce la salida del segundo cuadro.

Nota: DOP es muy versátil, así que confíe en que para la mayoría de los ejemplos usted se sentira libre para sustituirlo por otro método, pero tenga en cuenta que el código que interactúa con la base de datos se verá de manera diferente si lo hace.

4.12 Estructura de tabla y un curso acelerado en la planificación

Como las aplicaciones PHP son más complicadas, el rendimiento de su aplicación comenzará a desempeñar un papel clave en el desarrollo. MySQL es un asesino potencial de rendimiento en las aplicaciones, la creación de cuellos de botella que prevenir, que los scripts se ejecuten lo más rápido que ellos quieren.

Parte de su papel como desarrollador es conocer los riesgos relacionados con las consultas de MySQL y eliminar muchos de los problemas de rendimiento posible:

- Mala planificación de las tablas de base de datos.
- Las solicitudes de datos innecesarios (usando el selector de acceso directo `[*]`).

Tenga en cuenta que la arquitectura de base de datos y la optimización es una tarea enorme, es en realidad una carrera en sí misma. Solo tocamos algunos de los aspectos básicos de este tema, sin embargo, conocer los problemas de rendimiento potencial asociado con sus bases de datos es importante porque no siempre se puede tener un arquitecto de bases de datos disponibles para usted.

4.12.1 Planeando tablas de bases de datos

Las tablas de bases de datos son bastante simples, pero las tablas entre sí pueden ser un obstáculo para los rendimientos de las secuencias de comandos si no se planifica adecuadamente. Por ejemplo, una lista de las entradas de blog y una lista de páginas del sitio web. Podrían ser algo como esto (véanse las tablas 412.1.1 y 412.1.2).

title	Text
Entrada.	Este es un texto.
Entrada de Titulo.	Aquí hay más texto.
Ejemplo de Titulo.	Esta es una tercera entrada.

Tabla 4.12.1.1 Entradas de la tabla.

Page_name	Type
Blog	Multi
About	Static
Contact	Form

Tabla 4.12.1.2 Páginas de la tabla.

Page	Entry
Blog	Ejemplo de Entrada
Blog	Titulo de Entrada
About	Ejemplo de Titulo.

Tabla 4.12.1.3 Entradas de las páginas de la talba.

Desafortunadamente, usted tiene almacenamiento de datos redundantes en el presente caso, que son innecesarios y potencialmente perjudiciales para el rendimiento de su aplicación. Si sabes que estás en la página de blog, puede utilizar dos consultas para recuperar los datos de entrada:

```
<?php
// Iniciar la DOP objeto
$dbinfo = 'MySQL:dbname=test;host=localhost';
$user = 'root';
$pass = '';
try {
    $db = new PDO($dbinfo, $user, $pass);
} catch(PDOException $e) {
    echo 'Connection failed: ', $e->getMessage();
}
// Crea la primera consulta
$sql = "select entry
FROM entry_pages
WHERE page='Blog'";
// Inicializar la variable $entradas en caso de que no hay datos guardados
$entries = NULL;
// Recupera las entradas de la tabla
foreach($db->query($sql) as $row) {
    $sql2 = "select text FROM entries WHERE title='$row[entry]'";
    foreach($db->query($sql2) as $row2) {
        $entries[] = array($row['title'], $row2['entry']);
    }
}
```



```
}
// Mostrar la salida
print_r($entries);
?>
```

Este código devuelve lo siguiente:

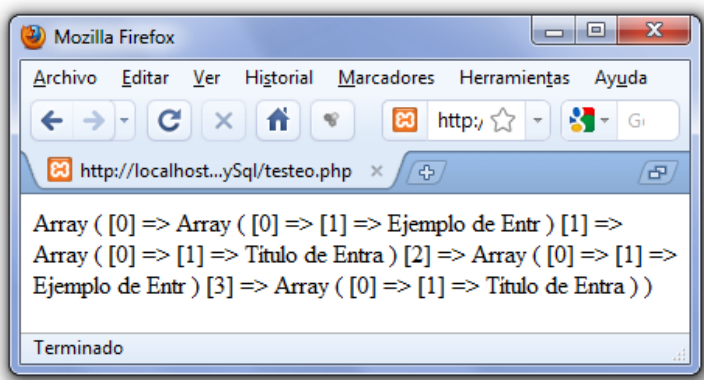


Ilustración 4.12.1.1 Respuesta del evridor local.

Nota: para obtener una salida del código anterior trabajando, es necesario crear la tabla entradas de la página y rellenarla con los datos que se describen en las Tabla 4.12.1.3.

Sin embargo, este enfoque es muy ineficiente porque se están recuperando los datos redundantes de las tablas entry_pages y de las entradas. Una forma de evitar este problema es añadir una columna adicional a las tablas que contienen un identificador para cada fila que automáticamente se incrementará cuando se agregan filas. Esto permite una menor redundancia de datos almacenados. Por ejemplo, suponga que desea volver a visitar sus páginas y entradas mediante la adición de una columna de identificación (véase Tablas 4.12.1.4 y 4.12.1.5).

id	Page_name	Type
1	Blog	Multi
2	About	Static
3	Contact	Form

Tabla 4.12.1.4 Revisión de las páginas de la tabla.

Id	Page_id	Title	Texto
1	1	Entrada Simple	Este es algún texto.
2	1	Entrada Título	Este es algún texto.
3	2	Entrada Simple	Este es algún texto.

Tabla 0.1 Revisión de las entradas de la tabla.

La adición de una columna id le permite eliminar por completo entry_pages, que te deja con una manera fácil de referencias cruzadas de las tablas. Esto significa que usted puede volver a escribir la secuencia de comandos para recuperar las entradas para una página en particular mucho más eficiente:

```
<?php
$dbinfo = 'MySQL:dbname=test;host=localhost';
$user = 'root';
$pass = '';
```



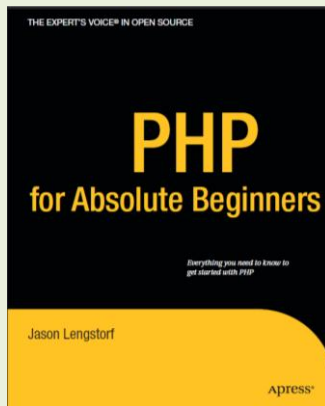
```
try {  
    $db = new PDO($dbinfo, $user, $pass);  
} catch(PDOException $e) {  
    echo 'Connection failed: ', $e->getMessage();  
}  
$sql = "select title, text  
FROM entries  
WHERE page_id=1";  
foreach($db->query($sql) as $row) {  
    $entries[] = array($row['title'], $row['text']);  
}  
print_r($entries);  
?>
```

Este código devuelve un resultado idéntico al ejemplo anterior, pero tiene sólo la mitad de tiempo para ejecutarse, ya que sólo se ejecuta una consulta en lugar de las dos originales.

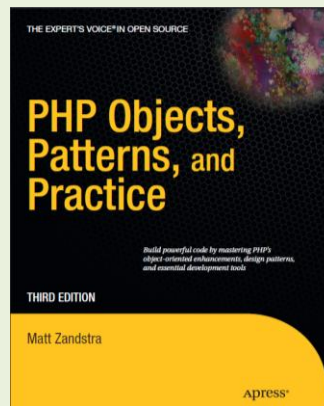
Tenga en cuenta que el diseño de base de datos es un área altamente especializada de la programación. Puede obtener más información al respecto por recoger una copia del diseño de partes de bases de datos: desde el principiante al profesional de Clare Churcher (Apress, 2007).

Bibliografía del Capítulo 4

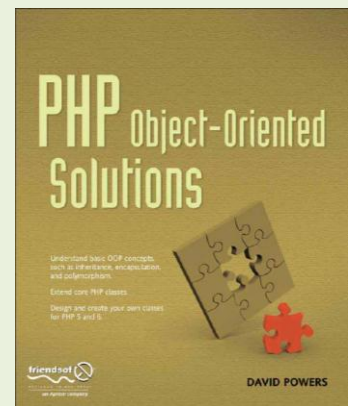
PHP for Absolute Beginners



PHP Objects Patterns, and Practice



PHP Object-Oriented Solutions





Capítulo 5

XML

5 XML

5.1. ¿Qué es el XML?

XML significa eXtensible markup language, o lenguaje de anotación extensible. Ya conocemos el lenguaje HTML (*hypertext markup language*), lenguaje de anotación para página webs que permite navegación tipo hipertexto; sin embargo, XML no es sólo un lenguaje, es una forma de especificar lenguajes, de ahí lo de extensible. Todo lenguaje que se exprese de una forma determinada puede ser XML. Por lo tanto, XML *no es un lenguaje para hacer mejores páginas web*, sino un lenguaje para información auto-descrita, o al menos, auto-descrita si las etiquetas están bien puestas.

XML es interesante en el mundo de Internet y el e-bussiness, ya que existen muchos sistemas distintos que tienen que comunicarse entre sí, pero como se ha podido imaginar, interesa por igual a todas las ramas de la informática y el tratamiento de datos, ya que permite muchos avances a la hora de trabajar con ellos.

XML significa Extensible Markup Language

XML es un lenguaje muy similar a HTML

XML fue diseñado para transportar datos, no para mostrar los datos

Etiquetas XML no están predefinidas. Usted debe definir sus propias etiquetas

XML está diseñado para ser auto-descriptivo

XML es una Recomendación del W3C

5.2. La diferencia entre XML y HTML:

- XML fue diseñado para transportar y almacenar los datos, con especial atención a los datos que se envían.
- HTML fue diseñado para mostrar los datos, con especial atención a la cantidad de datos que se ven.

Antes de continuar, usted debe tener un conocimiento básico de:

HTML.

JavaScript.

Tal vez sea un poco difícil de entender, pero XML no hace nada. XML fue creado para estructurar y almacenar la información de transporte.

Ejemplo:

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

La nota de arriba se describe a sí misma. Tiene remitente e información del receptor, también tiene un título y un cuerpo del mensaje.

Pero todavía, este documento de XML no HACE NADA. Es sólo información envuelta en las etiquetas. Alguien debe escribir un trozo de software para enviar, recibir o desplegarla.

Con XML usted inventa sus propias etiquetas.

Las etiquetas en el ejemplo sobre (<to> y <from>) no se definen en cualquier norma de XML. Estas etiquetas son "inventadas" por el autor del documento de XML. En XML el autor puede definir sus propias etiquetas y su propia estructura para sus documentos.

Eso es porque el idioma de XML no tiene ninguna etiqueta predefinida. Las etiquetas usadas en HTML son predefinidas. En los documentos HTML puede usar sólo etiquetas definidas en los estándares de HTML normal (como <p>, <h1>, etc.).

Es importante entender que XML no es un reemplazo para HTML. En la mayoría de aplicaciones web, XML se acostumbra para transportar los datos, mientras HTML se usa para estructurar y desplegar los datos.

XML es un software - y herramienta hardware-independiente para llevar la información. XML es una recomendación de la W3School desde el 10 de febrero de 1998.

5.3. Utilización XML

Se utiliza en muchos aspectos del desarrollo web, a menudo para simplificar el almacenamiento e intercambio de datos. Si necesita mostrar datos dinámicos en un documento HTML, que tendrá una gran cantidad de trabajo para editar el código HTML cada vez que cambian los datos. Con XML, los datos pueden ser almacenados en archivos XML. De esta manera usted puede concentrarse en el uso de HTML para el diseño y la presentación por pantalla, asegurándose de que los cambios en los datos subyacentes no requieran de ningún cambio en el código HTML.

Con unas pocas líneas de código JavaScript, se puede leer un archivo XML externo y actualizar el contenido de los datos de su página web. En el mundo real, los sistemas informáticos y bases de datos contienen datos en formatos incompatibles. Los datos XML se almacenan en formato de texto plano. Esto proporciona un software independiente del hardware y forma de almacenamiento de datos. Esto hace que sea mucho más fácil para crear datos que pueden ser compatibles por diferentes aplicaciones.

XML simplifica el transporte de los datos, uno de los retos que más tiempo consume a los desarrolladores es el intercambio de datos entre sistemas incompatibles a través de Internet. El intercambio de datos con XML reduce en gran medida esta complejidad, ya que los datos pueden ser leídos por diferentes aplicaciones incompatibles.

XML simplifica los cambios de plataformas, con la actualización de nuevos sistemas (plataformas de hardware o software), siempre consume mucho tiempo. Grandes cantidades de datos se deben convertir y a menudo cuando son incompatibles, se pierde información. Los datos XML se almacenan en formato de texto. Esto hace que sea más fácil de ampliar o actualizar a los nuevos sistemas operativos, aplicaciones nuevas, o nuevos navegadores, sin perder los datos.

Las diferentes aplicaciones pueden acceder a sus datos, no sólo en las páginas HTML, sino también de fuentes de datos XML.

Con XML, sus datos pueden estar disponibles para todo tipo de “máquinas de lectura” (computadoras de mano, máquinas de voz, noticias, etc), y hacerlo más accesible para las personas ciegas o personas con otras discapacidades.

El futuro nos puede dar los procesadores de texto, hojas de cálculo y bases de datos que pueda leer los datos de otros en formato XML, sin ningún tipo de utilidades de conversión en el medio.

5.4. Árbol XML.

Los documentos XML utilizan una auto-descripción y sintaxis simple:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Erika</to>
  <from>Renier</from>
  <heading>Remitente</heading>
  <body>No se olvide de mí, en este fin de semana!</body>
</note>
```

La primera línea es la declaración XML. Esto define la versión XML (1.0) y la codificación usada (ISO-8859-1 = Latin-1/West European carácter set).

La siguiente línea describe el elemento root del documento (dice: "Este documento es una nota")
<nota>

Las siguientes 4 líneas describen 4 elementos hijos del elemento root (to, from, heading y body):

```
<para>Erika</para>
<de>Reniery</de>
<encabezado>Recordatorio</encabezado>
<cuerpo>No te olvides de mi este fin de semana!</cuerpo>
```

Finalmente la última línea define el final del element root:
</nota>

Usted puede asumir y deducir, de este ejemplo, que el documento de XML contiene una nota a Erika de Reniery. ¿Usted no está de acuerdo que XML es bastante descriptivo por si mismo?

Todos los documentos XML forman un árbol. Los documentos de XML deben contener un elemento de la raíz. Este elemento es "el padre" de todos los otros elementos. El árbol empieza en la raíz y ramas al nivel más bajo del árbol.

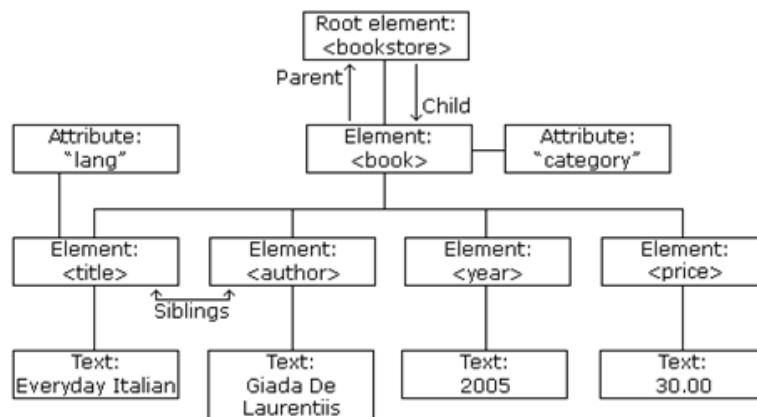
Todos los elementos pueden tener los elementos del subalterno (los elementos del hijo):

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

Los terminos padre, hijo y hermano son usados para describir las relaciones entre elementos. Los elementos padre tienen elementos hijos. Los hijo que estén en el mismo nivel son llamados (hermanos o hermanas).

Todos los elementos pueden contener texto y atributos (tales como en HTML).

Ejemplo:



La imagen de arriba, representa un libro en el XML que mostramos aquí:

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
```

```
<year>2005</year>
<price>30.00</price>
</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

El elemento de la raíz en el ejemplo es `<bookstore>`. Todos los elementos `<book>` en el documento se contienen dentro de la etiqueta `<bookstore>`.

El elemento `<book>` tiene 4 hijos: `<title>`, `<author>`, `<year>`, `<price>`.

5.5. Reglas de sintaxis

Las reglas de sintaxis de XML son muy sencillas y lógicas. Las reglas son fáciles de aprender y fácil de usar.

1. Todos los elementos XML deben tener una etiqueta de cierre.

En HTML, algunos elementos no tienen que tener una etiqueta de cierre:

```
<p> Este es un párrafo.
<p> Este es otro párrafo.
```

En XML es ilegal omitir la etiqueta de cierre. Todos los elementos deben tener una etiqueta de cierre.

```
<p> Este es un párrafo </p>
<p> Este es otro párrafo </p>
```

Usted podría haber notado del ejemplo anterior que la declaración de XML no tenía una etiqueta del cierre. Éste no es un error. La declaración no es una parte del propio documento de XML, y no tiene ninguna etiqueta del cierre.

2. Los elementos XML deben estar propiamente anidados.

En HTML usted podría ver los elementos inadecuadamente anidados:

```
<b><i> Este texto esta en negrita </b></i>
```

En XML, todos los elementos deben anidarse propiamente uno dentro de otro:

```
<b><i> Este texto esta en negrita </i></b>
```

En el ejemplo de arriba, "Propiamente anidó" simplemente los elementos medios como el elemento `<i>` deben anidarse dentro del elemento `<b>`.

3. Los documentos XML deben tener un elemento Root (raíz).

Los documentos de XML deben contener un elemento que es el padre de todos los otros elementos. Este elemento se llama el elemento de la raíz.

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

4. Los valores de los atributos XML deben estar entre comillas dobles.
Los elementos de XML pueden tener los atributos simplemente en los pares del name/value como en HTML. En XML, los valores del atributo deben citarse siempre.

Estudiamos los dos documentos XML de debajo. El primero es incorrecto, el segundo es correcto:

```
<note date=12/11/2007>  
<to>Tove</to>  
<from>Jani</from>  
</note>
```

```
<note date="12/11/2007">  
<to>Tove</to>  
<from>Jani</from>  
</note>
```

El error en el primer documento es que el atributo de la fecha en el elemento de la nota no se cierra entre comillas.

5. Las Referencias de la entidad.
Algunos caracteres tienen un significado especial en XML.
Si usted pone un carácter como este "<" dentro de un elemento de XML, generará un error porque el parser lo interpreta como la salida de un nuevo elemento.

Esto generará un error de XML:

```
<message>if salary < 1000 then</message>
```

Para evitar este error, reemplace el "<" el carácter con una referencia de la entidad:

```
<message>if salary &lt; 1000 then</message>
```

6. Comentarios en XML.
La sintaxis para escribir comentarios en XML es similar a la de HTML.
`<!--Este es un comentario -->`

7. Los espacios en blanco son reservados por XML.
HTML trunca los caracteres de espacio-blanco múltiples a un solo espacio- blanco:
HTML: Hello Tove
Output: Hello Tove

Con XML, el espacio- blanco en un documento no se trunca.

8. XML almacena la Nueva Línea como LF.
En las aplicaciones de Windows, una nueva línea se guarda normalmente como un par de caracteres: el retorno del carruaje (CR) y la línea alimentaba (LF). En las aplicaciones de Unix, una nueva línea se guarda normalmente como un carácter de LF. Las aplicaciones de Macintosh también usan un LF para guardar una nueva línea.

XML almacena la línea como un LF.

5.6. ¿Qué es un elemento XML?

Un elemento XML es todo lo que a partir de (incluyendo) la etiqueta inicial del elemento (a incluido) etiqueta final del elemento.

Un elemento puede contener:

- otros elementos
- texto
- atributos
- o una mezcla de todo lo anterior ...

```
<bookstore>
  <book category="CHILDREN">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

En el ejemplo de arriba, <bookstore> y <book> tiene los volúmenes del elemento, porque ellos contienen otros elementos. <book> también tiene un atributo (la categoría = "CHILDREN"). <title>, <author>, <year>, y <price>, ellos contienen el texto.

5.7. Atributos en XML

Los elementos XML pueden tener atributos, al igual que HTML. Los atributos proporcionan información adicional acerca de un elemento.

```

<a href="demo.asp">
```

Los atributos proporcionan a menudo información que no es parte de los datos. En el ejemplo de bajo, el tipo del archivo es no pertinente a los datos, pero puede ser importante para el software que quiere manipular el elemento:

```
<file type="gif">computer.gif</file>
```

Los atributos en XML deben ponerse entre comillas. Siempre deben ponerse entre comillas los valores del atributo. Pueden usarse las solas o dobles. Para el sexo de una persona, el elemento de la persona puede escribirse así:

```
<person sex="female">
```

O como esto:

```
<person sex='female'>
```

Si el propio valor del atributo contiene las comillas dobles, usted puede usar las sencillas, como en este ejemplo:

```
<gangster name='George "Shotgun" Ziegler'>
```

o usted puede usar las entidades del carácter:

```
<gangster name="George &quot;Shotgun&quot; Ziegler">
```

Los Elementos de XML contra los Atributos:

Eche una mirada a estos ejemplos:


```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

En el primer ejemplo, sexo es un atributo. En el último, el sexo es un elemento. Ambos ejemplos proporcionan la misma información.

No hay ninguna regla sobre cuándo usar los atributos o cuándo usar los elementos. Los atributos son hábiles en HTML. En XML mi consejo es evitarlos. Mejor use los elementos.

5.8. Validación con XML

Un documento "bien formado" en XML tiene una sintaxis correcta. El aspecto de la sintaxis la hemos descrito en páginas anteriores.

Documentos XML deben tener un elemento raíz.
Elementos XML deben tener una etiqueta de cierre.
Etiquetas XML distinguen entre mayúsculas y minúsculas.
Los elementos XML deben estar anidados correctamente.
Valores de atributos XML deben ser citados.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Documentos validos en XML:

Un documento "Válido" de XML está "Bien Formado" si está documentado, porque también conforma las reglas de una Definición de Tipo de Documento (DTD):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

La declaración de DOCTYPE en el ejemplo de arriba, es una referencia a un archivo de DTD externo. El volumen del archivo se muestra en el párrafo debajo.

El propósito de un DTD es definir la estructura de un documento de XML. Define la estructura con una lista de elementos legales:

```
<!DOCTYPE note
[
  <!ELEMENT note (to,from,heading,body)>
```

```
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

El Esquema de XML

W3C apoya una alternativa XML-basado a DTD, el Esquema de XML:

```
<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

5.9. Visualización de archivos XML

No espere que los archivos XML se muestren como páginas HTML.

Visualización de los archivos XML:

```
<? Xml version = "1.0" encoding = "ISO-8859-1">
- <> nota
<to> Erika </ to>
<from> Reniery </ from>
<título> Recordatorio </ título>
<body> No te olvides de mí este fin de semana! </ body>
</ nota>
```

El documento XML se muestra con un código de colores de raíces y elementos secundarios. Un signo más (+) o menos (-) a la izquierda de los elementos se puede hacer clic para expandir o contraer la estructura del elemento. Para ver el código XML sin procesar (sin signos + y -), seleccione "fuente de la página View" o "Ver código fuente" en el menú del navegador.

Nota: En Chrome, Opera y Safari, sólo el texto del elemento en la pantalla. Para ver el XML sin procesar, debe hacer clic derecho en la página y seleccione "Ver código fuente".

Si se abre un archivo erróneo en el navegador, éste mostrará el error.

5.10. El objeto XMLHttpRequest

El objeto XMLHttpRequest se utiliza para intercambiar datos con un servidor detrás de las escenas. El objeto XMLHttpRequest es el **sueño de un desarrollador**, ya que puede:

- Actualización de una página web sin necesidad de recargar la página.
- Datos de la solicitud de un servidor después de la página se haya cargado.
- Recibir datos de un servidor después de la página se haya cargado.
- Enviar datos a un servidor en segundo plano.

Ejemplo XMLHttpRequest:

Cuando se escribe un carácter en el campo de entrada siguiente, un XMLHttpRequest se envía al servidor - nombre y sugerencias son devueltos (a partir de un archivo en el servidor):

Crear un archivo XMLHttpRequest:

Todos los navegadores modernos (Internet Explorer 7 +, Firefox, Chrome, Safari y Opera) tiene un objeto XMLHttpRequest incorporado.

Sintaxis para crear un objeto XMLHttpRequest:

```
xmlhttp=new XMLHttpRequest();
```

Las versiones viejas de Internet Explorer (IE5, IE6) usan un ActiveX Object:

```
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
```

Todos los navegadores modernos tienen un analizador integrado en XML. Un analizador de XML se convierte un documento XML en un objeto XML DOM - que se pueden manipular con un JavaScript.

Analizar un documento XML:

El fragmento de código siguiente se analiza un documento XML en un objeto XML DOM:

```
if (window.XMLHttpRequest)
{
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
}
else
{
    // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.open("GET","books.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;
```

Analice una cadena de XML.

El fragmento del código siguiente analiza una cadena de XML en un XML del objeto de DOM:

Escribir una letra en el cuadro de entrada:

Nombre

Sugerencias:

```
txt="<bookstore><book>";
txt=txt+"<title>Everyday Italian</title>";
txt=txt+"<author>Giada De Laurentiis</author>";
txt=txt+"<year>2005</year>";
txt=txt+"</book></bookstore>";

if (window.DOMParser)
{
```

```
parser=new DOMParser();
xmlDoc=parser.parseFromString(txt,"text/xml");
}
else // Internet Explorer
{
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.loadXML(txt);
}
```

Nota: Internet Explorer usa el método loadXML () para analizar una cadena de XML, mientras otros navegadores usan el objeto de DOM Parser.

Acceda Por los Dominios.

Por las razones de seguridad, los navegadores modernos no permiten el acceso por los dominios. Esto significa, que la página web y " los archivos XML que intenta cargar, debe localizarse en el mismo servidor.

XML y DOM

Como es de su conocimiento, DOM (Document Object Model), define una manera estándar para acceder y manipular los documentos.

El DOM XML define una manera estándar para acceder y manipular documentos XML.

El punto de vista de un documento desde DOM XML es como una estructura de árbol.

Todos los elementos se pueden acceder por medio del árbol DOM. Su contenido (texto y atributos) pueden ser modificados o eliminados, y nuevos elementos se pueden crear. Los elementos, su texto y sus atributos son conocidos como nodos. Todos los elementos HTML se pueden acceder a través del DOM.

El siguiente ejemplo, analiza un documento XML ("nota.xml") en un objeto XML DOM y luego extrae algo de información desde un JavaScript.

```
<html>
<body>
<h1>Facultad de Ciencias UNAN, Nota Interna</h1>
<div>
<b>To:</b> <span id="to"></span><br />
<b>From:</b> <span id="from"></span><br />
<b>Message:</b> <span id="message"></span>
</div>

<script type="text/javascript">
if (window.XMLHttpRequest)
{ // code for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp=new XMLHttpRequest();
}
else
{ // code for IE6, IE5
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.open("GET","note.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;

document.getElementById("to").innerHTML=
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
document.getElementById("from").innerHTML=
xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
```

```
document.getElementById("message").innerHTML=
xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
</script>

</body>
</html>
```

Para extraer el texto “Erika” del element <a> del archivo XML que esta arriba, la sintaxis que utilizaremos será:

```
getElementsByTagName("a")[0].childNodes[0].nodeValue
```

Note que aun cuando el archivo de XML contiene un único elemento <a>, usted tiene que especificar el índice de la serie [0]. Esto es porque el método `getElementsByTagName()` devuelve una serie.

Agregar código HTML a un documento XML.

En el siguiente ejemplo, recorrer un archivo XML (“cd catalogo.xml”), y mostrar el contenido de cada elemento de CD como una fila de la tabla HTML.

```
<html>
<body>

<script type="text/javascript">
if (window.XMLHttpRequest)
    { // code for IE7+, Firefox, Chrome, Opera, Safari
      xmlhttp=new XMLHttpRequest();
    }
else
    { // code for IE6, IE5
      xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
xmlhttp.open("GET","cd_catalog.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;

document.write("<table border='1'>");
var x=xmlDoc.getElementsByTagName("CD");
for (i=0;i<x.length;i++)
    {
      document.write("<tr><td>");
      document.write(x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue);
      document.write("</td><td>");
      document.write(x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue);
      document.write("</td></tr>");
    }
document.write("</table>");
</script>

</body>
</html>
```

Ahora, obtendremos los datos XML del primer elemento CD y lo mostraremos en un elemento HTML con `id="showID"`. El método `displayCD()` es llamado cuando se carga la página:

```
x=xmlDoc.getElementsByTagName("CD");
i=0;

function displayCD()
{
  artist=(x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue);
  title=(x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue);
```

```
year=(x[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue);  
txt="Artist: " + artist + "<br />Title: " + title + "<br />Year: "+ year;  
document.getElementById("showCD").innerHTML=txt;  
}
```

Para navegar entre los CD en el ejemplo de arriba, agregue las funciones siguiente () y anterior ():

```
function siguiente()  
{ // display the next CD, unless you are on the last CD  
if (i<x.length-1)  
{  
    i++;  
    displayCD();  
}  
}  
  
function anterior()  
{ // displays the previous CD, unless you are on the first CD  
if (i>0)  
{  
    i--;  
    displayCD();  
}  
}
```

5.11. Espacios de nombres XML

Los espacios de nombres XML proporcionan un método para evitar conflictos entre nombres de elementos.

Este XML lleva información de la tabla HTML:

```
<table>  
  <tr>  
    <td>Apples</td>  
    <td>Bananas</td>  
  </tr>  
</table>
```

Este XML lleva información acerca de una tabla (una pieza de mobiliario):

```
<table>  
  <name>African Coffee Table</name>  
  <width>80</width>  
  <length>120</length>  
</table>
```

Si estos dos fragmentos estuvieran en un mismo documento, habría un conflicto de nombres. Los dos contienen un elemento <tabla>, como podemos observar los elementos del volumen tienen diferente contenido y significado.

Este problema se puede resolver fácilmente usando un prefijo de nombre.

En este siguiente ejemplo, el XML lleva información de dos tablas HTML:

```
<h:table>  
  <h:tr>  
    <h:td>Apples</h:td>  
    <h:td>Bananas</h:td>  
  </h:tr>  
</h:table>
```

```
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

En el ejemplo de arriba, no hay ningún conflicto porque los dos elementos <table> tienen nombres diferentes.

Al usar los prefijos en XML, debe definirse un namespace el cual es llamado para el prefijo. El namespace se define por el atributo del xmlns en la etiqueta de la salida de un elemento. La declaración del namespace tiene la sintaxis siguiente:

```
<root>
<h:table xmlns:h="http://www.w3.org/TR/html4/" >
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.w3schools.com/furniture" >
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

En el ejemplo de arriba, los xmlns atribuyen a la etiqueta <tabla> los prefijos h: y f: con un namespace calificado. Cuando un namespace se define para un elemento, todos los elementos del hijo con el mismo prefijo son asociados con el mismo namespace.

Los Namespaces pueden declararse en los elementos donde ellos se usan o en el elemento XML raíz:

```
<root
xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.w3schools.com/furniture" >

  <h:table>
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
  </h:table>

  <f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>

</root>
```

Nota: El namespace con URL no es acostumbrado. Los Namespaces pueden declararse en los elementos donde ellos se usan o en el XML raíz elemento.

A menudo las compañías acostumbran el namespace como un indicador a una página WEB que contiene la información del namespace.

5.12. Codificación XML

Los documentos XML pueden contener caracteres no ASCII, como æ ø å noruego, francés o E E E. Si carga un documento XML, puede obtener dos errores diferentes que indican problemas de codificación:

Un carácter no válido se encontró en el texto.

Usted recibe este error si el código XML contiene caracteres no ASCII, y el archivo se ha guardado como un solo byte ANSI (o ASCII), sin codificación especificada.

Cambio de codificación actual a la codificación especificada no es compatible.

Usted recibe este error si el archivo XML se ha guardado como Unicode de doble byte (o UTF-16) con una codificación de un solo byte (Windows-1252, ISO-8859-1, UTF-8) especificado. También este error ocurre si el archivo XML se ha guardado con un solo byte ANSI (o ASCII), de doble byte de codificación (UTF-16).

5.13. Windows notepad

Windows Notepad guarda archivos como un solo byte ANSI (ASCII) por defecto. Si se selecciona "Guardar como...", se puede especificar de dos bytes Unicode (UTF-16). Guarde el archivo XML siguiente como Unicode (tenga en cuenta que el documento no contiene ningún atributo de codificación):

```
<?xml version="1.0"?>
<note>
  <from>Jani</from>
  <to>Tove</to>
  <message>Norwegian: æøå. French: èèé</message>
</note>
```

El archivo de arriba, no generará un error. Pero si usted especifica que un solo-byte y lo pone en código. Sin embargo, se acostumbra el namespace como un indicador a una página web que contiene la información del namespace.

5.14. XML en el servidor

XML puede ser fácilmente almacenado y generado por un servidor web estándar. Los archivos XML se pueden almacenar en un servidor de Internet de la misma manera como archivos HTML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <from>Jani</from>
  <to>Tove</to>
  <message>Remember me this weekend</message>
</note>
```

Guarde el archivo en su servidor web con un nombre apropiado, podría usar "note.xml."

Generando XML con PHP.

Para generar una contestación de XML del servidor que usa PHP, puede usar el código siguiente:

```
<?php
header("Content-type: text/xml");
echo "<?xml version='1.0' encoding='ISO-8859-1'?>";
echo "<nota>";
echo "<de>Erika</de>";
```



```
echo "<a>Renieri</a>";  
echo "<texto>Recuerdame este fin de semana</texto>";  
echo "</nota>";  
?>
```

XML puede generarse también desde una base de datos sin instalar ningún software de XML. Para generar una contestación XML desde una Base de datos de un servidor, simplemente escriba el código siguiente y guárdelo como un archivo de ASP en el servidor web:

```
<%  
response.ContentType = "text/xml"  
set conn=Server.CreateObject("ADODB.Connection")  
conn.provider="Microsoft.Jet.OLEDB.4.0;"  
conn.open server.mappath("/db/database.mdb")  
  
sql="select fname,lname from tblGuestBook"  
set rs=Conn.Execute(sql)  
  
response.write("<?xml version='1.0' encoding='ISO-8859-1'?>")  
response.write("<guestbook>")  
while (not rs.EOF)  
response.write("<guest>")  
response.write("<fname>" & rs("fname") & "</fname>")  
response.write("<lname>" & rs("lname") & "</lname>")  
response.write("</guest>")  
rs.MoveNext()  
wend  
  
rs.close()  
conn.close()  
response.write("</guestbook>")  
%>
```

5.15. Islas de datos en XML

¿Qué es? Una isla de datos XML, son datos XML que están incrustados en una página HTML.

¿Por qué evitarlas? Las Islas de datos XML sólo funcionan con los navegadores de Internet Explorer.

¿Qué usar en su lugar? Usted debe usar JavaScript y XML DOM para analizar y mostrar XML en HTML.

En este ejemplo se utiliza el documento XML " [cd_catalog.xml](#) ". Enlazar el documento XML a una etiqueta de <xml> en el documento HTML. El atributo id define un identificador de la isla de datos, y los puntos del atributo src en el archivo XML:

En este ejemplo, sólo funciona en Internet Explorer:

```
<html>  
<body>  
  
<xml id="cdcat" src="cd_catalog.xml"></xml>  
  
<table border="1" datasrc="#cdcat">  
<tr>  
<td><span datafld="ARTIST"></span></td>  
<td><span datafld="TITLE"></span></td>  
</tr>  
</table>  
  
</body>  
</html>
```

5.16. Tecnologías relacionadas con XML

A continuación, se muestra una lista de las tecnologías XML:

XHTML: Una versión mas estricta y limpia basada en XML de HTML.

XML DOM: Un modelo de documento estándar para acceder y manipular XML.

XSL: Consta de tres partes:

- * XSLT: transforma XML en otros formatos, como HTML.

- * XSL-FO: para el formato XML a la pantalla, papel, etc.

- * XPath: Un lenguaje para navegar por los documentos XML.

XQuery: Un lenguaje basado en XML para consultar datos XML.

DTD: Un estándar para la definición de los elementos jurídicos de un documento XML.

XSD: Una alternativa basada en XML para DTD.

XLink: Un lenguaje para crear hipervínculos en documentos xml.

XPointer: Permite a los hipervínculos XLink apuntar a partes especificadas del documento XML.

XForms: Utiliza XML para definir los datos del formulario.

SOAP: Un protocolo basado en XML para permitir que aplicaciones de intercambio de información a través de HTTP.

WSDL: Un lenguaje basado en XML para describir servicios web.

RDF: Un lenguaje basado en XML para describir los recursos web.

RSS: Un formato para la indicación de noticias y el contenido de las noticias-como los sitios.

SMIL: Un lenguaje para describir presentaciones audiovisuales.

SVG Define los gráficos en formato XML.

5.17. XML en la vida real.

XMLNews es una especificación para el intercambio de noticias y otra información. El uso de estos estándares hace que sea más fácil de producir tanto para los productores de noticias y los consumidores de noticias, recibir y archivar cualquier tipo de información de noticias a través de hardwars diferentes, el software y lenguajes de programación.

Un ejemplo XMLNews documento:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nitf>
  <head>
    <title>Colombia Earthquake</title>
  </head>
  <body>
    <headline>
      <hl1>143 Dead in Colombia Earthquake</hl1>
    </headline>
    <byline>
      <bytag>By Jared Kotler, Associated Press Writer</bytag>
    </byline>
    <dateline>
      <location>Bogota, Colombia</location>
      <date>Monday January 25 1999 7:28 ET</date>
    </dateline>
  </body>
</nitf>
```

Un ejemplo de un XML el servicio de tiempo nacional de NOAA (la Administración Oceánica y Atmosférica Nacional):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<current_observation>

<credit>NOAA's National Weather Service</credit>
<credit_URL>http://weather.gov/</credit_URL>
```

```
<image>
  <url>http://weather.gov/images/xml_logo.gif</url>
  <title>NOAA's National Weather Service</title>
  <link>http://weather.gov</link>
</image>

<location>New York/John F. Kennedy Intl Airport, NY</location>
<station_id>KJFK</station_id>
<latitude>40.66</latitude>
<longitude>-73.78</longitude>
<observation_time_rfc822>Mon, 11 Feb 2008 06:51:00 -0500 EST
</observation_time_rfc822>

<weather>A Few Clouds</weather>
<temp_f>11</temp_f>
<temp_c>-12</temp_c>
<relative_humidity>36</relative_humidity>
<wind_dir>West</wind_dir>
<wind_degrees>280</wind_degrees>
<wind_mph>18.4</wind_mph>
<wind_gust_mph>29</wind_gust_mph>
<pressure_mb>1023.6</pressure_mb>
<pressure_in>30.23</pressure_in>
<dewpoint_f>-11</dewpoint_f>
<dewpoint_c>-24</dewpoint_c>
<windchill_f>-7</windchill_f>
<windchill_c>-22</windchill_c>
<visibility_mi>10.00</visibility_mi>

<icon_url_base>http://weather.gov/weather/images/fcicons/</icon_url_base>
<icon_url_name>nfew.jpg</icon_url_name>
<disclaimer_url>http://weather.gov/disclaimer.html</disclaimer_url>
<copyright_url>http://weather.gov/disclaimer.html</copyright_url>

</current_observation>
```

XML está basado en texto

XML es un lenguaje basado en texto marcado. Una gran cosa acerca de XML es que los archivos XML se pueden crear y editar con un simple editor de texto como el Bloc de notas. Sin embargo, cuando empiezas a trabajar con XML, pronto se dará cuenta de que es mejor editar documentos XML con un profesional editor de XML.

¿Por qué no el Bloc de notas?

Muchos desarrolladores web utilizar el Bloc de notas para editar documentos HTML y XML, el Bloc de notas se incluye con el sistema operativo más común y es fácil de usar. Personalmente suelo utilizar el Bloc de notas para la edición rápida de la simple HTML, CSS y XML.

Sin embargo, si utiliza el Bloc de notas para la edición de XML, pronto tendrá problemas.

Bloc de notas no sabe que usted está escribiendo XML, por lo que no será capaz de ayudarlo.

¿Por qué un editor de XML?

Hoy XML es una tecnología importante, y los proyectos de desarrollo utilizan tecnologías basadas en XML como:

- Esquema XML para definir estructuras y tipos de datos XML
- XSLT para transformar datos XML
- SOAP para el intercambio de datos XML entre aplicaciones

- WSDL para describir los servicios web
- RDF para describir los recursos Web
- XPath y XQuery para acceder a datos XML
- SMIL para definir gráficos

Para ser capaz de escribir sin errores los documentos XML, se necesita un editor de XML inteligente!

Los editores XML profesional les ayudarán a escribir sin errores los documentos XML, validar el código XML con una DTD o un esquema, y te obligan a seguir una estructura XML válido.

Un editor de XML debe ser capaz de:

- Agregar etiquetas de cierre de sus etiquetas de apertura automática
- Te obligan a escribir XML válido
- Verifique el código XML con una DTD
- Verifique que su XML contra un esquema
- Código de color la sintaxis XML

En W3Schools hemos estado usando XMLSpy desde hace muchos años. XMLSpy es el editor de XML favorito. Estas son algunas de las características que más nos gustan:

- ☐ Ahora disponible en versiones de 32 bits y 64 bits.
- ☐ Fácil de usar.
- ☐ Entrada sensible al contexto ayudantes.
- ☐ Sintaxis coloreada y la impresión bastante.
- ☐ Construido en la validación y comprobación de la buena formación.
- ☐ Cambiar fácilmente entre la vista de texto y la red.
- ☐ XML editor gráfico de esquemas.
- ☐ Base de datos de importación y exportación para todos los principales bases de datos.
- ☐ SharePoint® Server.
- ☐ Construido en las plantillas para muchos tipos de documentos XML.
- ☐ Creación de gráficos para mostrar datos XML.
- ☐ Inteligente XPath 1.0/2.0 auto-realización.
- ☐ XSLT 1.0/2.0 editor, analizador y depuración.
- ☐ XQuery editor, analizador y depuración.
- ☐ Cliente SOAP y depurador.
- ☐ Gráfica WSDL 1.1/2.0 editor.
- ☐ XBRL validación y edición de taxonomía.
- ☐ Soporte para Office 2007 / OOXML.
- ☐ La generación de código en Java, C + + y C #.

XML se puede utilizar para intercambiar, compartir y almacenar datos.

Los documentos XML forman [una estructura de árbol](#) que empieza en la "raíz" y termina en "las hojas".

XML tiene muy simples [reglas de sintaxis](#). XML con la sintaxis correcta es "muy útil".

[XSLT](#) se utiliza para transformar XML en otros formatos como HTML.

Todos los navegadores modernos tienen incorporado un [analizador de XML](#) que pueden leer y manipular XML.

El [DOM](#) (Document Object Model) define una manera estándar para acceder a XML.

El [XMLHttpRequest](#) objeto proporciona una manera de comunicarse con un servidor después que una página web se ha cargado.

[Los espacios de nombres XML](#) proporcionan un método para evitar conflictos de nombre de elemento.

El texto dentro de un [CDATA](#) sección es ignorada por el intérprete.

Ejemplo práctico de XML con PHP y Ajax.

En este ejemplo, usted hará uso de XML y JSON con PHP, por lo que puede crear el código en el servidor que se comunica con su cliente JavaScript.

El documento XML que usted creará será una versión simplificada de los documentos XML que usted ha aprendido en este capítulo. Para refrescar su memoria, este es el documento XML que usaremos:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response>
  <books>
    <book>
      <title>
        AJAX and PHP: Construyendo una moderna aplicación Web, 2nd Ed.
      </title>
      <isbn>
        978-1904817726
      </isbn>
    </book>
  </books>
</response>
```

PHP, XML y AJAX del lado del servidor.

1. En la carpeta de navegación de su servidor crear una carpeta llamada xml.
2. En la carpeta php, crear un archivo llamado phptest.html, y agregar el siguiente código:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.
w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>
    Fundamentos de AJAX: Usando DOM y PHP para crear un archivo XML.
  </title>
  <script type="text/javascript" src="phptest.js"></script>
</head>
<body onload="process()">
  <p>El libro de AJAX del 2010 es:</p>
  <div id="myDivElement" />
</body>
</html>
```

3. El código del lado del cliente lo crearemos dentro del archivo phptest.js.

```
// retener una instancia de XMLHttpRequest
var xmlHttp = createXmlHttpRequestObject();
// crear una instancia de XMLHttpRequest
function createXmlHttpRequestObject()
{
  // guarde la referencia del objeto XMLHttpRequest
  var xmlHttp;
  // crear el objeto XMLHttpRequest
  try
  {
    // asume que Internet Explorer 7 o un buscador mas moderno
    xmlHttp = new XMLHttpRequest();
  }
  catch(e)
  {
    // asume que Internet Explorer 6 o mas viejos
    try
```

```
{
xmlHttp = new ActiveXObject("Microsoft.XMLHttp");
}
catch(e) { }
}
// retorna el objeto creado o muestra un mensaje de error
if (!xmlHttp)
alert("Error creando el objeto XMLHttpRequest.");
else
return xmlHttp;
}
// leer un archivo desde el servidor
function process()
{
// continúe únicamente si xmlHttp no está vacío
if (xmlHttp)
{
// probar conectarse con el servidor
try
{
// iniciando la lectura de un archivo desde el servidor
xmlHttp.open("GET", "phptest.php", true);
xmlHttp.onreadystatechange = handleRequestStateChange;
xmlHttp.send(null);
}
// desplegar un mensaje de error en caso de fallo
catch (e)
{
alert("No se puede conectar con el servidor:\n" + e.toString());
}
}
}
// esa función es llamada cuando el estado de la petición HTTP cambia
function handleRequestStateChange()
{
// ... Toma el código
// cuando el estado cambia a 4, nosotros podemos leer la respuesta del servidor
if (xmlHttp.readyState == 4)
{
// continúe únicamente si el estado de HTTP es "OK"
if (xmlHttp.status == 200)
{
try
{
// Hacer algo con la contestación del servidor
handleServerResponse();
}
catch(e)
{
// desplegar un mensaje de error
alert("Error leyendo la respuesta: " + e.toString());
}
}
else
{
// despliega el estado del mensaje
alert("Había un problema recuperando los datos:\n" + xmlHttp.statusText);
}
}
}
// manejando la contestación recibida desde el servidor
function handleServerResponse()
{
// leer el mensaje desde el servidor.
```

```
var xmlResponse = xmlHttp.responseXML;
// obtener el element del document XML
xmlRoot = xmlResponse.documentElement;
// obtener el arreglo con titulos llamados book y ISBNs
titleArray = xmlRoot.getElementsByTagName("title");
isbnArray = xmlRoot.getElementsByTagName("isbn");
// generar la salida HTML
var html = "";
// iterar a través del arreglo y crear una estructura HTML
for (var i=0; i<titleArray.length; i++)
html += titleArray.item(i).firstChild.data + ", " + isbnArray.item(i).firstChild.data + "<br/>";
// obtener una referencia del element <div> en la página
myDiv = document.getElementById("myDivElement");
// desplegar la salida en HTML
myDiv.innerHTML = "<p>El servidor dice: </p>" + html;
}
```

4. Finalmente el archivo phptest.php

```
<?php
// fijar la salida en un contenido de tipo xml
header('Content-Type: text/xml');
// crear un nuevo documento XML
$dom = new DOMDocument();
// crear un elemento raíz (root <response>)
$response = $dom->createElement('response');
$dom->appendChild($response);
// crear el elemento <books> y añadirlo como un hijo de <response>
$books = $dom->createElement('books');
$response->appendChild($books);
// crear el elemento titulo para book
$title = $dom->createElement('title');
$titleText = $dom->createTextNode
('AJAX y PHP: Construyendo una moderna aplicacion Web, 2nd Ed');
$title->appendChild($titleText);
// crear el elemento isbn para book
$isbn = $dom->createElement('isbn');
$isbnText = $dom->createTextNode('978-1904817726');
$isbn->appendChild($isbnText);
// crear el elemento <book>
$book = $dom->createElement('book');
$book->appendChild($title);
$book->appendChild($isbn);
// añadir <book> como un hijo de <books>
$books->appendChild($book);
// construir la estructura XML en una variable de tipo cadena
$xmlString = $dom->saveXML();
// salida de la cadena XML
echo $xmlString;
?>
```

5. Primero vamos a hacer una simple prueba de la página phptest.php, la cual debe retornar una estructura xml cuando carguemos la página <http://localhost/xml/phptest.php> en nuestro navegador, la salida debe ser así:

Ahora que hemos obtenido un buen funcionamiento de la página phptest.php, cargaremos desde nuestro navegador web <http://localhost/XML/phptest.html>, la salida debe ser así:

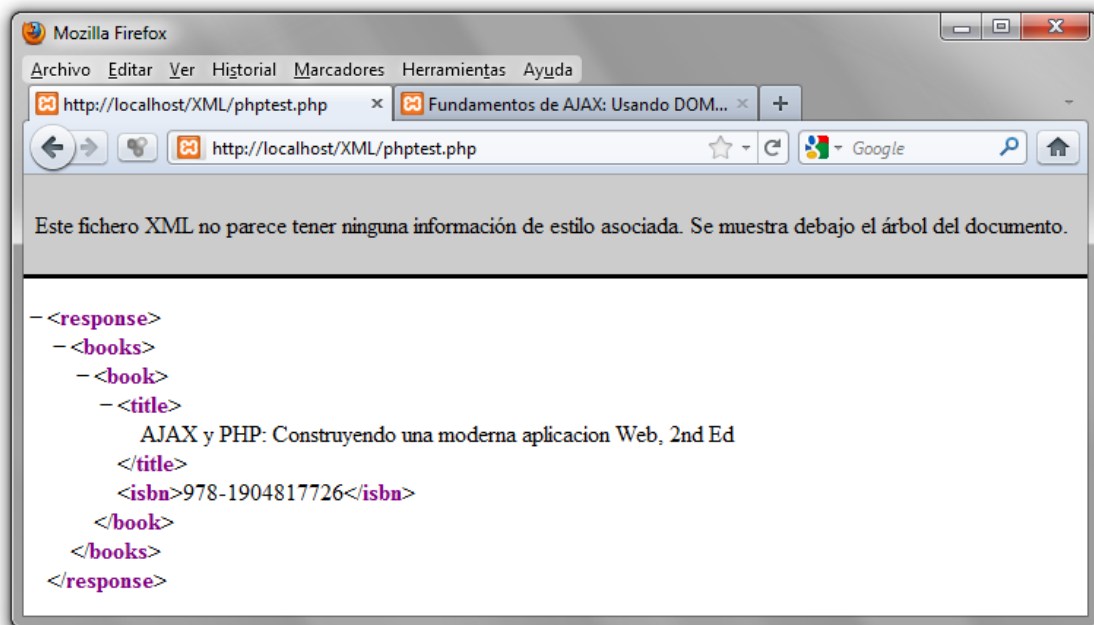


Ilustración 5.1 Estructura XML simple, generada por PHP.



Ilustración 5.2 Ajax con PHP.

Explicación

Cuando se trata de generar estructuras XML, no sólo en el lado del cliente, sino en el del servidor, usted tiene que elegir entre crear el documento XML utilizando DOM, o uniéndose a las cadenas. Su script PHP, phptest.php, comienza estableciendo el contenido de la salida a text / xml:

```
<?php
// poner la salida del contenido en tipo texto xml
header('Content-Type: text/xml');
```

Si desea una mayor familiaridad con las cabeceras de PHP, puede encontrar mas documentación en: <http://www.php.net/manual/en/function.header.php>.

El uso de DOM con PHP, no es muy sorprendente, se parece mucho al DOM de JavaScript. Todo comienza por la creación de un objeto de documento DOM, que en PHP es representado por la clase DOMDocument:

```
// create the new XML document
```



```
$dom = new DOMDocument();
```

A continuación, seguir creando la estructura XML utilizando métodos como los métodos `createElement()`, `createTextNode()`, `appendChild()`, y así sucesivamente:

```
// create the root <response> element
$response = $dom->createElement('response');
$dom->appendChild($response);
// create the <books> element and append it as a child of <response>
$books = $dom->createElement('books');
$response->appendChild($books);
```

Al final, se guarda toda la estructura XML como una cadena, con la función `saveXML()`, y el eco de la cadena a la salida.

```
$xmlString = $dom->saveXML();
// output the XML string
echo $xmlString;
?>
```

El documento XML se lee y se muestra en el lado del cliente utilizando técnicas que ya hemos explicado en este capítulo en la sección 5.10 donde se habló del objeto `XMLHTTPREQUEST`.

En los ejemplos de este capítulo, usted va a generar documentos XML en el servidor, y los lee el cliente, pero por supuesto que usted puede hacer todo lo contrario. En este capítulo, vimos cómo crear documentos XML y en los capítulos 2 y 3 la creación de elementos DOM con JavaScript. A continuación, puede pasar estas estructuras para PHP (mediante GET o POST). Para leer las estructuras XML desde PHP también se puede usar el DOM, o una API fácil de usar llamada SimpleXML.

Bibliografía del Capítulo 5

www.w3school.com



.....



Capítulo 6

AJAX

6 AJAX

6.1. Análisis Detallado

Todas las aplicaciones realizadas con técnicas de AJAX deben instanciar en primer lugar el objeto XMLHttpRequest, que es el objeto clave que permite realizar comunicaciones con el servidor en segundo plano, sin necesidad de recargar las páginas. La implementación del objeto XMLHttpRequest depende de cada navegador, por lo que es necesario emplear una discriminación sencilla en función del navegador en el que se está ejecutando el código.

Los navegadores que siguen los estándares (Firefox, Safari, Opera, Internet Explorer 7 y 8) implementan el objeto XMLHttpRequest de forma nativa, por lo que se puede obtener a través del objeto Windows. Una vez obtenida la instancia del objeto XMLHttpRequest, se prepara la función que se encarga de procesar la respuesta del servidor. La propiedad onreadystatechange del objeto XMLHttpRequest permite indicar esta función directamente incluyendo su código mediante una función anónima o indicando una referencia a una función independiente.

Uno de los usos más populares de Ajax es crear búsquedas en vivo, como ya ha visto un ejemplo cuando usa Google. Con una búsqueda en vivo, el usuario puede introducir un término de búsqueda parcial, y usando Ajax, la aplicación Web se conecta al servidor y encuentra coincidencias con ese término de búsqueda parcial.

Hay un montón de búsquedas en línea en vivo, además de Google. Por ejemplo, eche un vistazo a Gollum en <http://gollum.easycp.de/en/>, que es una búsqueda en directo de Wikipedia, la línea de enciclopedia libre en www.wikipedia.org.

De las diferencias visibles, tal vez la más obvia es la solicitud de página. Con el fin de hacer que algo suceda en una aplicación web, una llamada tiene que ser hecho al servidor. Para ello, la página debe ser restaurada para recuperar la información actualizada desde el servidor hacia el cliente (normalmente un navegador web como Firefox o Internet Explorer). Esto no es una responsabilidad específica del navegador, sino que la petición HTTP / protocolo de respuesta inherente a todos los navegadores web (ver Ilustración) está diseñado para funcionar de esta manera; Si bien teóricamente este funciona bien, los desarrolladores han comenzado a pedir un enfoque más transparente, para que sus tiempos de respuesta de las aplicaciones se puedan parecer más a las aplicaciones de escritorio.

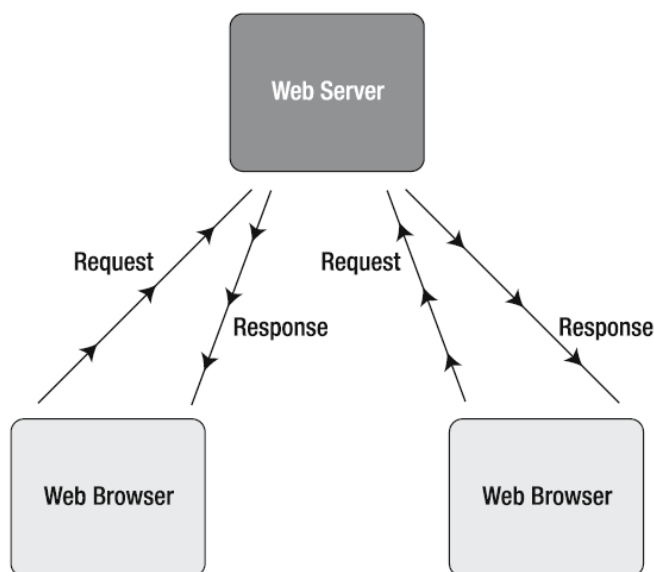


Ilustración 6.1 El método de solicitud / respuesta utilizado en la mayoría de los sitios web que existen en Internet. Con el fin de empezar a hacer uso de Ajax y PHP para crear aplicaciones Web, es necesario obtener primero un firme entendimiento de los conceptos básicos. Cabe señalar que el Ajax es una herramienta de JavaScript, y así



aprender los conceptos básicos de JavaScript será muy importante cuando se trate de entender las aplicaciones tipo Ajax. Comencemos con lo básico.

6.2. Desde el CGI a Flash para DHTML

La comunidad de desarrollo ha pedido, y las empresas han respondido. Herramientas tras herramientas para diseñadores han sido creadas, cada una con su propio conjunto de ventajas y desventajas. Tal vez el primer lenguaje de scripting para permitir que las aplicaciones web verdaderamente tuvieran libertad que pedían en el lado del servidor de procesamiento del lenguaje CGI (Common Gateway Interface).

Con la llegada del CGI, los desarrolladores ahora pueden realizar acciones complejas, pero no se limita a la creación de imágenes dinámicas, gestión de base de datos, cálculo complejos, y la creación dinámica de contenidos web. Lo que hemos llegado a esperar de nuestras aplicaciones web de hoy comenzó con CGI. Desafortunadamente, mientras CGI abordado muchas cuestiones, el problema difícil de interacción sin fisuras y la respuesta se mantuvo.

En un intento de crear condiciones reales de vida, la respiración, el movimiento de los contenidos web de Macromedia (www.macromedia.com) dio a conocer su gran funcionalidad, e impresionante lugar (por el momento) Suite Flash. Flash fue, y sigue siendo hasta hoy en día llamado muy acertadamente. Permite a un desarrollador web crear visualmente impresionantes "películas" que pueden funcionar como sitios web, aplicaciones, y mucho más. Estos sitios web se consideraron significativamente "llamativos" en los sitios web, debido a su capacidad de tener movimientos prestados en todo el navegador.

En las manos de un diseñador profesional, sitios web habilitados con Flash pueden ser muy impresionantemente visuales. Del mismo modo, en manos de un desarrollador profesional, que puede ser muy poderoso.

Sin embargo, es raro que una persona posea un considerable tanto de habilidades en el diseño y desarrollo, por lo tanto, las aplicaciones de Flash tienden a ser visualmente impresionantes, con muy poca funcionalidad, o funcionalmente increíble con una interfaz que deja mucho que desear. Además, este dilema se combina con un problema de compatibilidad adicional: para que Flash pueda funcionar, un plug-in debe estar instalado en su navegador.

Otra tecnología visual dinámica que ha existido por muchos años, pero no tienen una base importante de usuarios es DHTML (un acrónimo de hipertexto dinámico Markup Language). DHTML-un término que describe el matrimonio de JavaScript y HTML básicamente combina elementos de HTML y CSS con JavaScript en un intento de hacer que las cosas sucedan en su navegador web de forma dinámica. Mientras DHTML en las manos de un experto JavaScript profesionales pueden lograr resultados impresionantes, el nivel de conocimientos técnicos necesarios para ello tiende a mantener fuera del alcance de la mayoría de la comunidad de desarrollo.

Mientras los scripts, como menús desplegables, rollovers y punta de la herramienta del pop-ups son bastante comunes, es sólo por personas calificadas que la creación de paquetes de todos los días pueden implementarse por desarrolladores. Muy pocos individuos aplican códigos de estos paquetes de software desde cero, y hasta hace poco, no muchas personas consideran JavaScript en una herramienta muy potente para Internet.

Ajax tomó el mundo de Internet y no por sorpresa, no sólo en su facilidad de uso y muy fresca funcionalidad, sino también en su capacidad para llamar la atención de casi todos los desarrolladores del planeta. Cuando hace dos años, el Ajax se llevó a cabo más dubitativo, sin ningún tipo de forma de la norma (y, ciertamente, había muy pocos sitios que construyeron su núcleo alrededor del Ajax completamente), el Ajax es ahora aparentemente tan común como el vuelco del vehículo.

Toda las aplicaciones web están surgiendo de la nada, completamente basado en funcionalidad Ajax. No sólo son los usos más ingeniosos de la tecnología, son líderes de la industria de la web en una nueva era por que el navegador web estándar puede llegar a ser tanto más, incluso puede rivalizar con la aplicación de escritorio ahora. Tomemos, por ejemplo, Flickr (www.flickr.com) o Gmail (www.gmail.com). En su superficie, ofrecen servicios que no son realmente nada nuevos. (Después de todo, ¿cuántos Álbumes de fotos en línea y servicios de correo web están ahí fuera?) ¿Por qué entonces estas dos aplicaciones ganaron tanta prensa y publicidad, en particular en la comunidad en línea?

La razón para la nueva popularidad de las aplicaciones basadas en Ajax no es que su funcionalidad contenida dentro sea nueva ni sorprendente, sino simplemente el hecho de la forma en que se presentó la información, la funcionalidad para nosotros se hace de una manera muy eficiente y la forma ergonómica.

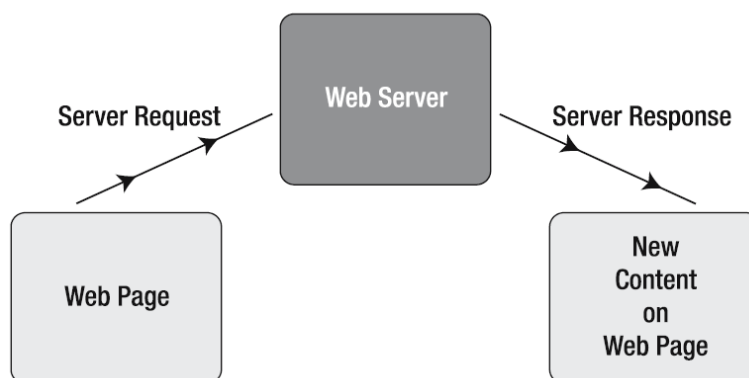
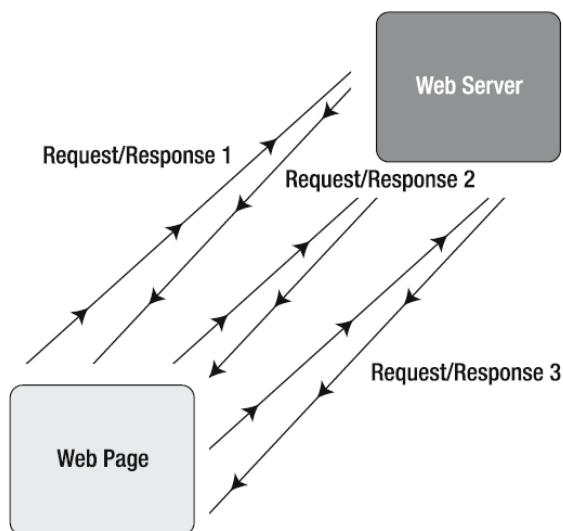


Ilustración 6.2 Solicitud de petición / respuesta usando el modelo asíncrono de Ajax; multiples peticiones al servidor pueden hacerse de una página sin tener que recargarla o actualizarla.

Ilustración 6.3 petición del servidor tradicional modelo / respuesta utilizado en la mayoría de aplicaciones basadas en web hoy en día, cada vez que se hizo una petición del servidor, la página debe cargar para revelar nuevos contenidos.



6.3. Solicitudes y respuestas fundamentales HTTP.

Para entender exactamente los conceptos de Ajax, es importante saber cómo un sitio web, procesa una solicitud y recibe una respuesta de un servidor web. La norma actual que los navegadores utilizan para obtener información de un servidor web es el método HTTP. Este es el medio que un navegador web utiliza para enviar una solicitud de un sitio web y recibir a continuación una respuesta desde el servidor web, que está actualmente a cargo de devolver la respuesta.

Las peticiones HTTP dan tanto trabajo como el correo electrónico. Es decir que cuando se envía una solicitud, las cabeceras de algunos se pasan a lo largo del servidor web para saber exactamente lo que se va a servir y cómo manejar la petición. Aunque la mayoría de las cabeceras son opcionales, hay un encabezado que es absolutamente necesario: el encabezado de host. Este encabezado es crucial ya que permite que el servidor sepa qué servir.

Una vez que una solicitud ha sido recibida, el servidor decide entonces cuál es la respuesta para regresar. Hay muchos códigos de respuesta diferentes. La tabla 6.1 tiene una lista de algunos de los más comunes.

Código	Descripción
200 Ok	Este código de respuesta se devuelve si el documento o archivo en cuestión se encuentran y se desempeñó correctamente.
304 Not Modified	Este código de respuesta se devuelve si un navegador ha indicado que hay una copia local en caché, y la copia del servidor no ha cambiado desde esta copia en caché.
401 Unauthorized	Este código genera una respuesta si la petición de que se trate requiere autorización para tener acceso al documento solicitado.
403 Forbidden	Este código de respuesta se devuelve si el documento solicitado no tiene los permisos adecuados para tener acceso a la solicitante.
404 Not Found	Este código de respuesta se envía de nuevo si el archivo que está intentando ser accesado no se pudo encontrar (por ejemplo, si no existe).
500 International Server Error	Este código le será devuelto si el servidor que se está en contacto tiene un problema.
503 Service Unavailable	Este código de respuesta se genera si el servidor está demasiado abrumado para atender la solicitud.

Tabla 6.1 Códigos comunes de respuesta HTTP.

Cabe señalar que hay varias formas de métodos de solicitud disponible. Algunos de ellos, como GET y POST, probablemente sonarán muy familiar. Tabla 6.2 enumera los métodos disponibles de petición (aunque por lo general sólo los métodos GET y POST se utilizan).

Método	Descripción
GET	El medio más común de envío de una solicitud; pide simplemente una especificación de los recursos del servidor.
HEAD	Similar a una solicitud GET, salvo que la respuesta vendrá de nuevo sin el cuerpo de la respuesta, útil para la recuperación de cabeceras.
POST	Permite enviar una solicitud a lo largo de los datos presentados por el usuario (ideal para los formularios basados en la web).
PUT	Transfiere una versión de la petición del archivo en cuestión.
DELETE	Envía una solicitud para eliminar el documento especificado.
TRACE	Envía una copia de la solicitud con el fin de monitorear su progreso.
OPTIONS	Devuelve una lista completa de los métodos disponibles, útil para el control de lo que los métodos de un servidor admite.
CONNECT	Una petición basada en proxy utilizado para hacer un túnel SSL.

Tabla 6.2 Métodos solicitud HTTP.

Ahora que usted tiene una comprensión básica de cómo se envía una petición desde un navegador a un servidor y, a continuación tiene una respuesta enviada de vuelta, será más fácil de entender cómo el Objeto XMLHttpRequest trabaja. En realidad, es bastante similar, pero funciona en segundo plano sin la actualización de la página de requisitos previos.

6.4. El objeto XMLHttpRequest

Ajax es en realidad un concepto utilizado para describir la interacción del cliente Objeto XMLHttpRequest con scripts basados en servidor. Con el fin de hacer una petición al servidor a través de Ajax, un objeto debe ser creado para ser utilizado para diferentes formas de funcionalidad. Cabe señalar que el objeto XMLHttpRequest es una instancia tanto para manejo de una o diferentes gamas de navegadores. De particular interés es que Microsoft Internet Explorer crea el objeto como un control ActiveX, mientras que navegadores como Firefox y Safari utiliza un objeto a base de JavaScript. Esto es algo crucial en la ejecución de código entre navegadores ya que es imprescindible para poder ejecutar el Ajax en cualquier tipo de configuración del navegador.

Una vez que una instancia del objeto XMLHttpRequest se ha creado, hay una serie de métodos disponibles para el usuario. Estos métodos se ampliaron con más detalle en Tabla 6.3. Dependiendo de cómo desee utilizar el objeto, existen métodos diferentes pueden llegar a ser importantes que otros.

Metodos	Descripción
abort()	Cancela la solicitud actual.
getAllResponseHeaders()	Devuelve todas las cabeceras HTTP como una variable de tipo String.
getResponseHeader()	Devuelve el valor del encabezado HTTP especificado en el método.
open()	Especifica los diferentes atributos necesarios para establecer una conexión con el servidor, le permite hacer selecciones tales como GET o POST, si desea conectarse de forma asincrónica, y la URL para conectarse a.
setRequestHeader()	Agrega una etiqueta / valor a la cabecera cuando se envía.
send()	Envía la solicitud actual.

Tabla 6.3 Métodos del objeto XMLHttpRequest.

Propiedades de XMLHttpRequest: Por supuesto, cualquier objeto tiene un conjunto completo de propiedades que pueden ser usados y manipulados con el fin de que funcione al máximo. Una lista completa de las propiedades objeto XMLHttpRequest se presenta en la Tabla 6.4. Es importante tomar nota de estas propiedades, se va haciendo uso de ellos a medida que usted avance en su funcionalidad.

onreadystatechange	Se utiliza como un controlador de eventos para los eventos que se disparan en los cambios de estado.
readyState	Contiene el estado actual del objeto (0: sin inicializar, 1: cargando, 2: cargado, 3: interactiva, 4: completa)
responseText	Devuelve la respuesta en formato de cadena.
responseXML	Devuelve la respuesta adecuada en el formato XML.
Status	Devuelve el estado de la solicitud en formato numérico.
statusText	Devuelve el estado de la solicitud, pero en formato de cadena.

Tabla 6.4 Propiedades del objeto XMLHttpRequest.

Aunque en el momento de escribir esto, Internet Explorer de Microsoft sigue dominando el mercado de los navegadores, los competidores como Firefox han realizado avances significativos. Por lo tanto, es más importante que nunca para asegurarse de que las aplicaciones Ajax crossbrowser son compatibles. Uno de los aspectos más importantes de la funcionalidad de Ajax es que se puede implementar a través de navegadores sin problemas, con sólo una pequeña cantidad de trabajo necesario para que funcione en la mayoría de los navegadores (la excepción es más bien versiones antiguas de los navegadores actuales).

Considere el siguiente fragmento de código, que crea una instancia del objeto XMLHttpRequest, y funciona en cualquier navegador que soporte XMLHttpRequest. En la Ilustración 6.4 muestra la diferencia del resultado entre el Internet Explorer y otros navegadores.

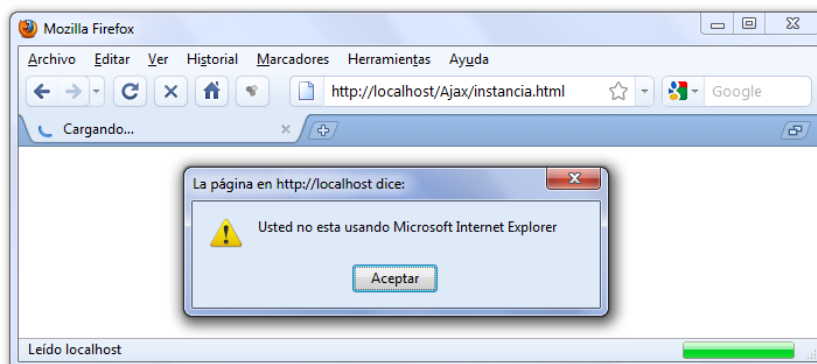
```
<script type="text/javascript">
    // Crear una variable booleana para comprobar si hay una instancia válida de Internet Explorer.
    xmlhttp = false;
    //Comprueba si está utilizando Internet Explorer.
    try {
        //Si la versión de Javascript es superior a 5.
        xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
        alert ("Usted est'a usando Microsoft Internet Explorer.");
    } catch (e) {

        //Si no, entonces utilizar otro objeto Active X.
        try {
            //Si nosotros estamos usando Internet Explorer.
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
            alert ("Usted esta usando Microsoft Internet Explorer");
        } catch (E) {
            //También nosotros debemos utilizar un navegador que no sea Internet Explorer.
            xmlhttp = false;
        }
    }
    //Si nosotros no estamos usando Internet Explorer, crea una instancia del objeto javascript.
    if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
        xmlhttp = new XMLHttpRequest();
        alert ("Usted no esta usando Microsoft Internet Explorer");
    }
</script>
```

Como puede ver, el proceso de crear un objeto XMLHttpRequest pueden diferir, pero el resultado final es siempre el mismo, tienes un medio para crear un objeto XMLHttpRequest utilizable. Microsoft se convierte en un poco más complicado en este sentido que la mayoría de los demás navegadores.

Ilustración 6.4 Identificando que navegador estoy usando.

El flujo de este ejemplo de código en particular es bastante simple. Básicamente, se comprueba si el usuario está utilizando una versión más reciente de Internet Explorer (tratando de crear el objeto ActiveX). Si se determina que ninguno de ellos es el caso, entonces el usuario debe estar utilizando un navegador que no sea Internet Explorer, y el objeto XMLHttpRequest estándar.



Ahora bien, es importante tener en cuenta que este método para iniciar un objeto XMLHttpRequest no es la única manera de hacerlo. El fragmento de código siguiente hará en gran medida la misma cosa, pero es bastante más simple:

```
<script type="text/javascript">
```

```
var xmlhttp;
//If, the activexobject is available, we must be using IE.
if (window.ActiveXObject){
xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
} else {
//Else, we can use the native Javascript handler.
xmlhttp = new XMLHttpRequest();
}
</script>
```

Como puede ver, este caso es una manera que utiliza mucho menos código para invocar el Objeto XMLHttpRequest.

6.5. Interacción con el servidor

6.5.1. Enviando una petición al Servidor.

Ahora que tienes tu XMLHttpRequest brillante, nuevo y listo para su uso, el siguiente paso es utilizarlo en una petición al servidor. Esto se puede hacer de diferentes maneras, pero el aspecto clave a recordar es que usted debe validar una respuesta adecuada, y debe decidir si desea utilizar el método GET o POST para hacerlo. Cabe señalar que si utiliza Ajax para recuperar información desde el servidor, el método GET es probablemente la forma mas apropiada.

Si va a enviar información al servidor, POST es la mejor manera de manejar esto, tenga en cuenta que GET no es recomendable para enviar información debido a sus limitaciones inherentes al tamaño.

Con el fin de hacer una petición al servidor, es necesario confirmar algunas funcionalidades basicas. En primer lugar, debe decidir a qué página (o script) desea conectarse y, a continuación en qué área de carga de la página o script. Considere la siguiente función, que recibe como argumentos la página (o script) que desea cargar y el div (o el otro objeto) en el que desea cargar el contenido.

```
function makerequest(serverPage, objID) {
var obj = document.getElementById(objID);
xmlhttp.open("GET", serverPage);
xmlhttp.onreadystatechange = function() {
if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
obj.innerHTML = xmlhttp.responseText;
}
}
xmlhttp.send(null);
}
```

Básicamente, el código aquí está en la identificación de los elementos HTML y la página del servidor. A continuación, los intentos de abrir una conexión a la página del servidor usando el método open() del Objeto XMLHttpRequest. Si la propiedad readyState devuelve un 4 (completa) y el código la propiedad devuelve un estado 200 (correcto), entonces usted puede cargar la respuesta de la página solicitada (o script) en el elemento innerHTML del pasado-en el objeto que después de enviar la solicitud.

Básicamente, lo que se lleva a cabo aquí es un medio para crear un nuevo objeto XMLHttpRequest y luego usarlo para disparar una secuencia de comandos o una página y lo cargan en el elemento apropiado en la página. Ahora usted puede comenzar a pensar en nuevas y emocionantes maneras de utilizar este muy sencillo concepto.

6.5.2. Ejemplo Básico utilizando Ajax con PHP:

Uno de los usos más espectaculares de la funcionalidad de Ajax es ocultar y mostrar contenidos en respuesta a los clics del usuario (o cuando coloca el puntero sobre un elemento, o cuando hace clic sobre un botón). Esta funcionalidad nos permite dar acceso a cantidades ingentes de información sin saturar inicialmente la página.

Al ocultar los contenidos en enlaces de menús que se pueden expandir y contraer, podemos incluir mucha información en muy poco espacio. Para nuestro primer ejemplo, utilizaremos Ajax para expandir y contraer un calendario a base de clics. Al aplicar Ajax para ocultar y mostrar información y PHP para generar dinámicamente el calendario del mes actual, podremos reutilizar cómodamente este código en cualquier aplicación sin ocupar demasiado espacio en la página.

Esta es una página web básica:

```
<!DOCTYPE html>
<html>

<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Primer ejemplo con Ajax y PHP</title>
<script type="text/javascript" src="functions.js"></script>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>

<body>
  <div id="createtask" class="formclass"></div>
  <div id="autocompletediv" class="autocomp"></div>
  <p><a href="javascript:/" onclick="showHideCalendar()">
    </a><a
    href="javascript:/" onclick="showHideCalendar()">Mi Calendario</a></p>
  <div id="calendar" style="width:105px; text-align: left"></div>
</body>

</html>
```

```
//functions.js

//Crear una variable de tipo Bool para comprobar si se esta usando IE.
var xmlhttp=false;

//Comprobar si se está usando IE.
try {
  //Si la version de JavaScript es superior a la 5.
  xmlhttp = new ActiveXObject ("Msxml2.XMLHTTP");
} catch(e){
  //Si no, utilizamos el tradicional objeto ActiveX.
  try{
    //Si estamos usando IE.
    xmlhttp = new ActiveXObject ("Microsoft.XMLHTTP");
  } catch(E){
    //En caso contrario no debemos estar usando IE.
    xmlhttp = false;
  }
}

//Si no estamos usando IE, creamos la instancia JavaScript del objeto.

if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
  xmlhttp = new XMLHttpRequest ();
}

//Variable para distinguir si se debe abrir o cerrar el calendario.
var showCalendar = true;

function showHideCalendar(){

  //Ubicación en la que se carga la página.
  var objID="calendar";
```

```
//Cambiar el signo menos por el más.
if (showCalendar == true){
    //Mostrar el calendario.
    document.getElementById("opencloseimg").src = "images/mins.gif";
    //Nosotros estamos cargando la página.
    var serverPage = "calendar.php";
    //Poner la variable open close
    showCalendar = false;

    var obj = document.getElementById(objID);
    xmlhttp.open("GET", serverPage);
    xmlhttp.onreadystatechange = function(){
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200){
            obj.innerHTML = xmlhttp.responseText;
        }
    }
    xmlhttp.send(null);
}else{
    //Ocultar el calendario
    document.getElementById("opencloseimg").src = "images/plus.gif";
    showCalendar = true;
    document.getElementById(objID).innerHTML = "";
}
}
```

Hasta aquí todo es muy sencillo ¿verdad? Lo más importante es el código JavaScript que contiene el archivo functions.js. En él se crea una función llamada showHideCalendar, que mostrará u ocultará el módulo del calendario en función del valor de la variable showCalendar. Si showCalendar contiene el valor true, se realiza una llamada Ajax al servidor para obtener el calendario del script calendar.php. El resultado de este script se muestra dentro del elemento calendar de la página. Por supuesto, puede introducir las modificaciones que desee para mostrarlo en cualquier otro elemento. El script también cambia el estado de la imagen menos/más para ofrecer una verdadera funcionalidad abrir/cerrar.

Una vez realizada la llamada al servidor, el script PHP utiliza la funcionalidad del lado del servidor para generar el calendario del mes actual. Analice a continuación el código siguiente:

```
<?php
//calenar.php
//Comprobar si existen los valores del mes y anyo
if ((!$_GET['month']) && (!$_GET['year'])){
    $month = date("n");
    $year = date("Y");
}else{
    $month = $_GET['month'];
    $year = $_GET['year'];
}

//Calcular el mes
$timestamp = mktime (0,0,0, $month, 1, $year);
$monthname = date ("F", $timestamp);

//Ahora se crea la tabla apropiada para el mes
?>

<table style="width: 105px; border-collapse: collapse;" border="1" cellpadding="3" cellspacing="0" bordercolor="#000000">
<tr>
<td colspan="7" class="calendartodayoff" onmouseover="this.className='calendartodayover'"
onmouseout="this.className='calendartodayoff'">
<span style="font-weight:bold;"><?php echo $monthname . " " . $year; ?>
</span>
```

```
</td>"
</tr>
<tr>
<td class="calendartodayoff" onmouseover="this.className='calendartodayover'"
onmouseout="this.className='calendartodayoff'">
    <span style="font-weight:bold;">Do</span>
</td>
    <td class="calendartodayoff" onmouseover="this.className='calendartodayover'"
onmouseout="this.className='calendartodayoff'">
        <span style="font-weight: bold;">Lu</span>
    </td>
    <td class="calendartodayoff" onmouseover="this.className='calendartodayover'"
onmouseout="this.className='calendartodayoff'">
        <span style="font-weight: bold;">Ma</span>
    </td>
    <td class="calendartodayoff" onmouseover="this.className='calendartodayover'"
onmouseout="this.className='calendartodayoff'">
        <span style="font-weight: bold;">Mi</span>
    </td>
    <td class="calendartodayoff" onmouseover="this.className='calendartodayover'"
onmouseout="this.className='calendartodayoff'">
        <span style="font-weight:bold;">Ju</span>
    </td>
    <td class="calendartodayoff" onmouseover="this.className='calendartodayover'"
onmouseout="this.className='calendartodayoff'">
        <span style="font-weight:bold;">Vi</span>
    </td>
    <td class="calendartodayoff" onmouseover="this.className='calendartodayover'"
onmouseout="this.className='calendartodayoff'">
        <span style="font-weight:bold;">Sa</span>
    </td>
</tr>

<?php
    $monthstart = date ("w", $timestamp);
    //$lastday = date ("d", mktime (0,0,0, $month + 1,0, $year));
    $lastday = date ("d", mktime (0, 0, 0, $month + 1, 0, $year));
    $startdate = -$monthstart;

    //Discernir cuantas filas se necesitan
    $numrows = ceil (((date("t",mktime (0,0,0, $month +1, 0, $year))+$monthstart) / 7));

    //Se crea el numero de filas apropiado...
    for($k = 1; $k <= $numrows; $k++){
        ?><tr><?php
            //Se utilizan 7 columnas (para los 7 días)...
            for ($i = 0; $i < 7; $i++){
                $startdate++;
                if(($startdate <= 0) || ($startdate > $lastday)){
                    //Si existe un dia en blanco en el calendario.
                    ?><td style="background: #FFFFFF;">&nbsp;</td><?php
                }else{
                    if($startdate == date("j") && $month == date("n") && $year ==
date("Y")){
                        ?><td style="text-align: center; background: #FFBC37;"
onmouseover="this.style.background='#FECE6E'"
onmouseout="this.style.background='#FFBC37'"><?php echo
date ("j"); ?>
                    </td><?php
                    }else{
                        ?><td style="text-align: center;background: #A2FABA;"
onmouseover="this.style.background='#CAD7F9'"
onmouseout="this.style.background='#A2BAFA'">
                        <?php echo $startdate; ?></td><?php
```



```
    top: 0px;
    visibility: hidden;
    height: 0px;
    width: 0px;
    background: #A2BAFA;
    border-style: solid;
    border-width: 1px;
    border-color: #000000;
}

.autocomp {
    position: absolute;
    left: 0px;
    top: 0px;
    visibility: hidden;
    width: 0px;
}

.taskboxclass{
    position: absolute;
    left: 0px;
    top: 0px;
    visibility: hidden;
    width: 0px;
}

.calendarover{
    text-align: center;
    background: #CAD7F9;
    width: 15px;
}

.calendaroff{
    text-align: center;
    background: #A2BAFA;
    width: 15px;
}

.calendartodayover{
    text-align: center;
    background: #FECE6E;
    width: 15px;
}

.calendartodayoff{
    text-align: center;
    background: #FFBC37;
    width: 15px;
}

.taskchecker{
    width: 150px;
    background-color: #FFBC37;
    border-style: solid;
    border-color: #000000;
    border-width: 1px;
}

.tcpadding{
    padding: 10px;
}
```

```
//functions.js
```

```
function createform (e){

    theObject = document.getElementById ("createtask");

    theObject.style.visibility = "visible";
    theObject.style.height = "200px";
    theObject.style.width = "200px";

    var posx = 0;
    var posy = 0;

    posx = e.clientX + document.body.scrollLeft;
    posy = e.clientY + document.body.scrollTop;

    theObject.style.left = posx + "px";
    theObject.stule.top = posy + "px";

    //La ubicación en la que cargamos la página.

    var objId = "createtask";
    var serverPage = "theform.php";

    var obj = document.getElementById(objId);
    xmlhttp.open("GET", serverPage);
    xmlhttp.onreadystatechange = function(){
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200){
            obj.innerHTML = xmlhttp.responseText;
        }
    }
    xmlhttp.send(null);
}

function closetask(){

    theObject = document.getElementById("createtask");

    theObject.style.visibility = "hidden";
    theObject.style.height = "0px";
    theObject.style.width = "0px";

    acObject = document.getElementById("autocompletediv");

    acObject.style.visibility = "hidden";
    acObject.style.height = "0px";
    acObject.style.width = "0px";
}

function findPosX(obj){
    var curleft = 0;
    if (obj.offsetParent){
        while (obj.offsetParent){
            curleft += obj.offsetLeft           // Al final no le puse el punto y
        }                                     coma porque en el libro no lo tiene.
        obj = obj.offsetParent;
    }
    }else if (obj.x){
        curleft += obj.x;
    }
    return curleft;
}

function findPosY(obj){
    var curtop = 0;
```



```
        if (obj.offsetParent){
            while (obj.offsetParent){
                curtop += obj.offsetTop; // Al final no le puse el
                obj = obj.offsetParent;
            }
        }
        }else if (obj.y){
            curtop += obj.y;
        }
        return curtop;
    }
}

function autocomplete (thevalue, e){

    theObject = document.getElementById ("autocompletediv");

    theObject.style.visibility = "visible";
    theObject.style.width = "152px";

    var posx = 0;
    var posy = 0;

    posx = (findPosX (document.getElementById("yourname")) + 1);
    posy = (findPosY (document.getElementById("yourname")) + 23);

    theObject.style.left = posx + "px";
    theObject.style.top = posy + "px";

    var theextrachar = e.which;

    if (theextrachar == undefined){
        theextrachar = e.keyCode;
    }

    //La ubicación en la que cargamos la página.
    var objId = "autocompletediv";

    //Tener en cuenta el carácter Backspace.
    if (theextrachar == 8){
        if (thevalue.length == 1){
            var serverPage = "autocomp.php";
        }else{
            var serverPage = "autocomp.php" + "?string=" + thevalue.substr (0,
(thevalue.length -1));
        }
    }else{
        var serverPage = "autocomp.php" + "?sstring=" + thevalue +
String.fromCharCode (theextrachar);
    }

    var obj = document.getElementById(objId);
    xmlhttp.open("GET", serverPage);
    xmlhttp.onreadystatechange = function(){
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200){
            obj.innerHTML = xmlhttp.responseText;
        }
    }
    xmlhttp.send(null);
}

function setvalue (thevalue){
    acObject = document.getElementById("autocompletediv");

    acObject.style.visibility = "hidden";
}
```

```
acObject.style.height = "0px";
acObject.style.width = "0px";

document.getElementById ("yourname").value = thevalue;
}
```

Básicamente hemos agregado algunas funciones; La primera de ellas se llama createform y sirve para mostrar un div oculto tras la posición actual del puntero del ratón y cargar, mediante Ajax, el resultado de theform.php. Esta función utiliza sobre todo JavaScript para hacer su trabajo (a través de los estilos hidden y visible), pero Ajax también desempeña un importante papel, por ser el responsable de cargar el archivo.

El código del archivo theform.php (que genera un sencillo formulario de introducción de datos) se muestra a continuación.

```
<-- theform.php -->

<div style="padding: 10px;">
  <form method="post">
    Su nombre: <br/>
    <input id="yourname" style="width: 150px; height: 16px;"
    type="text" value="" onkeypress="autocomplete(this.value,
    event)" /> <br />
    Tu pregunta<br />
    <textarea style="height: 80px;"></textarea><br />
    <div align="right"><a href="javascript:
    closetask()">Close</a></div>
  </form>
</div>
```

El siguiente conjunto de funciones se encarga de funciones de limpieza y de gestión de peticiones. La función closetask “cierra”, es decir, oculta, la ventana de cita / tarea si el usuario decide que ya no quiere configurar una cita.

Las funciones findPosX y findPosY devuelven las coordenadas del campo al que queremos aplicar la funcionalidad de autocompletar.

Las dos funciones siguientes, autocomplete y setvalue, son las que realmente se encargan de autocompletar el campo. Básicamente, la función autocomplete comprueba la cadena que ya ha escrito el usuario (mediante el evento onkeypress) y la envía al archivo autocomp.php a través de Ajax. Ha sido necesario introducir un poco de código para lograr que esta función sea compatible con el mayor número de navegadores posible (la gestión de pulsaciones de teclas y eventos puede variar de unos navegadores a otros).

Lo importante es que pasamos una cadena que contiene lo que ha escrito el usuario en el cuadro de texto (yourname) al archivo PHP de forma dinámica.

El siguiente bloque de código corresponde al programa PHP que recibe esta información.

```
<?php
//autocomp.php

//Una lista de todos los nombres
//Generalmente esto se almacenaría en algún tipo de base de datos.

$name = array("Lee Babin","Joe Smith","John Doe");
$foundarr = array();

//Recorrer los nombres de la matriz y cargar los coincidentes en foundarr.
if ($_GET['sstring'] != ""){
    for ($i = 0; $i < count ($names); $i++){
        if (substr_count (strtolower ($names[$i]), strtolower ($_GET['sstring']))
        > 0){
            $foundarr[] = $names[$i];
        }
    }
}
```

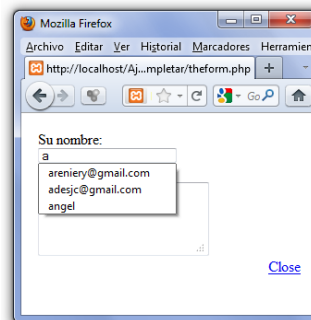
```
    }  
  }  
  //Si existe alguna coincidencia...  
  if (count ($foundarr) > 0){  
    //...lso modtramos.  
    ?>  
    <div style="background: #cccccc; border-style: solid; border-width: 1px; border-  
    color: #000000;">  
    <?php  
      for ($i = 0; $i < count ($foundarr); $i++){  
        ?><div style="padding: 4px; height: 14px;"  
        onmouseover="this.style.background = '#EEEEEE'"  
        onmouseout="this.style.background = '#cccccc'"  
        onclick="setvalue('<?php echo $foundarr[$i];?>')">  
        <?php echo $foundarr[$i];?></div><?php  
      }  
    ?>  
  </div>  
  <?php  
  }  
  ?>
```

El archivo autocomp.php recibe la cadena e intenta localizar nombres que coincidan con ella. Las coincidencias que encuentra las almacena en otra matriz. La razón para utilizar una razón alternativa es mantener nula la altura del div, al menos que se encuentre alguna coincidencia. Si se detecta alguna o varias coincidencias, las mostrará la función de autocompletar. Si el usuario hace clic sobre una de estas coincidencias, el nombre correspondiente se cargará en el cuadro de texto (mediante la función setvalue) y se cerrará el formulario emergente. Ya tenemos una función operativa de estilo autocompletar creada mediante Ajax, a como se muestra en la siguiente ilustración:

Ilustración 6.6 La función de autocompletar logra que introducir datos resulte sencillo y efectivo.

Validación de formularios

No trataremos a profundidad este tema, pero exponemos aquí un bonito truco que nos permite realizar validaciones mediante Ajax, especialmente teniendo en cuenta que este tipo de validaciones pueden resultar bastante complejas



sin Ajax. La mayoría de los campos de un formulario pueden validarse desde el lado del servidor utilizando JavaScript, 'para detectar si ean vacíos, si contienen información correcta, ... Sin embargo, siempre ha resultado engorroso realizar validaciones que dependen de los atos almacenados en una base de datos a los que sólo se puede acceder desde el lado del servidor.

La solución que solía utilizarse antes de Ajax era enviar todo el formulario, validarlo en el servidor y devolver todos los datos que había escrito el usuario, recargándolos en el formulario antes de mostrarlo de nuevo. Aunque este sistema funciona perfectamente bien, resulta complejo de codificar y no se puede aplicar a campos de tipo file (que no admiten ser recargados de forma automática).

En el siguiente ejemplo ampliaremos la función de creación de citas en el calendario, pero antes de enviar el formulario, comprobaremos mediante un script que el nombre escrito existe. De este modo estaremos simulando un

validador de nombres de usuario, en el que el usuario no puede utilizar un nombre que ya ha sido elegido por otro usuario que se registró anteriormente.

En lugar de volver a incluir todo el código, nos centraremos sólo en los cambios. En primer lugar añadiremos una nueva función, llamada `validateform`, cuyo código es el siguiente:

```
function validateform (thevalue){
    serverpage = "validator.php?sstring=" + thevalue;
    objID = "messagebox";

    var obj = document.getElementById(objID);
    xmlhttp.open("GET", serverPage);
    xmlhttp.onreadystatechange = function(){
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200){
            obj.innerHTML = xmlhttp.responseText;
        }
    }
    xmlhttp.send(null);
}
```

Esta función carga un script PHP en cierta sección de la página. El siguiente código contiene los cambios en el formulario:

```
<!-- theform.php -->

<div style="padding: 10px;">
    <div id="messagebox"></div>      <!-- Esta línea es nueva -->
    <form method="post">
        Su nombre: <br/>
        <input id="yourname" style="width: 150px; height: 16px;" type="text" value=""
        onkeypress="autocomplete(this.value, event)" /> <br />
        Tu tarea/cita<br />
        <textarea style="height: 80px;"></textarea><br />
        <input type="button" value="submit"
        onclick="validateform(document.getElementById('yourname').value)" /> <!--
        Esta línea es nueva -->
        <div align="right"><a href="javascript: closetask()">close</a></div>
    </form>
</div>
```

Como puede comprobar, hemos añadido una división con el nombre `messagebox` (en las que mostraremos los errores de validación) y un botón que nos servirá para ejecutar la función `validateform`.

Al hacer clic sobre este botón se ejecutará la función `validateform` que, a su vez, accederá al script PHP que contiene el archivo `validator.php`. El código de este script es el siguiente:

```
<?php
//validator.php

//Una lista de nombres v'alidos.
//Tambi'en en este caso, lo normal ser'ia utilizar una base de datos.
$names = array ("Msc. Aldo M. Delgadillo","Angel Reniery","Edilberto R. Navas.");

if(!in_array(strtolower($_GET['sstring']), explode(", ", strtolower(implode(", ",
$names))))){//La respuesta con un error.
    ?><span style="color: #FF0000;">No hay nombre...</span><?php
}else{
    //En este punto volvemos al script de procesado.
    ?><span style="color: #FF0000;">El formulario ser'a enviado</span><?php
}
?>
```

Ajax se convierte en una técnica cada vez más ampliamente utilizada, para uno de los usos más comunes como la navegación. Es un proceso bastante sencillo cargar de forma dinámica el contenido en una página a través del método Ajax. Sin embargo, con Ajax cargas el contenido exactamente en donde usted lo pida, sin actualizar la página, es importante tener en cuenta el lugar en que exactamente se carga el contenido.

6.6. Precauciones y consejos sobre MySQL.

Al utilizar Ajax para establecer conexiones con MySQL hay varios aspectos que debemos tener en cuenta. En primer lugar, conviene hacer notar que establecer conexiones con bases de datos a través de interfaces basadas en Ajax puede suponer una sobrecarga de trabajo insoportable para el servidor de bases de datos si no tomamos precauciones. Si tenemos en cuenta que cada usuario desde cada página puede iniciar varias consultas a la base de datos, la probabilidad de saturar las posibilidades del servidor aumenta drásticamente. Por ejemplo, pensemos en una página que dispone de tres elementos capaces de comunicarse con bases de datos a través de Ajax. Esto significa que cada página podría mantener hasta tres peticiones abiertas por cada usuario. Si esta página se encuentra en un sitio con mucha afluencia de visitantes, es probable que el servidor de bases de datos no pueda soportarlo.

Es de esperar que las opciones de gestión de las conexiones mejoren con el tiempo, por lo que esto dejará de ser un problema para Ajax, pero de momento es importante tenerlo en cuenta para no caer en la tentación de desarrollar aplicaciones virtualmente insostenibles.

En segundo lugar debemos analizar la ergonomía de los elementos en los que cargamos los resultados de MySQL. Por ejemplo, si refrescásemos completamente la página y necesitésemos mostrar un mensaje de error, simplemente tendríamos que cargarlo en una posición de la página en la que resulte muy visible. Sin embargo, al trabajar con Ajax, frecuentemente tendremos que cargar la información en elementos menos visibles, más reducidos y menos llamativos. Consecuentemente, tendremos que meditar con más cuidado como atraer la atención del usuario sobre lo que realmente es más importante. En particular, los errores MySQL suelen ser bastante largos, por lo que podría ser aconsejable redirigir estos mensajes de error al correo electrónico del administrador, y mostrar al usuario un aviso más conciso.

En lo concerniente a la seguridad, debemos ser más cuidadosos que nunca. Aunque a priori podría parecer que los scripts a los que accedemos mediante Ajax son más seguros que los que generan páginas completas, en realidad son igual de vulnerables o incluso más. La razón se encuentra en que todo el código JavaScript es visible para cualquiera que indique a su navegador que le muestre el código fuente de la página.

Consiguientemente, un usuario malintencionado podría hacerse con la referencia a un archivo script del servidor e intentar explotarlos si estos no impiden que se acceda a ellos de forma directa. Como hasta ahora todas nuestras peticiones Ajax han sido de tipo GET, también existe la posibilidad de que suframos ataques de inyección de código, y más concretamente, por inyección SQL.

Una inyección SQL es una técnica que consiste en introducir código malintencionado dentro de la cadena de la consulta (en la barra de direcciones del navegador), con el fin de manipular las consultas dinámicas y obtener información que debería mantenerse oculta.

No hay una regla simple para la identificación de vulnerabilidades de inyección SQL: ni anomalías de activación por el envío de datos inesperados. A pesar de ello, podemos destacar algunas formas:

- Usted puede identificar todos los datos de entrada en una aplicación web.
- Saber que tipo de solicitudes pueden provocar anomalías.
- Detectar anomalías en las respuestas del servidor.

Es tan simple como eso. Primero que hay que ver cómo el navegador web envía peticiones al servidor web. Las diferentes aplicaciones se comportan de manera diferente, pero los fundamentos se deben a lo mismo, ya que todos son entornos basados en la Web.



Una vez que identifica todos los datos aceptados por la aplicación, es necesario modificarlos y analizar la respuesta del servidor. A veces, la respuesta incluirá un error SQL directamente de la base de datos y le hará la vida muy fácil, sin embargo, otras veces se necesitará mantener el enfoque y detectar diferencias sutiles.

Los entornos web son un ejemplo de arquitectura cliente / servidor. Su navegador (que actúa como cliente), envía una petición al servidor y espera una respuesta. El servidor recibe la petición, genera una respuesta, y lo envía de vuelta al cliente. Obviamente, debe haber algún tipo de entendimiento entre las dos partes, de lo contrario, el cliente solicita algo y el servidor no sabría qué contestar.

La comprensión de ambas partes está dada por el uso de un protocolo, en este caso, HTTP. Nuestra primera tarea es identificar todas las entradas de datos aceptados por la aplicación Web remotos. HTTP define una serie de acciones que un cliente puede enviar al servidor, sin embargo, nos centraremos en los dos más relevantes para el propósito de descubrir la inyección SQL: los métodos GET y POST.

Peticiones GET

GET es un método HTTP que envía peticiones al servidor y toda la información se indica en la dirección URL. Este es el tipo de método que se utiliza normalmente cuando se hace clic en un enlace. Por lo general, el navegador Web se crea la petición GET, se envía al servidor web, y este muestra el resultado en el navegador.

A pesar de que es transparente para el usuario, la solicitud GET que se envía al servidor web es la siguiente:

```
GET /search.aspx?text=lcd%20monitors&cat=1&num=20 HTTP/1.1
Host:www.victim.com
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.8.1.19)
Gecko/20081216 Ubuntu/8.04 (hardy) Firefox/2.0.0.19
Accept: text/xml,application/xml,application/xhtml+xml,
text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
```

Este tipo de solicitud envía los parámetros dentro de la URL en el siguiente formato:

```
?parameter1=value1&parameter2=value2&parameter3=value3...
```

En el ejemplo anterior, se pueden ver tres parámetros: text, cat, y num. El mando a distancia de la aplicación recupera los valores de los parámetros y los utiliza para el propósito con el que han sido diseñados. Con las peticiones GET se pueden manipular los parámetros, simplemente haciendo el cambio en la barra de herramientas de navegación de su navegador. Alternativamente, también puede utilizar una herramienta proxy.

Las solicitudes POST

POST es un método HTTP usado para enviar información al servidor Web. La acción que realiza el servidor viene determinada por la dirección URL de destino. Este suele ser el método utilizado cuando se rellena un formulario en el navegador y haces clic en el botón Enviar. Este es un ejemplo de lo que se envía al servidor web remoto:

```
POST /contact/index.asp HTTP/1.1
Host:www.victim.com
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.8.1.19) Gecko/20081216
Ubuntu/8.04 (hardy) Firefox/2.0.0.19
Accept: text/xml,application/xml,application/xhtml+xml,
text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
```

```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Referer: http://www.victim.com/contact/index.asp
Content-Type: application/x-www-form-urlencoded
Content-Length: 129
```

Primero=John&last=Doe&email=john@doe.com&phone=555123456&title=Mr&country=US&comments=I%20would%20like%20to%20request%20information.

Los valores que se envían al servidor Web tienen el mismo formato que se explica por la petición GET, pero ahora se encuentra en la parte inferior de la solicitud.

Otros datos inyectables.

La mayoría de las aplicaciones para recuperar datos de parámetros GET o POST. Sin embargo, otras partes de la Petición HTTP podrían dar lugar a vulnerabilidades de inyección SQL. Las cookies son un buen ejemplo. Se envían las cookies en el navegador del usuario y son automáticamente enviadas de vuelta al servidor en cada solicitud. Las cookies se utilizan generalmente para la autenticación, sesión de control y mantenimiento de información específica sobre el usuario, tales como preferencias en el sitio Web. Como se explicó anteriormente, usted tiene el control total de los contenidos enviados al servidor y por lo tanto usted debe considerar las cookies como una forma válida de entrada de datos del usuario y por lo tanto, susceptibles de ser objeto de la inyección.

Otros ejemplos de aplicaciones vulnerables a la inyección en otras partes del HTTP es solicitar la hostia, Referer, y encabezados User-Agent. El campo de encabezado de host especifica el Host de Internet y número de puerto del recurso que se solicita. El campo de referencia especifica el recurso del que se obtuvo la solicitud actual. El campo encabezado User-Agent determina el navegador web utilizado por el usuario. Aunque estos casos son poco comunes, algunas aplicaciones de monitoreo de red y Web tienden a utilizar el anfitrión, referer, y el encabezado User-Agent para crear gráficos y almacenarlos en bases de datos. En estos casos, es necesario probar las cabeceras por las vulnerabilidades de inyección potencial.

Manipulación de los parámetros

Empezaremos con un ejemplo muy simple para que pueda familiarizarse con la inyección de SQL y sus vulnerabilidades. Digamos que usted visita el sitio Web de la víctima, Inc., una tienda de comercio electrónico donde se puede comprar todo tipo de cosas. Puede comprobar la línea de productos, ordenarlos por precio, sólo mostrar una cierta categoría de productos, y así sucesivamente. Al navegar por las diferentes categorías de productos tenga en cuenta que la URL se parece a lo siguiente:

```
http://www.victim.com/showproducts.php?category=bikes
http://www.victim.com/showproducts.php?category=cars
http://www.victim.com/showproducts.php?category=boats
```

La página showproducts.php recibe un parámetro llamado categoría. Usted no tiene que escribir nada, ya que los enlaces anteriores se presentan en el sitio Web, por lo que sólo tienes que hacer clic en ellos. La aplicación en el servidor está esperando los valores conocidos y muestra los productos que pertenecen a la misma categoría.

Incluso sin iniciar el proceso de las pruebas de las que usted ya debe tener una idea aproximada sobre cómo se aplican. Se puede afirmar que la aplicación no es estática, sino que parece que dependiendo del valor del parámetro de la categoría de aplicación se muestran diferentes productos basados en el resultado de una consulta a una base de datos de back-end. Ahora puede comenzar a cambiar manualmente los valores de los parámetros de categoría para que la aplicación haga algo que no espera. Su primer intento puede ser algo como lo siguiente:

<http://www.victim.com/showproducts.php?category=attacker>

En el ejemplo anterior, se envió una petición al servidor con una categoría que no existe. La respuesta del servidor fue la siguiente:



Warning: MySQL_fetch_assoc(): supplied argument is not a valid MySQL result resource in /var/www/victim.com/showproducts.php on line 34

Esta advertencia es un error de base de datos MySQL devuelto por la base de datos cuando el usuario intenta leer un registro de un conjunto de resultados vacíos. Este error indica que la aplicación remota no está haciendo el manejo adecuado.

Continuando con el proceso de inferencia de su solicitud, añadiendo una comilla simple (') con el valor que ya haya enviado:

[http://www.victim.com/showproducts.php?category=attacker'](http://www.victim.com/showproducts.php?category=attacker)

El servidor devolvió el siguiente error:

Usted tiene un error en la sintaxis SQL, consulte el manual que corresponde a su versión del servidor MySQL para el correcto uso de la sintaxis hay un "ataque" en la línea...

Como puede ver, algunas aplicaciones pueden reaccionar de manera inesperada con el manejo de datos de usuario. No todas las anomalías detectadas en un sitio web van a ser debido a una vulnerabilidad de inyección SQL, ya que puede ser afectada por una serie de otras cuestiones. A medida que se familiarice con la explotación de inyección SQL, se darán cuenta de la importancia de los caracteres de cotización para fines de detección y usted aprenderá a enviar las solicitudes apropiadas para que el servidor determinar qué tipos de inyecciones son posibles.

Sanitización de Datos del Usuario.

Las Vulnerabilidades de inyección SQL se producen por dos razones:

- La falta de saneamiento de entrada del usuario.
- Los datos y las estructuras de control mixto en el mismo canal de transporte.

Estos dos problemas juntos han sido la causa de algunos de los más importantes tipos de vulnerabilidades explotadas hasta ahora en la historia de las computadoras, como la pila y desbordamientos de pila, y las cuestiones de formato de cadena. La falta de saneamiento de entrada de usuario permite a un atacante saltar desde la parte de datos (por ejemplo, una cadena encerrada entre comillas simples o un número) para inyectar comandos de control (como select, UNION, AND, OR, etc). Para defenderse de este tipo de vulnerabilidad de la primera medida a adoptar es llevar a cabo un estricto control de entrada del usuario desinfección y / o codificación de salida. Por ejemplo, puede adoptar un enfoque de lista blanca, por lo que si usted está esperando un número como el valor del parámetro, se puede configurar la aplicación Web para rechazar todos los personajes de la entrada proporcionada por el usuario que no es un dígito. Si usted está esperando una cadena, que sólo aceptan los caracteres que no son peligrosos. Cuando esto no sea posible, debe asegurarse de que todas las entradas se citaron correctamente / codificados antes de ser utilizado para evitar la inyección SQL.

Errores de base de datos

Hemos visto algunos errores de inyección SQL aparece como resultado de la manipulación de parámetros. A pesar de los errores se muestran en la respuesta del servidor Web, la inyección SQL sucede en la capa de base de datos. Estos ejemplos muestran cómo se puede llegar a un servidor de base de datos a través de la aplicación web. Es muy importante que se familiarice con los errores de bases de datos diferentes que puede obtener desde el servidor Web cuando pruebe las vulnerabilidades de inyección SQL. La ilustración 6.6 muestra cómo un error de inyección SQL sucede y cómo el servidor Web se ocupa de ella.

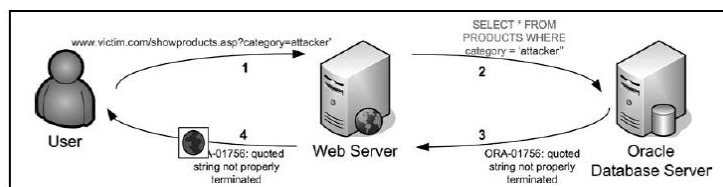


Ilustración 6.7 Flujo de información en un error de inyección SQL.

Como se puede ver en la Ilustración 6.6, ocurre lo siguiente durante un error de inyección de SQL:

1. El usuario envía una solicitud en un intento de identificar una vulnerabilidad de inyección SQL. En este caso, el usuario envía un valor con una sola cita que se le ha añadido.
2. El servidor Web recupera los datos del usuario y envía una consulta SQL al servidor de base de datos.
3. En este ejemplo, se puede ver que la instrucción SQL creada por el servidor Web incluye la entrada del usuario y forma una consulta sintácticamente incorrecta debido a las dos terminaciones de las cotizaciones.
3. El servidor de base de datos recibe la malformación consulta SQL y devuelve un error a la del servidor web.
4. El servidor Web recibe el error de la base de datos y envía una respuesta HTML para el usuario. En este caso, se envió el mensaje de error, pero es totalmente de la aplicación cómo se presenta algún error en el contenido de la respuesta HTML.

El ejemplo anterior ilustra el escenario de una solicitud por parte del usuario que activa un error en la base de datos. Dependiendo de cómo la aplicación esta codificada, el archivo devuelto en el paso 4 será construido y manejado como uno de los siguientes resultados:

- El error de SQL que se muestra en la página y que es visible para el usuario del Navegador web.
- El error de SQL que está escondido en la fuente de la página Web es para fines de depuración.
- Redirección a otra página se utiliza cuando se detecta un error.
- Un código de error HTTP 500 (Error interno del servidor) o el código de redirección HTTP 302 se devuelve.
- La aplicación controla el error correctamente y no simplemente muestra los resultados, tal vez que muestra una página de error genérica.
- Cuando usted está tratando de identificar a una vulnerabilidad de inyección SQL es necesario determinar el tipo de respuesta de la aplicación que va a regresar.

Para investigar acerca de la inyección MySQL, puede reforzar y profundizar sus conocimientos en el libro SQL Injection Attacks and Defense.

Ejemplo de Ajax con PHP y MySQL.

Este ejemplo es sencillo, no implica un grado de complejidad fuerte pero es suficiente para que usted comprenda la funcionalidad de Ajax con conexiones a bases de datos, particularmente usaremos una base de datos creada en MySQL, usted puede sentirse libre de crear la base de datos con el software de su preferencia, el script de la base de datos se presenta a continuación:

1. Conectar Ajax con la base de datos, la base de datos se llamará test (puede utilizar la misma que se creó en capítulos anteriores) y dentro de esta crearemos una tabla llamada usuario, cuyo código es el siguiente:

```
CREATE TABLE usuario
(
  user_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  user_name VARCHAR(32) NOT NULL,
  PRIMARY KEY (user_id)
);
```

2. Ejecute el siguiente commando de inserción (INSERT) para probar su tabla con algunos datos:

```
INSERT INTO usuario (user_name) VALUES ('bogdan');
INSERT INTO usuario (user_name) VALUES ('audra');
INSERT INTO usuario (user_name) VALUES ('cristian');
```

La columna user_id se auto incrementará, este valor será generado por la base de datos.

3. En su servidor local, crear una carpeta llamada MySQL.
4. Dentro de la carpeta MySQL, crear un archivo llamado config.php y agregar el código de la configuración de la base de datos (estos valores los puede cambiar dependiendo de la configuración de su propia base de datos):

```
<?php
// definiendo la conexión con la base de datos
define('DB_HOST', 'localhost');
define('DB_USER', 'root');
define('DB_PASSWORD', '');
define('DB_DATABASE', 'test');
?>
```

5. Crear un archivo llamado error_handler.php el cual controlará los errores.

```
<?php
// el método error_handler será el manejador de errores del usuario
set_error_handler('error_handler', E_ALL);
// función manejadora de errores
function error_handler($errNo, $errStr, $errFile, $errLine)
{
    // limpiar cualquier salida que hayan sido generadas
    if(ob_get_length()) ob_clean();
    // salida del mensaje de error
    $error_message = 'ERRNO: ' . $errNo . chr(10) .
        'TEXT: ' . $errStr . chr(10) .
        'LOCATION: ' . $errFile .
        ', line ' . $errLine;
    echo $error_message;
    // prevenir el procesamiento de cualquier otro scripts PHP
    exit;
}
?>
```

6. Crear el archivo llamado index.php y agregar el siguiente código:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.
w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
<head>
<title>Practica con AJAX: Trabajando con PHP y MySQL</title>
</head>
<body>
<?php
    // cargando archivos de configuración
    require_once('error_handler.php');
    require_once('config.php');
    // conectandose a la base de datos
    $mysqli = new mysqli(DB_HOST, DB_USER, DB_PASSWORD, DB_DATABASE);
    // ejecutando la consulta the SQL
    $query = 'select user_id, user_name FROM user';
    // ejecutando la consulta
    $result = $mysqli->query($query);
    // ejecutando ciclo a través de los resultados
    while ($row = $result->fetch_array(MYSQLI_ASSOC))
    {
```

```
// extrayendo user_id y sus nombres
$user_id = $row['user_id'];
$user_name = $row['user_name'];
// hacer algo con los datos (aquí es donde nosotros los mostramos)
echo 'El identificador del usuario es #' . $user_id . ' su nombre es ' . $user_name .
'<br/>';
}
// cerrar el flujo de entrada
$result->close();
// cerrar la conexión a la base de datos
$mysqli->close();
?>
</body>
</html>
```

7. Solo nos queda probar nuestro ejercicio, el cual deberá generar una salida como esta:

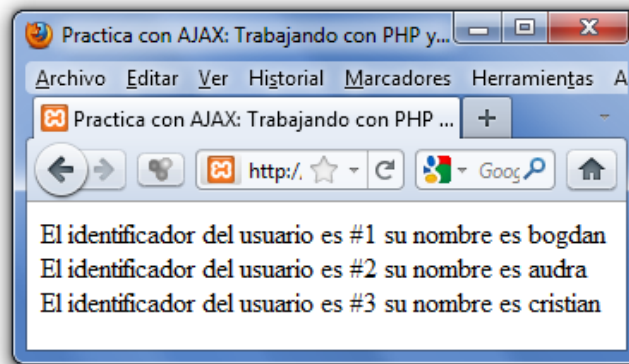


Ilustración 6.8 Los nombres de los usuarios son leídos desde la base de datos.

URL: www.unixwiz.net/techips/sql-injection.html

6.7. Validando Formularios

Obviamente, una cosa es obtener datos de una base de datos MySQL y realizar consultas dinámicamente para que se almacenen en esta base de datos. La forma mas habitual de solicitar al usuario es a través de formularios. Existen muchos tipos distintos de controles de formularios, que nos permiten recabar información de formas muy diversas. Para lograr que nuestros formularios sean lo más intuitivos posible, es importante conocer todas las posibilidades que tenemos a nuestra disposición para solicitar datos al usuario. Ver la sección de formularios en HTML del capítulo 1.

Durante años, los desarrolladores han venido haciendo buen uso (casi siempre) de estos controles de formulario para crear aplicaciones Web eficientes. Esta larga evolución ha permitido que los programadores y los diseñadores dispongan de implementaciones muy refinadas de todo tipo de funcionalidades Web. Por supuesto, lo que nosotros vamos a añadir es la posibilidad de que funcionen de forma inmediata (o, para ser más exactos, que así lo parezca), sin que se tenga que refrescar la página. Este sueño puede hacerse realidad gracias a Ajax.

Ejemplo:

1. Primero debemos crear el archivo xhr.js y agregar el siguiente código:

```
// El constructor XmlHttpRequest puede recibir cualquiera de las siguientes peticiones:
// url - La url del servidor.
// contentType - solicitudes de tipo de contenido
// tipo - solicitudes de tipo (por defecto es GET)
// datos - solicitudes de parametros opcionales
// async - si la petición es asíncrona (por defecto es verdadero)
```

```
// showErrors - desplegar (mostrar) errores
// complete - la función callback se llama cuando hay una petición
// completar
function XmlHttpRequest(settings)
{
    // almacenar las escenas del objeto en una clase propiedad
    this.settings = settings;
    // sustituir las escenas por defecto con los parámetros recibidos
    // la url por defecto apuntan a la página actual
    var url = location.href;
    if (settings.url)
        url = settings.url;
    // los tipos de contenido por defecto son tipos contenidos por formularios
    var contentType = "application/x-www-form-urlencoded";
    if (settings.contentType)
        contentType = settings.contentType;
    // las peticiones por defecto, son hechas a través de GET
    var type = "GET";
    if (settings.type)
        type = settings.type;
    // no se envían parámetros por defecto
    var data = null;
    if (settings.data)
    {
        data = settings.data;
        // si nosotros a través de GET nosotros ajustamos la url apropiadamente
        if (type == "GET")
            url = url + "?" + data;
    }
    // por defecto, el retorno final es asíncrono
    var async = true;
    if (settings.async)
        async = settings.async;
    // nosotros por defecto, mostramos todas las infraestructuras de los errores
    var showErrors = true;
    if (settings.showErrors)
        showErrors = settings.showErrors;
    // create the XMLHttpRequest object
    var xhr = XMLHttpRequest.create();
    // crear el objeto XMLHttpRequest
    xhr.open(type, url, async);
    xhr.onreadystatechange = onreadystatechange;
    xhr.setRequestHeader("Content-Type", contentType);
    xhr.send(data);
    // esta función despliega los errores
    function displayError(message)
    {
        // ignorar los errores si showErrors es falso
        if (showErrors)
        {
            // desplegar el mensaje de error
            alert("Error encountered: \n" + message);
        }
    }
    // esta función lee las respuestas del servidor
    function readResponse()
    {
        try
        {
            // recupere el tipo de contenido de las respuestas
            var contentType = xhr.getResponseHeader("Content-Type");
            // construir el objeto json si la respuesta fue uno
            if (contentType == "application/json")
            {

```

```
response = JSON.parse(xhr.responseText);
}
// obtener el elemento DOM si la respuesta es XML
else if (contentType == "text/xml")
{
response = xhr.responseXml;
}
// por defecto obtener la respuesta como texto
else
{
response = xhr.responseText;
}
// llamar a la función callback si cualquiera
if (settings.complete)
settings.complete (xhr, response, xhr.status);
}
catch (e)
{
displayError(e.toString());
}
}
// llamada cuando la petición cambio su estado
function onreadystatechange()
{
    // cuando la lectura del estado es 4, nosotros leemos la respuesta del servidor
    if (xhr.readyState == 4)
    {
        // continúe únicamente si el estado de HTTP "OK"
        if (xhr.status == 200)
        {
            try
            {
                // leer la respuesta desde el servidor
                readResponse();
            }
            catch(e)
            {
                // desplegar el mensaje de error
                displayError(e.toString());
            }
        }
        else
        {
            // desplegar el mensaje de error
            displayError(xhr.statusText);
        }
    }
}
// los métodos estáticos retornan un nuevo objeto XMLHttpRequest
XmlHttp.create = function()
{
    // almacenará las referencias de el objeto XMLHttpRequest
    var xmlHttp;
    // crear el objeto XMLHttpRequest
    try
    {
        // asume IE7 u otros modernos navegadores
        xmlHttp = new XMLHttpRequest();
    }
    catch(e)
    {
        // asume IE6 o mas viejos
        try
```

```
{
xmlHttp = new ActiveXObject("Microsoft.XMLHttp");
}
catch(e) { }
}
// retorna el objeto creado o despliega un mensaje de error
if (!xmlHttp)
alert("Error creando el objeto XMLHttpRequest.");
else
return xmlHttp;
}
```

2. Rápidamente probemos la funcionalidad de nuestra clase XmlHttp, creando un nuevo archivo llamado xhrtest.html y agregue el siguiente código:

```
<html>
<head>
  <script type="text/javascript" src="xhr.js"></script>
</head>
<body>
  <div id="test">
  </div>
  <script>
    XmlHttp
    ({url:'async.txt',
    complete:function(xhr,response,status)
    {
      document.getElementById("test").innerHTML = response;
    }
    });
  </script>
</body>
</html>
```

3. Ahora, crear async.txt con algún tipo de texto dentro, y finalmente cargar nuestra página <http://localhost/ajax/validar/xhrtest.html>, debemos obtener un resultado parecido al que se muestra en la ilustración:

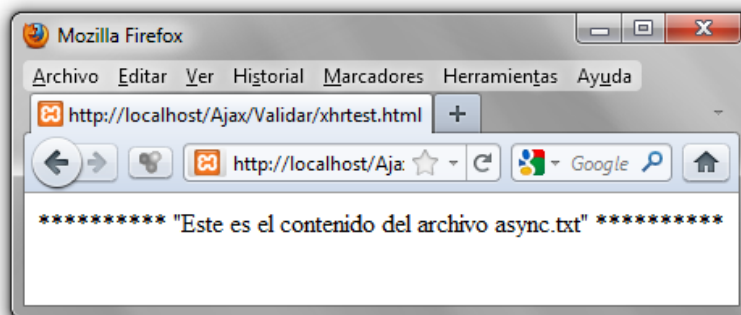


Ilustración 6.9 Salida de la clase XmlHttp

Explicación:

El nombre que escogido para nuestro objeto reusable es XmlHttp. Su funcionalidad se concentra en dos funciones:

- XmlHttp.create(): Es un método estático del objeto XmlHttp que se crea en el objeto XMLHttpRequest.
- XmlHttp(): Es el constructor del objeto XmlHttp.

Desde el punto de vista de diseño, el objeto XmlHttpRequest representa una envoltura alrededor del objeto XMLHttpRequest. El método XmlHttpRequest.create() contiene código del método XMLHttpRequestObject(). Esto simplemente actúa como un factor en el objeto XmlHttpRequest.

El constructor del objeto XMLHttpRequest, aunque puede parecer bastante complejo a primera vista, contiene en realidad un código simple siempre que conozca la teoría de JavaScript orientado a objetos. El constructor recibe como parámetro un Objeto JSON que contiene todos los ajustes para el objeto XMLHttpRequest. La elección de un objeto JSON es un tanto conveniente desde el punto de vista de extensibilidad y fácil desde el punto de vista de la programación. Nosotros guardamos la configuración de una propiedad con el mismo nombre.

```
function XmlHttpRequest(settings)
{
    // store the settings object in a class property
    this.settings = settings;
```

El objeto de configuración contiene las siguientes propiedades que se utilizan principalmente para el objeto XMLHttpRequest:

- url: La URL de la petición AJAX.
- Tipo: El tipo de petición (GET o POST).
- contentType: El tipo de contenido de la solicitud.
- Datos: Los datos que se envían al servidor.
- async: Una bandera que indica si la petición es síncrona o asíncrona.
- completa: La función se llama cuando la solicitud completa.
- ShowErrors: Una bandera que indica si los errores de la infraestructura se muestran o no.

Estos son los parámetros necesarios para hacer una petición AJAX. A pesar de que la estructura y el diseño de este objeto son simples, se puede ampliar fácilmente con características más avanzadas, que nos da la característica de flexibilidad que se define como una meta. La flexibilidad ofrecida por los objetos JSON significa que no obligan al usuario a pasar todos los las propiedades mencionadas anteriormente cada vez que se crea el objeto. En su lugar, hemos creado un conjunto estándar de valores por defecto que el usuario puede elegir para sobrescribir cuando sea necesario. Las siguientes líneas simplemente poner en práctica esta lógica.

Hacer una solicitud a través de GET o POST es diferente y nosotros nos encargamos de establecer los parámetros de la solicitud:

```
//Por defecto no hay parámetros enviados.
var data = null;
if(settings.data)
{
    data = settings.data;
    // if we go through GET we properly adjust the URL
    if(type == "GET")
        url = url + "?" + data;
}
```

Después de tener todas las opciones para la solicitud de AJAX, creamos el objeto XMLHttpRequest y lo abrimos.

```
// crear el objeto XMLHttpRequest
var xhr = XmlHttpRequest.create();
// establecer las propiedades de devolución de datos
xhr.open(type, url, async);
```

El siguiente paso es conectar el evento readystatechange:

```
xhr.onreadystatechange = onreadystatechange;
```

La función del manejador es una función interna del constructor y contiene los mismos código como el `handleRequestStateChange()` que usted ya conoce. Probablemente la pieza más interesante del código es el controlador de la respuesta. La función `readResponse()` es responsable para el manejo de la respuesta recibida desde el servidor. Se pone el tipo de contenido de la respuesta y, sobre esa base, se construye la respuesta el objeto JSON recupera la respuesta como un elemento XML. Si no hay tipo de contenido coincidente, el texto sin formato de la respuesta se utiliza en su lugar.

```
//recuperar el tipo de contenido de la respuesta
var contentType = xhr.getResponseHeader("Content-Type");
// crear el objeto Json si la respuesta tiene un
if (contentType == "application/json")
{
    response = JSON.parse(xhr.responseText);
}
// obtener el element DOM si la respuesta es XML
else if (contentType == "text/xml")
{
    response = xhr.responseXml;
}
// por defecto obtener la respuesta como texto.
else
{
    response = xhr.responseText;
}
```

Después de recopilar los datos necesarios, el objeto XMLHTTP pasa todo a la función de llamada de devolución (`settings.complete ()`) junto con el objeto XMLHTTP y el código de respuesta HTTP.

```
// llamar a la function de devolución de llamada si
if (settings.complete)
settings.complete (xhr, response, xhr.status);
```

Con todo, la próxima vez que necesite llamar a un script del servidor de forma asincrónica desde una página web, puede contar con que `XmlHttp` hace todo el trabajo sucio. Esto quiere decir que la URL de contacto, especifica los parámetros necesarios, y se obtiene la respuesta para usted.

El ejemplo anterior, lo ilustramos para que usted se familiarice y comprenda la estructura de nuestro siguiente ejemplo.

Ejemplo Validación de formulario con AJAX:

La aplicación contiene tres páginas:

- Una página hace el formulario que será validado.
- Otra página valida la entrada.
- En la tercera página se muestra si la validación es correcta.

La aplicación tendrá una estructura estándar, compuesta de los siguientes archivos:

- **index.php**: Es el archivo cargado inicialmente por el usuario. Contiene referencias a los archivos necesarios de JavaScript y hace peticiones asíncronas de validación de `validate.php`.
- **index_top.php**: Se trata de un archivo de ayuda cargado por `index.php` y contiene varios objetos para la representación de la forma HTML.
- **validate.css**: Es el archivo que contiene los estilos CSS para la aplicación.
- **json2.js**: Es el archivo JavaScript utilizado para la manipulación de objetos JSON.
- **xhr.js**: Es el archivo JavaScript que contiene nuestro objeto XMLHTTP utilizado para hacer peticiones AJAX.
- **validate.js**: Es el archivo JavaScript cargados junto con `index.php` en el lado del cliente. Hace peticiones asíncronas a un script PHP llamado `validate.php` para realizar la validación de AJAX.

- **validate.php**: Se trata de un script PHP que reside en el mismo servidor que index.php, y ofrece la funcionalidad de servidor de forma asíncrona solicitado por el código JavaScript en index.php.
 - **validate.class.php**: Se trata de un script PHP que contiene una clase llamada Validar, que contiene la lógica de negocio y las operaciones de base de datos y da soporte a la funcionalidad de validate.php.
 - **config.php**: se usará para guardar las opciones de la configuración globales para su aplicación, como los datos de conexión de bases de datos, y así sucesivamente.
 - **error_handler.php**: contiene el mecanismo de manejo de errores que cambia el texto de un mensaje de error en un formato humano-leíble.
 - **allok.php**: es la página a ser desplegada si la aprobación tiene el éxito.
1. Empecemos la escritura de nuestro código. Nosotros definiremos los estilos para nuestro formulario creando un archivo llamado validate.css, y agregando el código siguiente a él:

```
body
{
font-family: Arial, Helvetica, sans-serif;
font-size: 0.8em;
color: #000000;
}
label
{
float: left;
width: 150px;
font-weight: bold;
}
input, select
{
margin-bottom: 3px;
}
.button
{
font-size: 2em;
}
.left
{
margin-left: 150px;
}
.txtFormLegend
{
color: #777777;
font-weight: bold;
font-size: large;
}
.txtSmall
{
color: #999999;
font-size: smaller;
}
.hidden
{
display: none;
}
.error
{
display: block;
margin-left: 150px;
color: #ff0000;
}
```

2. Crear un nuevo archivo llamado index_top.php, y agregar el siguiente código, este script será cargada desde la página principal index.php.

```
<?php
// habilitar sesión PHP
session_start();
// Construir etiquetas HTML <option>
function buildOptions($options, $selectedOption)
{
    foreach ($options as $value => $text)
    {
        if ($value == $selectedOption)
        {
            echo '<option value="' . $value .
                '" selected="selected">' . $text . '</option>';
        }
        else
        {
            echo '<option value="' . $value . '">' . $text .
                '</option>';
        }
    }
}

// inicializando el array del género
$genderOptions = array("0" => "[Select]",
    "1" => "Male",
    "2" => "Female");
// inicializando el arreglo de los meses
$monthOptions = array("0" => "[Select]",
    "1" => "January",
    "2" => "February",
    "3" => "March",
    "4" => "April",
    "5" => "May",
    "6" => "June",
    "7" => "July",
    "8" => "August",
    "9" => "September",
    "10" => "October",
    "11" => "November",
    "12" => "December");
// inicializando algunas variables de sesiones para prevenir lanzar noticias PHP
if (!isset($_SESSION['values']))
{
    $_SESSION['values']['txtUsername'] = "";
    $_SESSION['values']['txtName'] = "";
    $_SESSION['values']['selGender'] = "";
    $_SESSION['values']['selBthMonth'] = "";
    $_SESSION['values']['txtBthDay'] = "";
    $_SESSION['values']['txtBthYear'] = "";
    $_SESSION['values']['txtEmail'] = "";
    $_SESSION['values']['txtPhone'] = "";
    $_SESSION['values']['chkReadTerms'] = "";
}
if (!isset($_SESSION['errors']))
{
    $_SESSION['errors']['txtUsername'] = 'hidden';
    $_SESSION['errors']['txtName'] = 'hidden';
    $_SESSION['errors']['selGender'] = 'hidden';
    $_SESSION['errors']['selBthMonth'] = 'hidden';
    $_SESSION['errors']['txtBthDay'] = 'hidden';
    $_SESSION['errors']['txtBthYear'] = 'hidden';
    $_SESSION['errors']['txtEmail'] = 'hidden';
    $_SESSION['errors']['txtPhone'] = 'hidden';
    $_SESSION['errors']['chkReadTerms'] = 'hidden';
}
?>
```

3. Crear el archivo index.php, y agregar el siguiente código:

```
<?php
    require_once ('index_top.php');
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.
w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>AJAX validaci&oacute;n de formulario con PHP y MySQL</title>
    <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8" />
    <link href="validate.css" rel="stylesheet" type="text/css" />
</head>
<body onload="setFocus();">
    <script type="text/javascript" src="json2.js"></script>
    <script type="text/javascript" src="xhr.js"></script>
    <script type="text/javascript" src="validate.js"></script>
    <fieldset>
        <legend class="txtFormLegend">
            Nuevo usuario Registrado en el formulario
        </legend>
        <br />
        <form name="frmRegistration" method="post"
        action="validate.php">
            <input type="hidden" name="validationType" value="php"/>
            <!-- Username -->
            <label for="txtUsername">Nombre de usuario deseado:</label>
            <input id="txtUsername" name="txtUsername" type="text"
            onblur="validate(this.value, this.id)"
            value="<?php echo $_SESSION['values']
            ['txtUsername'] ?>" />
            <span id="txtUsernameFailed"
            class="<?php echo $_SESSION['errors']['txtUsername']
            ?>">
                Este usuario est&aacute; en uso, o el campo esta vacío.
            </span>
            <br />
            <!-- Name -->
            <label for="txtName">Su nombre:</label>
            <input id="txtName" name="txtName" type="text"
            onblur="validate(this.value, this.id)"
            value="<?php echo $_SESSION['values']['txtName']
            ?>" />
            <span id="txtNameFailed"
            class="<?php echo $_SESSION['errors']['txtName']?>">
                Por favor, ingrese su nombre.
            </span>
            <br />
            <!-- Gender -->
            <label for="selGender">Genero:</label>
            <select name="selGender" id="selGender"
            onblur="validate(this.value, this.id)">
                <?php buildOptions($genderOptions,
                $_SESSION['values']['selGender']); ?>
            </select>
            <span id="selGenderFailed"
            class="<?php echo $_SESSION['errors']['selGender']
            ?>">
                Por favor seleccione su genero.
            </span>
            <br />
```

```
<!-- Birthday -->
<label for="selBthMonth">Fecha de nacimiento:</label>
<!-- Month -->
<select name="selBthMonth" id="selBthMonth"
onblur="validate(this.value, this.id)">
<?php buildOptions($monthOptions,
$_SESSION['values']['selBthMonth']);
?>
</select>
&nbsp;-&nbsp;
<!-- Day -->
<input type="text" name="txtBthDay" id="txtBthDay"
maxlength="2" size="2"
onblur="validate(this.value, this.id)"
value="<?php echo $_SESSION['values']['txtBthDay']
?>" />
&nbsp;-&nbsp;
<!-- Year -->
<input type="text" name="txtBthYear" id="txtBthYear"
maxlength="4" size="2"
onblur="validate(document.getElementById
('selBthMonth').options[document.getElementById
('selBthMonth').selectedIndex].value +
'#' + document.getElementById('txtBthDay').value +
'#' + this.value, this.id)"
value="<?php echo $_SESSION['values']['txtBthYear']
?>"
/>
<!-- Month, Day, Year validation -->
<span id="selBthMonthFailed"
class="<?php echo $_SESSION['errors']['selBthMonth']
?>">
Por favor selecciones su mes de nacimiento.
</span>
<span id="txtBthDayFailed"
class="<?php echo $_SESSION['errors']['txtBthDay']
?>">
Por favor ingrese su día de nacimiento.
</span>
<span id="txtBthYearFailed"
class="<?php echo $_SESSION['errors']['txtBthYear']
?>">
Por favor ingrese una fecha válida.
</span>
<br />
<!-- Email -->
<!-- <label for="txtEmail">E-mail:</label>
<input id="txtEmail" name="txtEmail" type="text"
onblur="validate(this.value, this.id)"
value="<?php echo $_SESSION['values']['txtEmail']
?>" />
<span id="txtEmailFailed"
class="<?php echo $_SESSION['errors']['txtEmail']
?>">
Invalid e-mail address.
</span>
<br />
<!-- Phone number -->
<label for="txtPhone">Número telefónico:</label>
<input id="txtPhone" name="txtPhone" type="text"
onblur="validate(this.value, this.id)"
value="<?php echo $_SESSION['values']['txtPhone']
?>" />
<span id="txtPhoneFailed"
```

```
class="<?php echo $_SESSION['errors']['txtPhone']
?>">
Por favor insertar una número telefónico valido (xxx-xxxx-xxxx).

</span>
<br />
<!-- Leer los terminos de los checkbox -->
<input type="checkbox" id="chkReadTerms"
name="chkReadTerms" class="left"
onblur="validate(this.checked, this.id)"
<?php if ($_SESSION['values']['chkReadTerms'] ==
'on') echo 'checked="checked"' ?> />
Tengo que leer los términos de uso
<span id="chkReadTermsFailed"
class="<?php echo $_SESSION['errors']
['chkReadTerms'] ?>">
Por favor asegúrese de leer los términos de uso.
</span>
<!-- End of form -->
<hr />
<span class="txtSmall">Nota: Todos los campos son requeridos.
</span>
<br /><br />
<input type="submit" name="submitButton" value="Registro"
class="left button" />
</form>
</fieldset>
</body>
</html>
```

4. Crear un nuevo archivo llamado allok.php, y agregue el siguiente código:

```
<?php
// limpiar cualquier dato guardado en la sesión
session_start();
session_destroy();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Validación de formulario con AJAX</title>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
<link href="validate.css" rel="stylesheet" type="text/css" />
</head>
<body>
Registro Satisfactorio!<br />
<a href="index.php" title="Go back">&lt;&lt; ir atrás</a>
</body>
</html>
```

5. Crear un archivo llamado validate.js. Este archivo funciona del lado del cliente, incluyendo las peticiones y demandas de AJAX.

```
// mantener la dirección del servidor remoto
var serverAddress = "validate.php";
// Cuando un error sea verdadero, desplegar los detalles del error en un mensaje
var showErrors = true;
// este es el manejador de la función para la validación de cualquier campo del formulario
function validate(inputValue, fieldID)
{
// el dato será enviado al servidor a través de POST
```

```
var data = "validationType=ajax&inputValue=" + inputValue +
"&fieldID=" + fieldID;
// construir el ambiente de objeto para XmlHttp
var settings =
{
url: serverAddress,
type: "POST",
async: true,
complete: function (xhr, response, status)
{
if (xhr.responseText.indexOf("ERRNO") >= 0
|| xhr.responseText.indexOf("error:") >= 0
|| xhr.responseText.length == 0)
{
alert(xhr.responseText.length == 0 ? "Server error." : response);
}
result = response.result;
fieldID = response.fieldid;
// encontrar el elemento HTML que desplegará el error
message = document.getElementById(fieldID + "Failed");
// mostrar el error u ocultar el error
message.className = (result == "0") ? "error" : "hidden";
},
data: data, showErrors: showErrors
};
// hacer una petición al servidor para validar la entrada de datos
var xmlhttp = new XmlHttp(settings);
}
// poner el foco en el primer campo del formulario
function setFocus()
{
document.getElementById("txtUsername").focus();
}
```

6. Es hora de agregar la lógica del lado del servidor. Iniciando con la creación de config.php, el cual tiene el siguiente código:

```
<?php
// definiendo la conexión con la base de datos
define('DB_HOST', 'localhost');
define('DB_USER', 'root');
define('DB_PASSWORD', '');
define('DB_DATABASE', 'test');
?>
```

7. Los manejadores de script de las peticiones de los clientes, y también los manejadores de las validaciones del método submit del formulario están en validate.php:

```
<?php
// iniciando sesión PHP
session_start();
// cargar el script manejador de errores y la clase validación
require_once('error_handler.php');
require_once('validate.class.php');
// Crear un nuevo objeto validador
$validator = new Validate();
// leer el tipo de validación (¿PHP o AJAX?)
$validationType = "";
if (isset($_POST['validationType']))
{
$validationType = $_POST['validationType'];
}
// Validación AJAX o validación PHP?
```

```
if ($validationType == 'php')
{
    // PHP la validación es realizada por el método ValidatePHP,
    // cuando se devuelve la página el visitante debería ser dirigido a
    //( allok.php si todos los datos son validos, o regresado a
    //index.php si no lo son)
    header('Location:' . $validator->ValidatePHP());
}
else
{
    // AJAX la validación es manejada por el método ValidateAJAX.
    // Los resultados son usados para formar un documento JSON que se envía
    // de regreso al cliente
    $response = array('result' => $validator->ValidateAJAX ($_POST['inputValue'],$_POST['fieldID']),
'fieldid' => $_POST['fieldID'] );
    // genera la respuesta
    if(ob_get_length()) ob_clean();
    header('Content-Type: application/json');
    echo json_encode($response);
}
?>
```

8. La clase que apoya la funcionalidad de validación se llama Validate, y se organiza en un archivo script llamado validate.class.php que mostramos aquí:

```
<?php
// cargar el manejador de errores y la configuración de la base de datos
require_once ('config.php');
// Los apoyos de la clase AJAX y PHP para validar un formulario web
class Validate
{
    // almacenar la conexión a la base de datos
    private $mMySQLi;
    // abriendo la conexión con la base de datos en el constructor
    function __construct()
    {
        $this->mMySQLi = new MySQLi(DB_HOST, DB_USER, DB_PASSWORD,
DB_DATABASE);
    }
    // el destructor cierra la conexión con la database
    function __destruct()
    {
        $this->mMySQLi->close();
    }
    // los soportes a la validación con AJAX, verifican un solo valor
    public function ValidateAJAX($inputValue, $fieldID)
    {
        // revise que el campo está validándose y que se realiza la validación
        switch($fieldID)
        {
            // Chequea si el nombre de usuario es válido
            case 'txtUsername':
                return $this->validateUserName($inputValue);
                break;
            // Cheue si el nombre es valido
            case 'txtName':
                return $this->validateName($inputValue);
                break;
            // Cheque si el genero fue seleccionado
            case 'selGender':
                return $this->validateGender($inputValue);
                break;
            // Cheque si el mes de nacimiento es valido
        }
    }
}
```

```
case 'selBthMonth':
return $this->validateBirthMonth($inputValue);
break;
// Chequea si el día de nacimiento es valido
case 'txtBthDay':
return $this->validateBirthDay($inputValue);
break;
// Chequea si el año de nacimiento es válido
case 'txtBthYear':
return $this->validateBirthYear($inputValue);
break;
/* Chequea si el correo electrónico es valido
case 'txtEmail':
return $this->validateEmail($inputValue);
break; */
// Chequea si el número telefónico es valido
case 'txtPhone':
return $this->validatePhone($inputValue);
break;
// Chequea si "Yo he leído las condiciones" si el checkbox ha sido verificado
case 'chkReadTerms':
return $this->validateReadTerms($inputValue);
break;
}
}
// validar todos los campos en el formulario con la acción submit
public function ValidatePHP()
{
    // etiqueta de error, devuelve 1 cuando un error es encontrado.
    $errorsExist = 0;
    // limpiar los errors de las etiquetas de sesiones
    if (isset($_SESSION['errors']))
        unset($_SESSION['errors']);
    // Por defecto, todos los campos son considerados validos
    $_SESSION['errors']['txtUsername'] = 'hidden';
    $_SESSION['errors']['txtName'] = 'hidden';
    $_SESSION['errors']['selGender'] = 'hidden';
    $_SESSION['errors']['selBthMonth'] = 'hidden';
    $_SESSION['errors']['txtBthDay'] = 'hidden';
    $_SESSION['errors']['txtBthYear'] = 'hidden';
    $_SESSION['errors']['txtEmail'] = 'hidden';
    $_SESSION['errors']['txtPhone'] = 'hidden';
    $_SESSION['errors']['chkReadTerms'] = 'hidden';
    // Validar el nombre de usuario
    if (!$this->validateUserName($_POST['txtUsername']))
    {
        $_SESSION['errors']['txtUsername'] = 'error';
        $errorsExist = 1;
    }
    // Validar nombre
    if (!$this->validateName($_POST['txtName']))
    {
        $_SESSION['errors']['txtName'] = 'error';
        $errorsExist = 1;
    }
    // Validar genero
    if (!$this->validateGender($_POST['selGender']))
    {
        $_SESSION['errors']['selGender'] = 'error';
        $errorsExist = 1;
    }
    // Validar mes de nacimiento
    if (!$this->validateBirthMonth($_POST['selBthMonth']))
    {
```



```
        $_SESSION['errors']['selBthMonth'] = 'error';
        $errorsExist = 1;
    }
    // Validar día de nacimiento
    if (!$this->validateBirthDay($_POST['txtBthDay']))
    {
        $_SESSION['errors']['txtBthDay'] = 'error';
        $errorsExist = 1;
    }
    // Validar año y fecha de nacimiento
    if (!$this->validateBirthYear($_POST['selBthMonth'] . '#' . $_POST['txtBthDay'] . '#' .
$_POST['txtBthYear']))
    {
        $_SESSION['errors']['txtBthYear'] = 'error';
        $errorsExist = 1;
    }
    // Validar correo electrónico
    /*if (!$this->validateEmail($_POST['txtEmail']))
    {
        $_SESSION['errors']['txtEmail'] = 'error';
        $errorsExist = 1;
    }*/
    // Validar número de teléfono
    if (!$this->validatePhone($_POST['txtPhone']))
    {
        $_SESSION['errors']['txtPhone'] = 'error';
        $errorsExist = 1;
    }
    // Validar lectura de términos
    if (!isset($_POST['chkReadTerms']) || !$this->validateReadTerms($_POST['chkReadTerms'])) {
        $_SESSION['errors']['chkReadTerms'] = 'error';
        $_SESSION['values']['chkReadTerms'] = "";
        $errorsExist = 1;
    }
    // Si no se encuentran errores, apuntan a una validación exitosa de la página
    if ($errorsExist == 0)
    {
        return 'allok.php';
    }
    else
    {
        // Si se encuentran errores, guardo la entrada del usuario actual
        foreach ($_POST as $key => $value)
        {
            $_SESSION['values'][$key] = $_POST[$key];
        }
        return 'index.php';
    }
}
// Para validar el nombre de usuario (debe esta vacío, y no debe estar registrado)
private function validateUserName($value)
{
    // recortar y escapar el valor de la entrada
    $value = $this->mMySQLi->real_escape_string(trim($value));
    // si el nombre de usuario esta vacío, es invalido
    if ($value == null)
        return 0; // not valid
    // comprobar si el nombre de usuario existe en la base de datos
    // $query = $this->mMySQLi->query('select user_name FROM user' 'WHERE user_name="" .
$value . "'");
    $query = $this->mMySQLi->query('select user_name FROM user' and 'WHERE
user_name="" . $value . "'");
    if ($this->mMySQLi->affected_rows > 0)
```

```
        return '0'; // not valid
    else
        return '1'; // valid
    }
    // validar nombre
    private function validateName($value)
    {
        // recortar y escapar el valor de la entrada
        $value = trim($value);
        // si el nombre de usuario está vacío, no es valido
        if ($value)
            return 1; // valido
        else
            return 0; // invalido
    }
    // validar genero
    private function validateGender($value)
    {
        // el usuario debe tener un género
        return ($value == '0') ? 0 : 1;
    }
    // validar mes de nacimiento
    private function validateBirthMonth($value)
    {
        // el mes no debe ser nulo, y estar entre 1 y 12
        return ($value == "" || $value > 12 || $value < 1) ? 0 : 1;
    }
    // validar día de nacimiento
    private function validateBirthDay($value)
    {
        // el día no debe ser nulo, y estar entre 1 y 31
        return ($value == "" || $value > 31 || $value < 1) ? 0 : 1;
    }
    // validar el año de nacimiento y la fecha en todo
    private function validateBirthYear($value)
    {
        // validar año de nacimiento su esta entre 1900 y 2000
        // obtener toda la fecha (mm#dd#yyyy)
        $date = explode('#', $value);
        // la fecha no puede ser válida si no hay un día, mes o año
        if (!$date[0]) return 0;
        if (!$date[1] || !is_numeric($date[1])) return 0;
        if (!$date[2] || !is_numeric($date[2])) return 0;
        // comprobar la fecha
        return (checkdate($date[0], $date[1], $date[2])) ? 1 : 0;
    }
    // validar correo electrónico
    /*private function validateEmail($value)
    {
        valid email formats: *@*.*, *@*.*.*, **@*.*, **@*.*.*
        return (!preg_match('/^[a-z0-9-]+\.[a-z0-9-]+@[a-z0-9-]+\.[a-z0-9-]+(\.[a-z]{2,3})$/i', $value)) ?
0 : 1;
    }*/
    // validar teléfono
    private function validatePhone($value)
    {
        // validar formato de teléfono: ###-####-####
        return (!preg_match('/^[0-9]{3}-[0-9]{3}-[0-9]{4}$/i', $value)) ? 0 : 1;
    }
    // comprobar que el usuario ha leído las condiciones de uso
    private function validateReadTerms($value)
    {
        // valor válido, es verdadero
        return ($value == 'true' || $value == 'on') ? 1 : 0;
    }
```

```
}  
?  
>
```

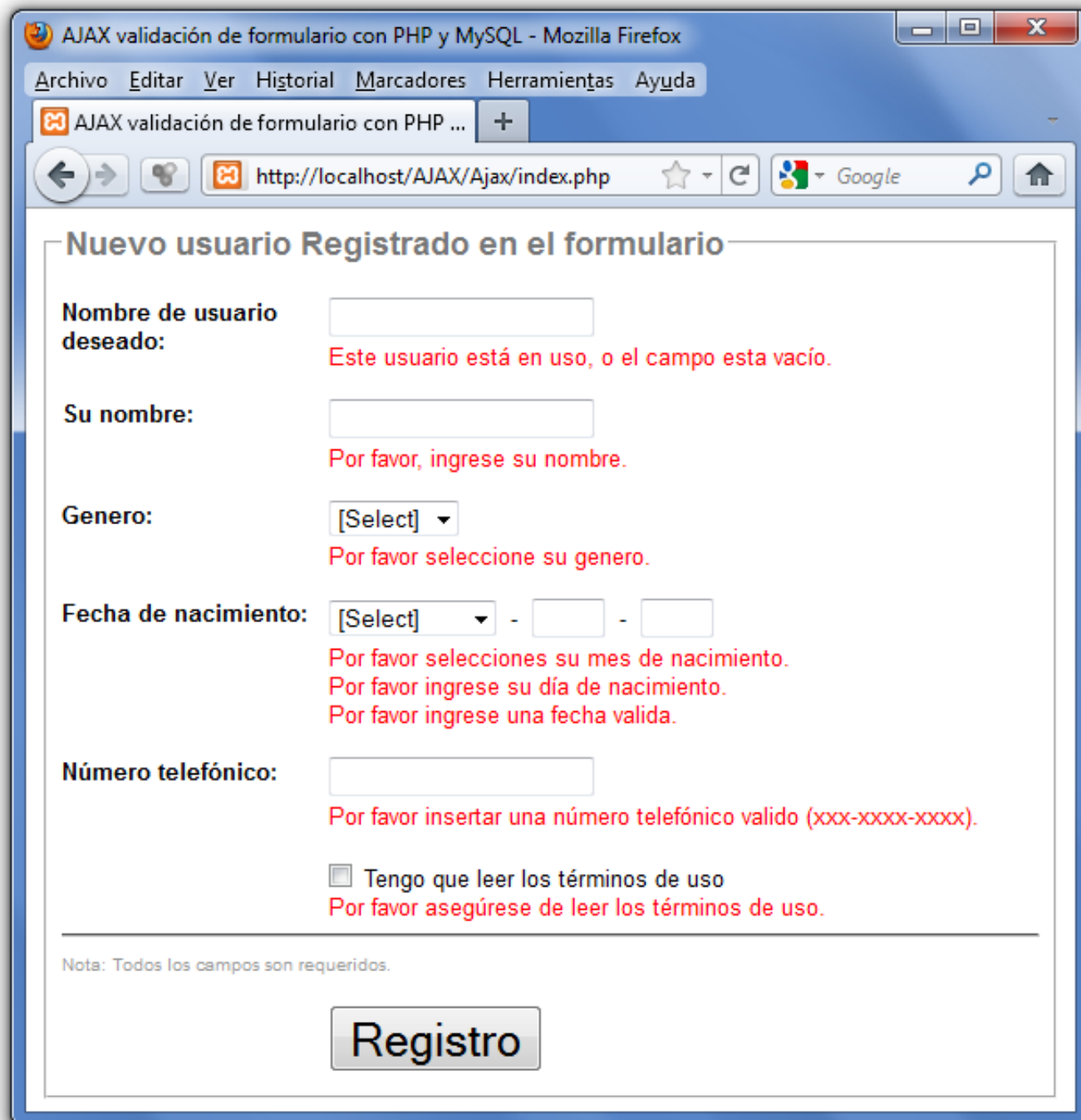
9. Falta poner a prueba el ejercicio cargando la página <http://localhost/ajax/validate/index.php> en un navegador web. El resultado debería ser el siguiente:



The screenshot shows a Mozilla Firefox browser window with the title "AJAX validación de formulario con PHP y MySQL - Mozilla Firefox". The address bar shows the URL "http://localhost/AJAX/Ajax/". The page content is a registration form with the following fields and elements:

- Nombre de usuario deseado:** A text input field.
- Su nombre:** A text input field.
- Genero:** A dropdown menu with "[Select]" as the selected option.
- Fecha de nacimiento:** A dropdown menu with "[Select]" as the selected option, followed by two empty text input fields for day and month.
- Número telefónico:** A text input field.
- ☐ **Tengo que leer los términos de uso**
- Nota:** Todos los campos son requeridos.
- Registro** button.

Ilustración 6.10 Página de registro.



The screenshot shows a Mozilla Firefox browser window titled "AJAX validación de formulario con PHP y MySQL". The address bar shows the URL "http://localhost/AJAX/Ajax/index.php". The page content is titled "Nuevo usuario Registrado en el formulario". It contains a registration form with the following fields and validation messages:

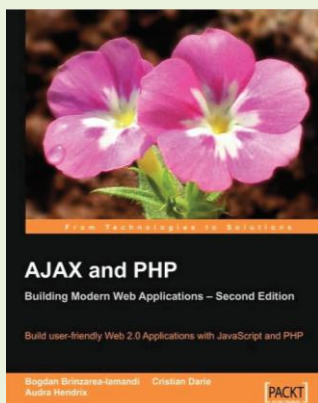
- Nombre de usuario deseado:** A text input field with a red message below it: "Este usuario está en uso, o el campo esta vacío."
- Su nombre:** A text input field with a red message below it: "Por favor, ingrese su nombre."
- Genero:** A dropdown menu with "[Select]" and a red message below it: "Por favor seleccione su genero."
- Fecha de nacimiento:** A date selection field with "[Select]" for the month and two empty boxes for the day and year. Red messages below it read: "Por favor selecciones su mes de nacimiento.", "Por favor ingrese su día de nacimiento.", and "Por favor ingrese una fecha valida."
- Número telefónico:** A text input field with a red message below it: "Por favor insertar una número telefónico valido (xxx-xxxx-xxxx)."
- ☐ **Tengo que leer los términos de uso**
Red message below it: "Por favor asegúrese de leer los términos de uso."

At the bottom, there is a note: "Nota: Todos los campos son requeridos." and a large "Registro" button.

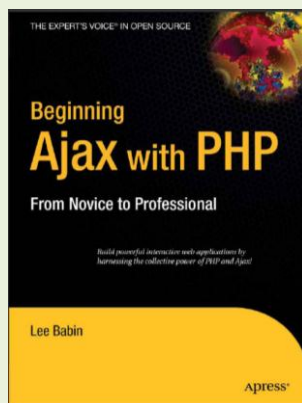
Ilustración 6.11 Solicitud de rellenar los campos.

Bibliografía del Capítulo 6

AJAX y PHP



Beginning Ajax with PHP



Ajax Bible





Capítulo 7

MODELO VISTA CONTROLADOR Y FRAMEWORKS

7 MVC

El patrón de diseño **Modelo Vista Controlador** o **MVC** describe una forma, muy utilizada en el Web, de organizar el código de una aplicación separando los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Modelo - Componente encargado del acceso a datos.

Vista - Define la interfaz de usuario, HTML+CSS... enviados en el navegador.

Controlador - Responde a eventos y modifica la vista y el modelo.

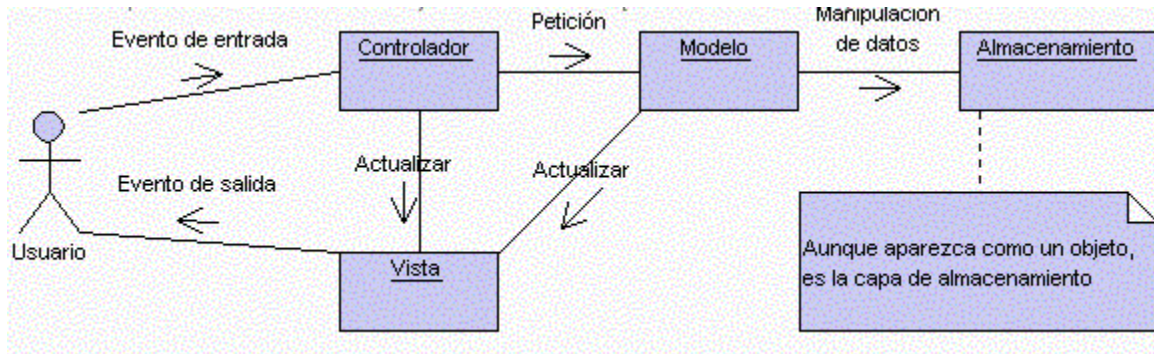


Ilustración 7.1 Elementos del MVC.

Un modelo puede tener diversas vistas, cada una con su correspondiente controlador. Un ejemplo clásico es el de la información de una base de datos, que se puede presentar de diversas formas: diagrama de tarta, de barras, tabular, etc. Veamos cada componente:

1. El **modelo** es el responsable de:
 - Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
 - Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
 - Lleva un registro de las vistas y controladores del sistema.
 - Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero bath que actualiza los datos, un temporizador que desencadena una inserción, etc).
2. El **controlador** es responsable de:
 - Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
 - Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega(nueva_orden_de_venta)".
3. Las **vistas** son responsables de:
 - Recibir datos del modelo y los muestra al usuario.
 - Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
 - Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

La principal ventaja de esta separación reside en la facilidad para realizar cambios en la aplicación puesto que:

- 1 Cuando realizamos un cambio de bases de datos, programación o interfaz de usuario solo tocaremos uno de los componentes.
- 2 Podemos modificar uno de los componentes sin conocer cómo funcionan los otros.

Un ejemplo de MVC con un modelo pasivo (aquel que no notifica cambios en los datos) es la navegación web, que responde a las entradas del usuario, pero no detecta los cambios en datos del servidor.

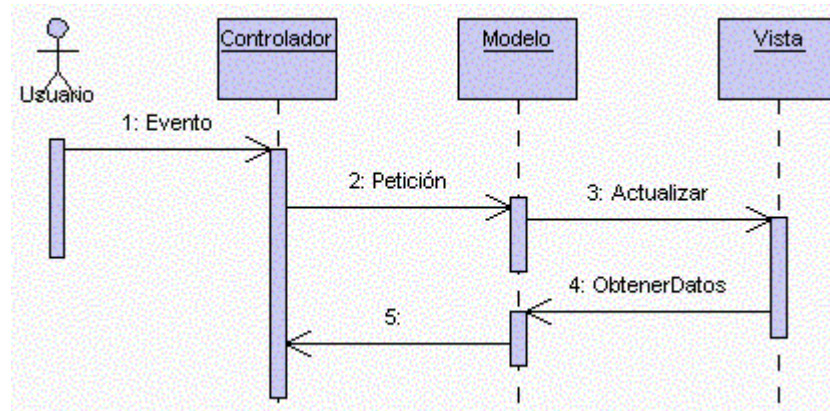


Ilustración 7.2 Esquema MVC pasivo.

Bien, pero esto ¿cómo se implementa? Existe una pequeña dificultad: la mayor parte de las herramientas de desarrollo incorporan en las clases de la vista gran parte o todo el procesamiento de eventos. Con lo que el controlador queda semioculto dentro de la vista. A pesar de ello, podemos acercarnos bastante al patrón. En el siguiente ejemplo en Java, el objeto vista es un Applet AWT. El controlador (controlador.java) puede gestionar el clic en un botón, de tal forma que recoge datos por medio del Modelo (model.cargar_texto(..)) y los manda a la Vista (el applet) para su actualización (vista.mostrar_texto()):

```

/*****
Responde al click en botón "abrir"
La respuesta al evento es hacer que se abra en la vista
el archivo correspondiente a la referencia seleccionada en el combo box
*****/

void b_abrir_actionPerformed(ActionEvent e) {
    ...
    String texto_archivo = model.cargar_texto( indice_ref ); // Obtener texto de archivo
    /** Si la carga de archivo es ok, lo muestro. Si no, aviso de error ****/
    if (texto_archivo != null) {
        vista.mostrar_texto(texto_archivo); // Mostrar texto
        vista.mostrar_aviso("Carga de " + path + " completada.");
    }
    else
        vista.mostrar_aviso("Error en la carga de " + path);
}

```

El modelo vista controlador de mayor uso en las aplicaciones Web mundiales de hoy en día, se usaron la primera vez en Smalltak y desde entonces se han popularizado para java. En la actualidad hay mas de una docena de aplicaciones web basados en el modelo MVC.

Este código es lo necesario para cumplir con el proposito de listar los artículos

```

<?
// conexion a la base de datos
$cn = MySQL_connect('localhost', 'usuario', 'clave');

```



```
MySQL_select_db('db', $cn);

// consulta SQL para traer los artículos
$resultado=MySQL_query('select fecha, título FROM articulo', $cn);
?>
<h1>Listado de Artículos</h1>
<table>
  <tr> <th>Fecha</th> <th>Titulo</th> </tr>
<?php
```

// recorre el resultado pintando la fecha y titulo

```
while ($fila = MySQL_fetch_array($resultado, MYSQL_ASSOC))
{
  echo "<tr>";
  echo "<td> ".$fila ['fecha'] ." </td>";
  echo "<td> ".$fila ['titulo'] ." </td>";
  echo "</tr>";
}
?>
</table>

<?php
//cerramos la conexion
MySQL_close($cn);
?>
```

Observaciones

- En la misma página nos conectamos al servidor (*solo funcionara si es MySQL*) y seleccionamos una base de datos.
- Realizamos una consulta (*no hay manejo de errores y/o excepciones*) .
- Pintamos el código HTML (*combinamos código HTML dentro del código PHP, dejando los tags (<tr>, <td>) ilegibles para la persona que implemente los estilos*) .

Utilizando el Controlador y la Vista

El ejemplo anterior los separaremos en dos archivos, uno se llamara controlador y el otro vista.

controlador.php

```
<?
$cn = MySQL_connect('localhost', 'usuario', 'clave');
MySQL_select_db('db', $cn);

$resultado = MySQL_query('select fecha, título FROM articulo', $cn);
$articulos = array();

while($articulo = MySQL_fetch_assoc($resultado))
{
  $articulos [] = $articulo;
}

MySQL_close();

require('vista.php');
?>
```

vista.php

```
<h1>Listado de Artículos</h1>
<table>
  <tr> <th>Fecha</th> <th>Titulo</th> </tr>
<?php foreach($articulos as $articulo): ?>
```

```
<tr>
  <td><?php echo $articulo ['fecha'] ?></td>
  <td><?php echo $articulo ['titulo'] ?></td>
</tr>
<?php endforeach;?>
</table>
```

De esta manera tenemos separada en el controlador.php casi todo el código php con la lógica de negocios, mientras que en vista.php solo recorremos un array con datos.

Lo recomendable en la vista seria utilizar instrucciones PHP mas conocidas, como: **if**, **endif**, **foreach**, **endforeach** . Y no utilizarlas con llaves “{” , ya que dificulta reconocer donde empieza y termina. Otro problema seria si queremos utilizar nuevamente el listar articulo en otra página, tendríamos que reescribir el controlador.php de nuevo.

Para esto separaremos el controlador (*controlador.php*) en modelo.php y controlador.php.

Utilizando el Modelo

modelo.php

```
<?php

function getArticulos()
{
    $cn = MySQL_connect('localhost', 'usuario', 'clave');
    MySQL_select_db('db', $cn);

    $resultado = MySQL_query('select fecha, título FROM articulo', $cn);
    $articulos = array();

    while($articulo = MySQL_fetch_assoc($resultado))
    {
        $articulos [] = $articulo;
    }

    MySQL_close();
}
?>
```

controlador.php

```
require('modelo.php');

$articulos = getArticulos();

require('vista.php');
```

Después de esta separación el controlador quedaría tan solo como un agente para pasar datos del modelo hacia la vista, pero en aplicaciones mas complejas el controlador es quien realiza las tareas de autenticación de usuarios, manejo de sesiones, filtrar y validar entradas de datos por GET o POST.

Cambiando el gestor de Base de Datos

La respuesta seria, cambiar todas las funciones del modelo (MySQL_connect, MySQL_query, ...) por las correspondientes, y eso nos tomaría mucho tiempo. Para hacer un mejor uso de MVC o mejor dicho cambiando el patrón un poco, se podría separar el modelo en dos capas:

- La capa de Acceso a Datos
- La capa de Abstracción de la Base de Datos

Si se diera el caso de cambiar de gestor de base de datos, solo tendríamos que actualizar la capa de abstracción de la base de datos

Ejemplo de las dos capas

Abstracción de datos

```
function crearConexion($servidor, $usuario, $clave)
{
    return MySQL_connect($servidor, $usuario, $clave);
}

function cerrarConexion($cn)
{
    MySQL_close($cn);
}

function consulta($consulta, $base_datos, $cn)
{
    MySQL_select_db($base_datos, $cn);

    return MySQL_query($consulta, $cn);
}

function getResultado($resultado, $tipo = MYSQL_ASSOC)
{
    return MySQL_fetch_array($resultado, $tipo);
}
```

Acceso a Datos

```
function getTodosLosArticulos()
{
    $cn = crearConexion('localhost', 'usuario', 'clave');

    $resultado=consulta('select fecha, título FROM articulo', 'db', $cn);

    $articulos = array();
    while ($articulo = getResultado($resultado))
    {
        $articulos [] = $articulo;
    }

    cerrarConexion($cn);

    return $articulos;
}
```

Ahora tenemos la capa de Acceso a Datos sin ninguna dependencia de funciones de algún gestor de base de datos, y podremos reutilizar las funciones de la capa de abstracción de base de datos en otras funciones de acceso a datos de nuestro modelo.

Frameworks

Un **framework** es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de *software* concretos, con base en la cual otro proyecto de [software](#) puede ser organizado y desarrollado. Típicamente, puede incluir soporte de [programas](#), [bibliotecas](#) y un [lenguaje interpretado](#) entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una [arquitectura de software](#) que modela las relaciones generales de las entidades del [dominio](#). Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Son diseñados con la intención de facilitar el desarrollo de *software*, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de *software* que tratando con los tediosos detalles de bajo nivel para proveer un sistema funcional. Por ejemplo, un equipo que usa [Apache Struts](#) para desarrollar un [sitio web](#) de un banco, puede enfocarse en cómo los retiros de ahorros van a funcionar en lugar de preocuparse de cómo se controla la navegación entre las páginas en una forma libre de errores. Sin embargo, hay quejas comunes acerca de que el uso de *frameworks* añade [código](#) innecesario y que la preponderancia de *frameworks* competitivos y complementarios significa que el tiempo que se pasaba programando y diseñando ahora se gasta en aprender a usar *frameworks*.

Fuera de las aplicaciones en la [informática](#), puede ser considerado como el conjunto de procesos y tecnologías usados para resolver un problema complejo. Es el esqueleto sobre el cual varios objetos son integrados para una solución dada. Después de todo, un *framework* no debe ser consciente de todos estos requerimientos sin tener que ser intrusivo con las aplicaciones que permite dentro de sí mismo. A esto le sumamos la capacidad de extenderse sin prejuicios para diversificar la expresión del programa mismo.

Arquitectura

Dentro de este aspecto, podemos basarnos en el modelo [MVC](#) (Controlador => Modelo => Vista) ya que debemos fragmentar nuestra [programación](#). Tenemos que contemplar estos aspectos básicos en cuanto a la implementación de nuestro sistema:

- **Controlador:**

Con este apartado podemos controlar el acceso (incluso todo) a nuestra [aplicación](#), esto pueden ser: [archivos](#), [scripts](#) o [programas](#); cualquier tipo de información que permita la [interfaz](#). Así, podremos diversificar nuestro contenido de forma dinámica, y estática (a la vez); pues, sólo debemos controlar ciertos aspectos (como se ha mencionado antes).

- **Modelo:**

Este miembro del [controlador](#) maneja las operaciones lógicas, y de manejo de información (previamente enviada por su ancestro) para resultar de una forma explicable, y sin titubeos. Cada miembro debe ser meticulosamente llamado, en su correcto nombre y en principio, con su verdadera naturaleza: el manejo de información, su complementación directa.

- **Vista:**

Al final, a este miembro de la familia le corresponde dibujar, o expresar la última forma de los datos: la interfaz gráfica que interactúa con el usuario final del programa ([GUI](#)). Después de todo, a este miembro le toca evidenciar la información obtenida hasta hacerla llegar con el controlador. Solo (e inicialmente), nos espera demostrar la información.

¿Cómo se usan los frameworks?

Los elementos (frameworks) son bastante fáciles de usar, simplemente siga los siguientes pasos:

1. Descargarlo y desempaquetarlo.

Lo primero que debemos hacer es descargar el archivo .zip y desempaquetarlo.

2. Renombrarlo.

Renombrar la carpeta del elemento por el nombre de su proyecto. Este será usualmente el nombre del sitio, pero si quieres que se vea mejor, le puedes poner el nombre codificado.

3. Comenzar a trabajar.

Iniciar trabajando en su sitio. Guarde todos sus archivos dentro de una carpeta conveniente.

4. Subir tu proyecto.

Después de haber trabajado un poco el código, es necesario cargar su sitio en un servidor web para probarlo. Usted puede tener algunas formas diferentes de manejar esto, porque podría fácilmente

cargar su sitio en un servidor local en su computadora personal. O podría cargarlo dentro de un dominio.

5. Restablezca su ruta absoluta.

Si usted usa los caminos absolutos para sus vínculos, sus vínculos interiores tendrán los iconos de los vínculos externos. Para arreglar esto usted necesita restablecer sus vínculos interiores.

Existen muchos frameworks, nosotros aquí les dejamos 12 de los más utilizados para que los pruebes y te quedes con el que más te agrade, o con ninguno y de pronto te programas el tuyo personalizado.

1. [Elements CSS Framework](#) (Source: [Project Designs](#))

2. [WYMstyle: a CSS framework](#) (Source: [Daniel Reszka](#))

3. [YAML CSS Framework](#) (Source: [High Resolution](#))

4. [YUI Grids CSS](#) (Source: [Yahoo](#))

5. [Boilerplate CSS Framework](#)

6. [Blueprint CSS](#)

7. [Schema Web Design Framework](#)

(Source: [David Golding Design](#))

8. [CleverCSS](#)

9. [Tripoli Beta CSS Framework](#)

(Source: [Monc](#))

10. [ESWAT Web Project Framework](#)

(Source: [Philip Karpiak](#))

11. [CwS CSS Framework](#)

(Source: [Content with Style](#))

12. [My \(not really mine\) CSS Framework](#)

(Source: [That Standards Guy](#))

Ahora comentaremos un poco sobre la funcionalidad de [YAML CSS Framework](#).

Hojas de estilo con YAML

YAML es un framework de desarrollo para CSS, el objetivo de utilizar esta herramienta es ayudar a los diseñadores a realizar hojas de estilo de una manera más ágil y estándar para los múltiples navegadores. YAML combina una serie de características bastante interesantes que sin duda nos harán más sencillo el trabajo a la hora de diseñar con [CSS](#).

Existen bastantes plataformas de desarrollo para diversas tecnologías. Los más sonados son los que interactúan del lado del servidor y en el nivel de programación. Pero [YAML](#) es digno de mención por que realmente nos facilita la labor de realizar hojas de estilo.

No debemos confundir el lenguaje de marcado YAML (Yet Another Markup Language) superconjunto de [JSON con YAML](#) (Yet Another Multicolumn Layout). Entre las principales características del framework están:

- Un esquema definido de plantillas que incluye una cabecera, un pie de página y un área para una o más columnas muy propio de las nuevas tendencias de diseño.
- Compatibilidad con la mayoría de los navegadores utilizando la plantilla por defecto proporcionada en la distribución.
- Esquemas de diseños fijos o fluidos.

Lo interesante de esta herramienta es que cuando el diseñador inicia a crear la hoja de estilo lo hace con un esquema ya fijado y con la compatibilidad multinavegador. Esta última característica es muy importante a mi parecer. Posteriormente y con el transcurso del diseño de la plantilla el diseñador puede ir restando elementos innecesarios dentro de la hoja de estilo. Pero es importante resaltar que YAML no es una plantilla prediseñada.

Estructura del paquete:

El paquete de distribución incluye la carpeta **YAML/** que contiene los archivos necesarios para trabajar con la herramienta. Las demás carpetas son **DOCUMENTATION/**, **EXAMPLES/** y **TOOLS/**. Dentro de la carpeta YAML están las siguientes hojas de estilo.

- **/yaml:** contiene los archivos `central_draft.css` y `markup_draft.css`. Esta carpeta es la carpeta principal del framework. El primer archivo embebe todos los componentes CSS necesarios para trabajar y el segundo muestra el código fuente de la estructura de YAML.
- **/yaml/core:** contiene los archivos `base.css`, `ie hacks.css`, `print_base.css`, `slim_base.css`, `slim_ie hacks.css` y `slim_print_base.css`. Como su nombre lo indica, estos archivos son los componentes base del framework. El primer archivo incluye toda la definición CSS para elaborar un diseño a tres columnas. El segundo (`ie hacks.css`) contiene todos los “ajustes” necesarios para la correcta visualización en el navegador Internet Explorer 5 al 7 independientemente del diseño elegido. Y el tercero proporciona soporte para CSS Print.
- **/yaml/screen:** componentes CSS para el diseño en plantilla. Incluye los archivos `basemod_draft.css` y `content_default.css`.
- **/yaml/navigation:** componentes para la definición de menús de navegación horizontal y vertical.
- **/yaml/print:** hojas de estilo con valores por defecto para la definición de la versión imprimible de la página.
- **/yaml/patches:** mas “ajustes” para Internet Explorer.

¿Cómo empezar a utilizar nuestra plantilla?

Cuando iniciamos el diseño, lo primero que debemos hacer es agregar nuestra plantilla como recurso de hoja de estilo, generalmente utilizando el atributo `link`. Esta hoja de estilo principal contiene nuestro diseño y la incorporación de las otras hojas de estilo que son parte de YAML mediante la regla `@import`.

```
<head>
...
<link href = "css/layout_3col_standard.css" rel = "stylesheet"
      type = "text/css"/>
...
</head>
/* import core styles | Basis-Stylesheets */
@import url(../yaml/core/base.css);
/* import screen layout | Screen-Layout einbinden */
@import url(../yaml/navigation/nav_shinybuttons.css);
@import url(screen/basemod.css);
@import url(screen/content.css);
/* import print layout | Druck-Layout einbinden */
@import url(../yaml/print/print_003_draft.css);
```

Reglas de oro para usar YAML correctamente

Las siguientes reglas resumen los principios básicos para el buen desarrollo con YAML:

- YAML no es una plantilla prefabricada.
- YAML está basado en el principio TOP-DOWN (iniciar el desarrollo de la hoja de estilo con definiciones de propiedades estándar para todos los navegadores).
- Todas las plantillas YAML necesitan de tres componentes básicos: base.css, iehacks.css y print_base.css.
- Se debe separar correctamente los archivos del framework, de los propios de nuestro diseño incluyendo los archivos modificados del framework.
- Lee el manual de documentación de YAML antes de iniciar el desarrollo con la herramienta.

El YAML Builder

Adicionalmente en el sitio del proyecto hay un enlace al [YAML Builder](http://www.yaml.de/en/). Esta aplicación web nos permite definir plantillas en tiempo real y de una manera más gráfica simplificando aún más el proceso de desarrollo de la hoja de estilo. Luego que hemos terminado nos permite obtener una copia de la plantilla lista para poder ser utilizada.



Ilustración 7.3 Página de YAML.

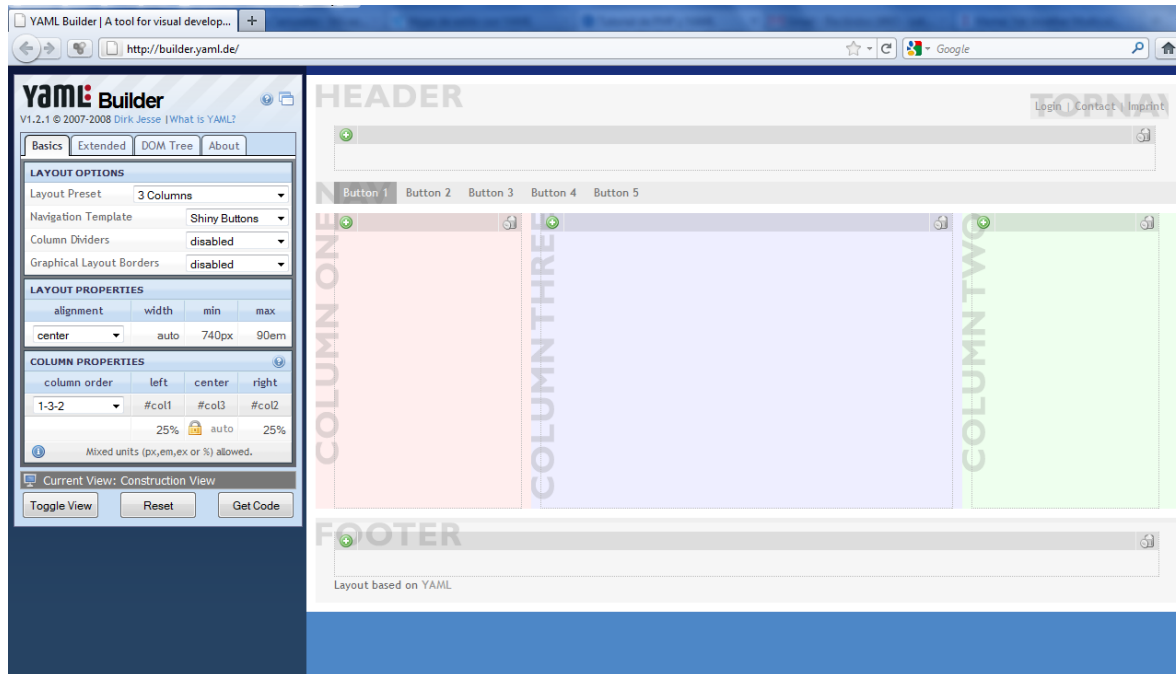


Ilustración 7.4 YAM Builder.

Bibliografía del Capítulo 7

www.top12cssframeworksandhowtounderstandthem.htm



ANEXOS



Anexo 1

Instalaciones

Creando una simulación (la maqueta) de nuestro proyecto.

Es recomendable antes de sentarse a programar o construir un sitio web, hacer uso de algunas herramientas que nos permitan como diseñadores enfocarnos en la creatividad antes que en la funcionalidad y programado de nuestras aplicaciones. El propósito de usar herramientas de diseño es que podamos figurarnos nuestra aplicación como si ya la tuviéramos terminada antes de comenzar a programarla, es importante tener bien claro que es lo que queremos crear y como queremos que se vea cuando ya esté terminada, para optimizar nuestro tiempo al momento de la creación del código para nuestras interfaces de interacción con el usuario.

Existen varias herramientas dedicadas a cumplir con este objetivo, usted puede sentirse en la libertad de utilizar cualquier aplicación que considere satisfaga sus necesidades y exigencias, para el caso de no estar muy entendido en el asunto, aquí le dejamos una pequeña lista de herramientas que puede utilizar:

- Project ROME.
- ZAM 3D.
- Expression Blend.
- Tan Blend.
- Flash.
- Sothink Quicker.

Para efecto de nuestra aplicación, hemos optado por utilizar la herramienta Expression Blend, no es una herramienta de uso libre (Requiere licencia para su debida activación), pero la hemos utilizado para crear una maqueta y simular nuestro sitio web como si estuviera terminado. Haciendo uso de herramientas de este tipo, podemos mejorar nuestras aplicaciones al momento de desarrollar nuestro código para nuestros proyectos.



Ilustración 0.1 Pantalla de inicio.

En éste software podremos modelar nuestro sitio web y podremos enfocarnos directamente en la parte de la interfaz gráfica, diseño, colores, etc., no programaremos nada aún, pero garantizaremos que el aspecto visual quede atractivo para los usuarios.

Instalando PHP, Apache y MySQL.

Uno de los primeros obstáculos para los programadores nuevos que se están iniciando, es que antes de poder escribir su primer línea de PHP, primero deben descargar Apache, PHP y MySQL por lo general, y luego luchar a través de las instrucciones de instalación que están llenos de jerga técnica que aún no pueden entender. Esta experiencia puede dejar a muchos desarrolladores sentirse inseguros de sí mismos, dudando de si han instalado el software necesario correctamente. Afortunadamente la comunidad de desarrollo ha respondido a la frustración de los principiantes, ya sea para crear aplicaciones Windows, Mac o Linux. Estas opciones incluyen todo en una solución para configurar e instalar Apache, MySQL, PHP. La más común de todo en una solución es un programa llamado "XAMPP" que combina Apache, MySQL, PHP y algunas otras aplicaciones útiles, junto a un fácil instalador. XAMPP es gratuito y está disponible para Windows, Mac y Linux, así que esta monografía lo va a utilizar para facilitar el aprendizaje de los alumnos.

La primera parte para poder ejecutar nuestras líneas de JavaScript o PHP, será obtener una copia del software XAMPP, descargando la última versión del sitio web www.apachefriends.org.

Instalación: visitar <http://www.apachefriends.org/es/xampp.html>.

Para verificar que Apache y PHP estén ejecutándose. Es un muy simple, si todos los servicios están corriendo apropiadamente sobre su Máquina, simplemente abra el navegador y ponga la siguiente dirección: <http://localhost>. Si están funcionando correctamente, te llevará directamente a la página inicial del xampp:

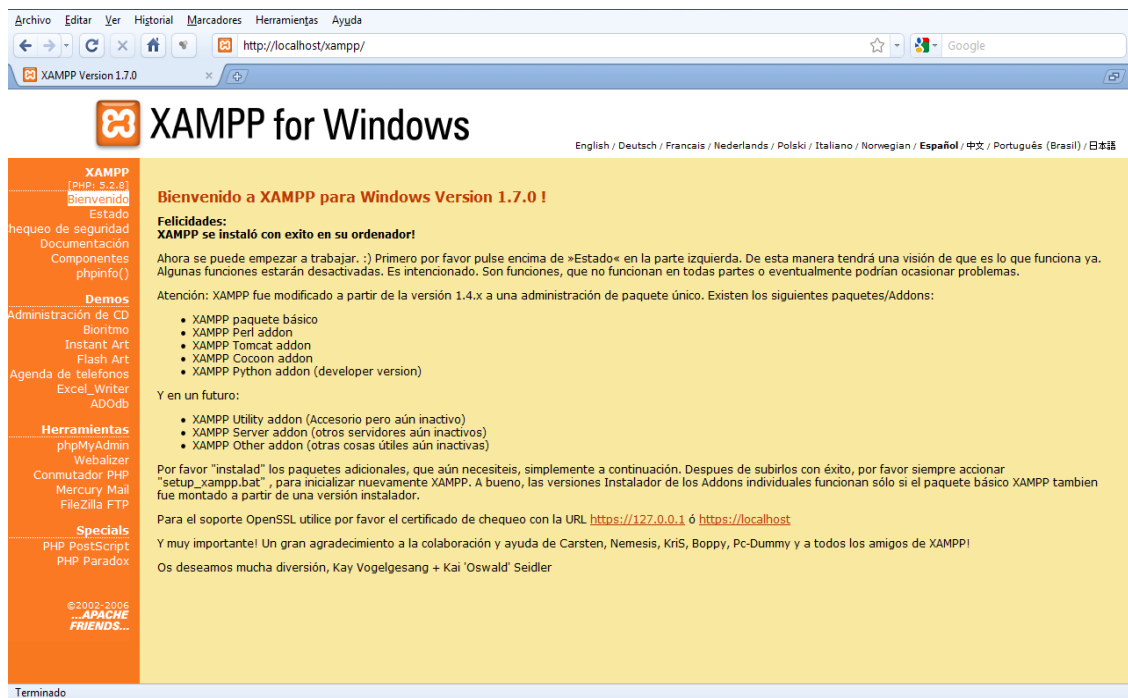


Ilustración 0.2 Página principal de XAMPP.

Si el navegador no es capaz de mostrar esta página, entonces los servicios tienen algún problema, por lo que deberá desinstalar y volver a instalar el Xampp.

Ahora que todo está en orden, solo debemos colocar nuestros archivos en la carpeta htdocs que se crea dentro de la carpeta xampp en la unidad C de nuestro ordenador. Posteriormente solo debemos acceder a ella desde nuestro navegador y listo



Anexo 2

Prácticas

XHTML5 y CSS.

Instrucciones de la práctica:

- Haber leído el capítulo 1 de la monografía.
- Se aconseja consultar al profesor sobre cualquier aspecto que no se comprenda sobre la práctica.

Objetivos

- Codificar nuestras páginas web de forma práctica, empleando los conocimientos obtenidos en el capítulo 1 de la monografía.
- Familiarizarse con el uso del Lenguaje de Marcado de Hipertexto, orientándonos a la implementación de HTML5.
- Presentar la información en bloques utilizando etiquetas DIV y SPAN, sin utilizar frames.
- Usar hojas de estilo en cascada CSS, con las que trabajaremos la apariencia de nuestro sitio web, sin alterar la estructura del sitio.

La Práctica

Se necesita crear un Sistema de Gestión Académica para un centro escolar, la aplicación debe funcionar vía web como cliente – servidor este sitio lo iremos mejorando a medida que avancemos con las siguientes prácticas. En principio, el sitio web debe contener:

- Una página principal llamada index.html: Esta contiene información general del centro escolar, con un menú que le permita a los visitantes desplazarse por el sitio.
 - Un formulario para Ingresar al sistema: Los Administradores del centro, Docentes y Alumnos, podrán acceder al sistema con sus respectivos login y password, se debe hacer el formulario, pero su programación se realizará en los capítulos subsiguientes.
1. Se le provee de la información necesaria que debe contener cada módulo de nuestro sitio (imágenes, código fuente de la página index.html a la que no hay necesidad de realizarle modificación alguna porque al cargarse y combinarse con la hoja de estilo estilos.css dan como resultado la página de la **ilustración 0.3**).
 2. También se le facilita el layout de las hojas de estilo para que usted las complete sin alterar el bosquejo original del sitio, estas hojas de estilo deben definirse en el código de la página index.html.
 3. Apoyarse en la presentación hecha en Expression Blend proporcionada para facilitar la creación de las páginas del sitio, usted debe crear el contenido de cada página a como se muestran en las ilustraciones sub siguientes.
 4. Tener en cuenta que no se está pidiendo la programación de nuestra aplicación, apenas estamos haciendo la interfaz gráfica, el entorno visual del sitio.
 5. Se deben probar las páginas desde el navegador de su pc, alojadas en un servidor local. Las páginas se deben acceder de esta manera: <http://localhost/Sistema-Académico/index.html>.
 6. El tiempo máximo que debe tomarse en la elaboración de la aplicación no debe exceder las 12 horas.

Requerimientos:

- ✓ Un editor de texto plano (Notepad++, Bloc de Notas, etc.).
- ✓ Un Navegar Web. (Explorer, Mozilla, Opera, etc.)
- ✓ Software que nos permita visualizar nuestras páginas desde un servidor, para el caso nuestro hemos escogido XAMPP que incorpora Apache.

Este es el modelo de nuestra página inicial:

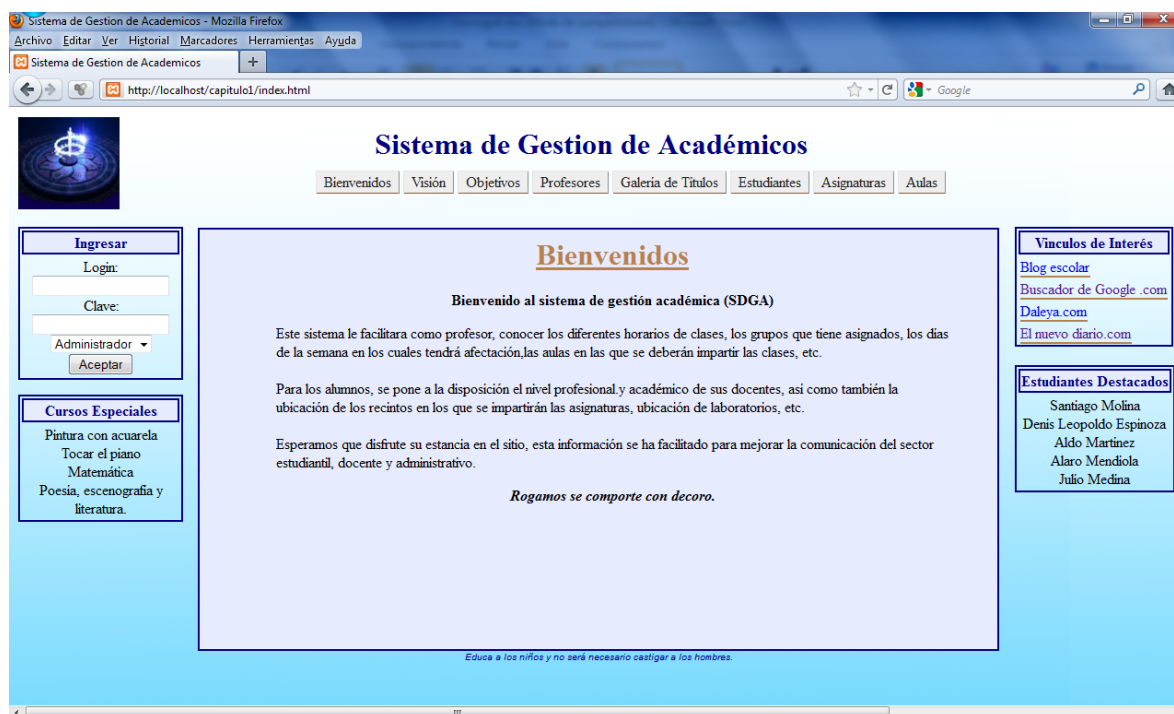


Ilustración 0.3 Página index.html.

Estructuralmente, nuestra página tiene los siguientes elementos:

- Una cabecera: El primer div, contiene una imagen de fondo y en el centro la frase “**SITEMA DE GESTIÓN ACADÉMICA.**”
- El menú: Es un div, conteniendo las diferentes opciones de nuestro menú. También contiene una imagen al extremo izquierdo (el logo o Insignia del colegio).
- Una columna lateral izquierda: Es un div y tiene la propiedad float:left, width:14% , contiene dos tablas con dos filas cada una (Una para el encabezado y la otra para el contenido). En la primera se ha colocado un campo para el login y password del usuario. En la segunda se han colocado algunos cursos especiales que el colegio está ofreciendo.
- Una columna central: Es un div que se utiliza para mostrar el contenido de cada una de las opciones del menú. El ancho del div es de margin-left:16%, margin-right:16%, height:70%.
- Una columna lateral derecha: Es un div, con un ancho de width:14%. En su interior posee dos tablas, en la primera están los vínculos a otros sitios de interés como el blog escolar, la segunda es un blog donde puedes compartir tu opinión sobre el colegio, sus docentes y demás.
- El pie de página que es un div que contiene únicamente la frase “Educa a los niños y no será necesario castigar a los hombres”.

Estructura de la página web

La distribución de los elementos sobre esta página tal y como se muestra en la imagen anterior, se ven más claramente en la siguiente ilustración:

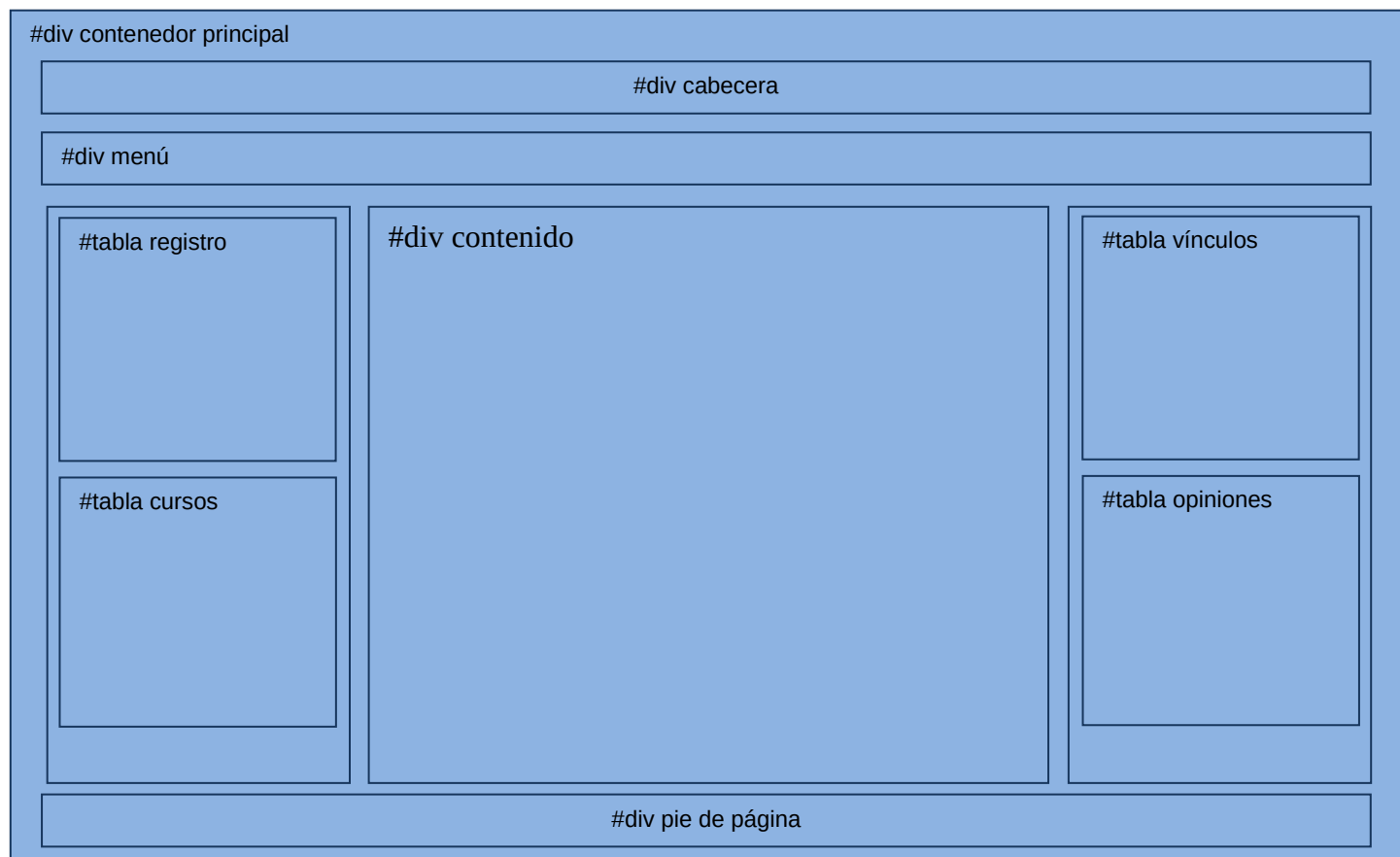


Ilustración 0.4 Estructura del sitio.

Nota: Recuerde que los div registro, contenido y destacados tienen su propiedad float:left y un ancho definido para lograr que se visualicen de la manera que se pide

Página Bienvenidos.

Bienvenidos

Bienvenido al sistema de gestión de académicos (SDGA)

Este sistema le facilitará como profesor, conocer los diferentes horarios de clases, los grupos que tiene asignados, los días de la semana en los cuales tendrá afectación, las aulas en las que se deberán impartir las clases, etc.

Para los alumnos, se pone a la disposición el nivel profesional y académico de sus docentes, así como también la ubicación de los recintos en los que se impartirán las asignaturas, ubicación de laboratorios, etc.

Esperamos que disfrute su estancia en el sitio, esta información se ha facilitado para mejorar la comunicación del sector estudiantil, docente y administrativo.

Rogamos se comporte con decoro.

Página Visión.

Vision

Esta Universidad se ha catalogado con el lema: "A la libertad Por la Universidad". Es por esto que se creó este sistema, para que todos con libertad podamos acceder a la información de mayor interés para cada uno de nosotros.

Se espera que con la implementación de este sistema, se mejoren los tiempos de asistencia a clases por parte de los docentes y alumnos, la comunicación y por ende la calidad de la educación.

No olviden que una de las siete normas de la calidad sita:
"Llegar a tiempo es calidad."

Página Objetivos.

Objetivos

Objetivo General.

Realizar un programa de prácticas resueltas usando diferentes tecnologías web actuales y documentar la realización de dichas prácticas

Objetivos Específicos

Documentar con ejemplos prácticos, las características de los lenguajes de marcado y de programación orientados a la web actuales y usados en este trabajo. Proponer un formato de prácticas para el desarrollo de las mismas.

Desarrollar un programa de prácticas de laboratorio y resolver dichas prácticas documentando debidamente la solución.

Página Profesores.



The screenshot shows a web browser window with the address bar displaying 'http://localhost/Capitulo1/index.html#'. The page title is 'Sistema de Gestion de Académicos'. The main navigation menu includes: Bienvenidos, Visión, Objetivos, Profesores, Galería de Títulos, Estudiantes, Asignaturas, and Aulas. The 'Profesores' menu item is selected.

Sistema de Gestion de Académicos

Profesores Titulares

Director
Sub-Director
Encargado de disciplina
Encargado de disciplina Lengua y Literatura
Matemáticas
Matemáticas
Geografía e Historia
Geografía e Historia
Ciencias Sociales
Ciencias Sociales
Psicólogo
Psicólogo
Musica
Dibujo y artes plasticas

MSC. Erian Joel Tercero Velasquez.
MSC. Daniel Enrique Pérez.
Lic. Lester Antonio Meraz.
Lic. Denis Fernando Rosales Blanco.
Lic. Ninfa del Rosario Guardado Vivas.
Lic. Rolando Marin Cuadra.
Lic. María de Jesús Cath.
Lic. Milton Alexander Voquín Arteaga.
Lic. Julia Rosa Santos Picado.
Lic. Reynaldo Francisco Quintero Manzanares.
Lic. Darling María Rojas Manzanares.
Lic. Donald Ramón Delgado Tórrez.
Lic. Miguel Faván Parrales García.
Sr. Juan Alvaro Delgado Tórrez.
Lic. Ángel Rafael Centeno Vargas.

Vinculos de Interés
[Blog escolar](#)
[Buscador de Google .com](#)
[Daleya.com](#)
[El nuevo diario.com](#)

Estudiantes Destacados
Santiago Molina
Denis Leopoldo Espinoza
Aldo Martínez
Alaro Mendiola
Julio Medina

Ingresar
Login:
Clave:
Administrador
Aceptar

Cursos Especiales
Pintura con acuarela
Tocar el piano
Matemática
Poesía, escenografía y literatura.

Educa a los niños y no será necesario castigar a los hombres.

Página Estudiantes.



The screenshot shows the same web browser window as the previous page, but the 'Estudiantes' menu item is selected. The main content area displays 'Estudiantes con reconocimientos' with two images: a football team and a portrait of a man.

Sistema de Gestion de Académicos

Estudiantes con reconocimientos

Equipo de futbol Poesia

Vinculos de Interés
[Blog escolar](#)
[Buscador de Google .com](#)
[Daleya.com](#)
[El nuevo diario.com](#)

Estudiantes Destacados
Santiago Molina
Denis Leopoldo Espinoza
Aldo Martínez
Alaro Mendiola
Julio Medina

Ingresar
Login:
Clave:
Administrador
Aceptar

Cursos Especiales
Pintura con acuarela
Tocar el piano
Matemática
Poesía, escenografía y literatura.

Educa a los niños y no será necesario castigar a los hombres.

Página Asignaturas.



The screenshot shows a web browser window with the URL `http://localhost/Capitulo1/index.html#`. The page title is "Sistema de Gestion de Académicos". The navigation menu includes: Bienvenidos, Visión, Objetivos, Profesores, Galería de Títulos, Estudiantes, Asignaturas, and Aulas. The main content area is titled "Asignaturas" and displays a grid of subjects for each grade from 1° to 6°.

1° Grado	2° Grado	3° Grado	4° Grado	5° Grado	6° Grado
Español	Español	Español	Español	Español	Español
Matemática	Matemática	Matemática	Matemática	Matemática	Matemática
Artes plásticas	Música	Historia1	Historia2	Geografía1	Geografía2
Dibujo	Dibujo	Danza	Música y Canto1	Música y Canto2	Música y Canto3
Inglés	Inglés1	Inglés2	Inglés3	Inglés4	Inglés5
			canto	Geografía	Historia Nacional
			Danza	Ciencias Naturales1	Ciencias Naturales2
				Danza	Danza

On the right side, there is a "Vinculos de Interés" section with a list of names: Santiago Molina, Denis Leopoldo Espinoza, Aldo Martínez, Alaro Mendiola, and Julio Medina.

Página Aulas.



The screenshot shows the same web application with the "Aulas" page selected. The main content area is titled "Aulas" and displays a grid of six small images showing students in a classroom setting.

The navigation menu and the "Vinculos de Interés" section on the right are identical to the previous page.

Formulario:

Ingresar

Login:

Clave:

Aceptar

```
body {
    width: expression(document.body.clientWidth > 601? "600px": "auto");
    margin: 0;
    padding: 0;
    background-image: url('Nombre de la imagen.gif');
    background-attachment:fixed;
    background-repeat:repeat-x;
}

#masthead {

    Altura mínima:50pixeles;
    Margen izquierdo:110pixeles;
    Margen derecho:180pixeles;
    color:negro;
}

/* Todas las capas quedan invisibles*/

.topsup{

    Tope superior:120pixeles;
    derecha:260pixeles;
    izquierda:290pixeles;
    posición: absoluta;

}

#Bienvenido{

    visibilidad: oculta;

}

#Vision{
    visibilidad:oculta;
}

#Objetivos {
    visibilidad:oculta;
}

#Titulos {
    visibilidad:oculta;
}

#Estudiantes {
    visibilidad:oculta;
}

#Asignaturas {
    visibilidad: oculta ;
}
```



```
#Aulas{
    visibilidad: oculta;
}

#Profesores{ visibilidad:oculta; }

#CrearUsuario{ visibilidad: oculta; }

#Logiarse{ visibilidad:oculta; }

#atop {
    ancho:165píxeles;
    margen-izquierdo:555píxeles;
    margen-superior:-45píxeles;
    tamaño-fuente:12píxeles;
    familia-fuente:Arial, Helvetica, sans-serif;
    posición:absoluta;
}

a.ColBan{
    color: negro;
}

a.ColBan:hover{
    fondo: transparente;
    color: navy;
}

a.ColBan:active{
    fondo: transparente;
    color: gris;
}

.Banner {
    tope:15píxeles;
    margen-izquierdo:29 por ciento;
    ancho:100 por ciento;
    color:navy;
}

.Titulos{
    alineación de texto: centrado;
    Decoración de texto:bajolinea;
    color:#B68150;
}

p .Justi{
    alineación de texto:justificado;
    relleno:5píxeles 5píxeles 5píxeles 5píxeles;
}

#logo {
    izquierda:10píxeles;
    tope:11píxeles;
    posición:absoluta;
    ancho:120píxeles;
    altura:120píxeles;
}
```

```
img {
    altura:100pixeles;
    ancho:110pixeles
}

img.otros {
    altura: 85pixeles;
    ancho:85pixeles;
}

a { margen: 1em 0; }

a.boton {
    decoración de texto: ninguno;
    fondo: #EEE; color: #222;
    borde: 1pixel outset #CCC; relleno: .1em .5em;
}

a.boton:active {
    borde: 1pixel inset #eae1f9;
    color: gray;
}

a.boton:hover {
    fondo: transparente ;
    color: navy;
}

#top_nav {
    margen-izquierdo:26porciento;
    ancho: 100porciento;
    posición: centrado;
    borde de radio: 15pixeles;
}

#container {
    Ancho mínimo: 100porciento;
    Imagen de fondo: url('nombdre de imagen.gif');
    fondo-attachment:fixed;
    fondo-repetir:no-repetir;
}

div .espac { margen: 5pixeles 2pixeles 3pixeles 2pixeles;}

div label{
    ancho:175pixeles;
    valor del float:inzquierda;
    margen-izquierdo: 25pixeles;
}

#page_content { /* Para el cuadro del centro */
    margen-izquierdo: 16porciento;
    margen-derecho: 16porciento;
    borde: 2pixeles lime inset ;
    borde de boton:2pixeles navy solido;
    borde-derecho:2pixeles navy solido;
    borde-izquierdo:2pixeles navy solido;
    borde-tope: 2pixeles navy solido;
    fondo: #E7EBFE;
```

```
        altura:70porciento;
    }

    #right_col {
        float: derecho;
        ancho: 200pixeles;
        posición: fijo;
        tope:115pixeles;
        izquierda:820pixeles;
    }

    #footer {
        limpiar: ambos;
        alineación de texto:centrado;
        familia de fuente:Arial, Helvetica, sans-serif;
        estilo de fuente: italic;
        tamaño de fuente:10pixeles;
        color: navy;
    }

    .Estilo1 {
        Fuente de familia: Tahoma;
        Tamaño de fuente: 12pixeles;
        Ancho de fuente: negrita;
        color: #000000;
    }

    .Estilo2 {
        color: #000000;
        familia de fuente: Tahoma;
        familia de fuente: 12pixeles;
        decoración de texto: ninguno;
    }

    .lateral{           /*Barra lateral izquierda*/
        ancho:14porciento;
        margen:10pixeles;
        posición: absoluto;
        tope:120pixeles;
        float:izquierdo;
    }

    .lateral h3{
        Decoración de texto: parpadeante;
    }

    .lateral th{
        color:navy;
        fondo: #E7EBFE;
        borde: 2pixeles navy solid;
    }

    .login{
        borde: 2pixeles navy solid;
        ancho=100porciento;
    }
```

```
.curso{
    borde: 2píxeles navy solid;
    ancho=100porcentaje;
}

.curso th{
    Decoración de texto:parpadeante;
}

.opciones{ /*Barra lateral derecha*/
    ancho:14porcentaje;
    margen:10píxeles;
    posición: absoluto;
    tope:120píxeles;
    derecha:0;
    izquierda:84.5porcentaje;
}

.opciones h3{
    Decoración de texto: parpadeante;
}

.opciones th{
    color:navy;
    fondo: #E7EBFE;
    borde: 2píxeles navy solid;
}

.mejores{
    borde: 2píxeles navy solid;
    ancho=100porcentaje;
}

.mejores th{
    Decoración de texto:parpadeante;
}

.blog{
    borde: 2píxeles navy solid;
    ancho=100porcentaje;
}

.blog a:link, a:visited {
    Borde de botones: 2píxeles sólido #B68150; /* #fff */
    relleno: 0.80píxeles;
    decoración de texto: parpadeante;
    margen: 1píxel;
}

.blog a:hover{
    a:hover { /* Esta parte es para agregar un color al fondo cuando el mouse este encima */
    color: yellow;
    background-color:#fff;
    }
}
```




Práctica N^a 2

Implementación de JavaScript y DOM.



JAVASCRIPT Y DOM.

Instrucciones de la práctica:

- Haber leído los capítulos 2 y 3 de la monografía.
- Se aconseja consultar al profesor sobre cualquier aspecto que no se comprenda sobre la práctica.

Objetivos:

- Dotar nuestras páginas web de interactividad utilizando funciones y demás elementos de JavaScript.
- Familiarizarse con la sintaxis de JavaScripts y DOM.
- Aplicar para cada uno de los formularios creados en la práctica 1, las validaciones correspondientes creando scripts que funcionen del lado del cliente, antes que los formularios sean enviados al servidor.

La práctica:

Para esta práctica se pretende que usted haga la validación del formulario Registrarse del lado del cliente antes que los valores estos sean enviados al servidor para su posterior procesamiento.

Para la validación del formulario Registrarse, es necesario que se cree un script utilizando JavaScript capaz de avisar al usuario que complete todos los campos si no les ha introducido información, todos los campos son requeridos.

Para el blog, es valido apoyarse en los ejemplos que se proporcionan en el capítulo de DOM, se necesita que cuando el usuario llene el formulario y de clic en el botón Agregar, aparezcan los comentarios que se han agregado, esto debe hacerse utilizando DOM. EL vinculo que desde el que se llama a la página del Blog, se encuentra a la derecha de la página principal en la división Vinculos de Interés.

Para cumplir con esta practica se pide:

- Validar todas las entradas del formulario Ingresar, incluyendo cuando los cajetines no tengan información utilizando un script llamado reg.js, el cual debe ser creado en un archivo independiente y debe llamarse desde la página index.html cuando el usuario haga clic en el botón Aceptar, si no hay información en los cajetines se le debe avisar al usuario que complete todos los campos del formulario.
- Crear la página del blog, por cada comentario que se agregue al blog, debe mostrarse el nombre de la persona que escribió el comentario en negrita, fecha y hora en que lo agrego. Para poder agregar otro comentario, por lo que la funcionalidad del blog debe permitir agregar comentarios mientras el blog esté cargado, los comentarios del blog no se guardaran en ninguna base de datos.
- La estructura de la página blog es el siguiente:
 - Una cabecera: Es un div con una imagen de fondo y con la fase "Blog de comentarios".
 - Una columna central: Es un div en el cual se mostrarán todos los comentarios que se han agregado. El ancho debe ser del 80%.
 - Un botón regresar: Este permitirá volver a la página index.html, se recomienda utilizar un gif.
 - Debe usarse la misma hoja de estilo que se utilizó en el capítulo 1 (Estilos.css).

Puede apoyarse con el ejemplo de Blog, resuelto de este capítulo.

Requerimientos:

- ✓ Un editor de texto plano (Notepad++, Bloc de Notas, etc.).
- ✓ Un Navegar Web. (Explorer, Mozilla, Opera, etc.)
- ✓ Software que nos permita visualizar nuestras páginas desde un servidor, para el caso nuestro hemos escogido XAMPP que incorpora Apache, PhpMyadmin, etc.



Código del script reg.js

```
function validar(){  
    //valido el nombre y la clave del usuario  
  
    if (nombre de usuario == 0 || clave de usuario == 0)  
    {  
        alert("Debe llenar los campos para ingresar")  
        poner el foco en el cajetín del nombre de usuario o login  
        return -----;  
    }  
}
```

Código del blog implementando DOM.

Apoyarse con el ejemplo de blog mostrado en el documento.



Práctica N^a 3

Programación con PHP del lado del servidor.



Programación con PHP.

Instrucciones de la práctica:

- Haber leído el capítulo 4 de la monografía.
- Se aconseja consultar al profesor sobre cualquier aspecto que no se comprenda sobre la práctica.

Objetivos:

- Reforzar los conocimientos y ganar experiencia en el manejo y manipulación de datos con PHP.
- Familiarizarse con la sintaxis de PHP tanto para el tratamiento de datos como con la interacción a bases de datos.
- Vincular las páginas web que se han creado en los capítulos anteriores con páginas programadas desde el lado del servidor con acceso a bases de datos.

La práctica:

Se pretende que el estudiante programe con PHP páginas del lado del servidor que le permitan tomar los datos que se envían desde la página index.html, que permitan además la interacción con bases de datos, manteniendo nuestro código fuera del alcance de los usuarios.

Desde las páginas de PHP en el servidor, se devolverá al usuario la información pertinente como una consulta a la base de datos, presentar un formulario para que sea completado por el usuario, hacer una inserción de datos, etc.

Para cumplir con esta práctica se pide:

- Crear la base de datos necesaria para crear la colección de datos y poder hacer la interacción con PHP, puede sentirse en la libertad de utilizar cualquier software como los enlistados a continuación:

- TurboDbAdmin.
- PhpMyAdmin.
- FileMaker.
- IBM Informix.
- Microsoft SQL server.

Nota: Se deberá investigar por aparte la forma de conectarse a las bases de datos dependiendo del software que elija para trabajar.

- Es necesario crear los siguientes módulos para la funcionalidad del sitio:
 1. Página modulos.php esta es accedida por los usuarios de tipo administrador desde la cual tenemos acceso a:
 - * Visualizar los horarios de clases.
 - * Visualizar los grupos de estudiantes.
 - * Consultas de docentes y alumnos.
 - * Agregar estudiantes. (Llenar datos de los alumnos.)
 - * Agregar docentes. (Llenar datos de los docentes.)
 2. Página grupos.html la cual es accedida por los usuario de tipo Docente y desde la cual podemos:
 - * Visualizar los horarios de clases.
 - * Visualizar los grupos de estudiantes.
 3. Página horarios.html, esta es accedida por los usuarios de tipo Estudiante desde la cual
 - * Solo puede visualizar sus horarios de clases.



Este es el script de la base de datos que utilizaremos:

```
--
-- Base de datos: `academia`
--
DROP DATABASE IF EXISTS `academia`;
CREATE DATABASE `academia`
-----

--
-- Estructura de tabla para la tabla `profesores`
--

DROP TABLE IF EXISTS `profesores`;
CREATE TABLE IF NOT EXISTS `profesores` (
  `nombres` varchar(20) NOT NULL,
  `apellidos` varchar(20) NOT NULL,
  `cedula` varchar(16) NOT NULL,
  `pasaporte` varchar(14) NOT NULL,
  `direccion` varchar(20) NOT NULL,
  `telefono` varchar(9) NOT NULL,
  `fecha_nacimiento` varchar(20) NOT NULL,
  `sexo` varchar(2) NOT NULL,
  `anios_experiencia` varchar(30) NOT NULL,
  PRIMARY KEY (`cedula`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Volcar la base de datos para la tabla `profesores`
--

INSERT INTO `profesores` (`nombres`, `apellidos`, `cedula`, `pasaporte`, `direccion`, `telefono`,
`fecha_nacimiento`, `sexo`, `anios_experiencia`) VALUES
('Angel Reniery', 'Pérez Delgado', '081-060186-0003P', '0810601860003P', 'Chinandega, Restaurante
Anton-Che 1 C. al este', '8472-5889', 'Chinandega, 06 enero 1986', 'M', '5'),
('Daniel Enrique', 'Pérez Sánchez', '081-510593-0001H', '0815105930001H', 'El Viejo, Puente Central 5
1/2 C. al este', '8722-8569', 'Chinandega, 15 mayo 1993', 'M', '3'),
('Erlan Joel', 'Tercero Velasquez', '084-160781-0000K', '0841607810000K', 'Estelí, UNAN 150 metros
oeste', '8773-9744', 'Chinandega, 16 julio 1981', 'M', '9'),
('Lester Antonio', 'Meraz López', '086-310355-0001J', '0863103550001J', 'El Viejo, Colonia Entimo
Andino del CDI 2 C. al norte', '8992-7051', 'El Viejo, 31 marzo 1955', 'M', '12'),
('Denis Fernando', 'Rosales Blanco', '080-130980-0002X', '0801309800002X', 'León, Iglesia El Calvario
1 C. al este, 50 metros al norte', '8457-7642', 'León, 13 septiembre 1980', 'M', '7'),
('Ninfa del Rosario', 'Guardado Vivas', '087-25021952-0001Y', '087250219520001Y', 'El Viejo, esquina
opuesta Iglesia Esquipulas', '8474-2949', 'El Viejo, 25 febrero 1952', 'F', '16'),
('Rolando Antonio', 'Marín Cuadra', '087-15031952-0000W', '087250219520000W', 'León, Contiguo a
Parque San Felipe', '8424-3677', 'León, 25 febrero 1952', 'M', '6'),
('María de Jesús', 'Catín', '081-30031949-0004I', '081300319490004I', 'León, Iglesia San José 30 varas
al norte', '8863-7211', 'Chontales, 30 marzo 1949', 'M', '18'),
('Milton Alexander', 'Voquín Arteaga', '081-16071978-0004S', '081160719780004S', 'León, las pallitas
75 varas al sur', '8604-6711', 'León, 16 julio 1978', 'M', '10'),
('Julia Rosa', 'Santos Picado', '083-11071965-0002F', '083110719650002F', 'Chinandega, puente los
Guarumos 3 c. al sur', '8896-8937', 'León, 11 julio 1965', 'M', '11'),
('Reynaldo Francisco', 'Quintero Manzanares', '081-25121968-0002R', '081251219680002R',
'Chinandega, hospital Materno 2 C. al oeste', '8808-9971', 'El Viejo, 25 diciembre 1968', 'M', '14');
```



```
-- -----  
  
--  
-- Estructura de tabla para la tabla `asignaturas`  
--  
  
DROP TABLE IF EXISTS `asignaturas`;  
CREATE TABLE IF NOT EXISTS `asignaturas` (  
  `codigo` varchar(5) NOT NULL,  
  `nombre` varchar(20) NOT NULL,  
  `descripcion` varchar(50) NOT NULL,  
  `departamento_academico` varchar(20) NOT NULL,  
  PRIMARY KEY (`codigo`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1;  
  
--  
-- Volcar la base de datos para la tabla `asignaturas`  
--  
  
INSERT INTO `asignaturas` (`codigo`, `nombre`, `descripcion`, `departamento_academico`) VALUES  
( '00100', 'Lengua y Literatura', 'Aspectos básicos del idioma español', 'Español'),  
( '00200', 'Matemáticas', 'Aritmética y Álgebra', 'Matemáticas'),  
( '00300', 'Geografía', 'Geografía de Nicaragua', 'Geografía'),  
( '00400', 'Historia', 'Historia de Nicaragua', 'Historia'),  
( '00500', 'Ciencias Sociales', 'Conducción y comportamiento en la sociedad', 'Ciencias Sociales'),  
( '00600', 'Música', 'Lectura de pentagramas y solfeo', 'Música'),  
( '00700', 'Dibujo', 'Técnicas para dibujar, combinación de trazos, delineamiento', 'Dibujo y Artes  
Plásticas'),  
( '00800', 'Artes Plásticas', 'Selección de colores, combinación de tonos, objetos en papel, etc.', 'Dibujo y  
Artes Plásticas');  
  
-- -----  
  
--  
-- Estructura de tabla para la tabla `estudiantes`  
--  
  
DROP TABLE IF EXISTS `estudiantes`;  
CREATE TABLE IF NOT EXISTS `estudiantes` (  
  `carnet` varchar(12) NOT NULL,  
  `nombres` varchar(20) NOT NULL,  
  `apellidos` varchar(20) NOT NULL,  
  `direccion` varchar(50) NOT NULL,  
  `telefono` varchar(9) NOT NULL,  
  `fecha_nacimiento` varchar(20) NOT NULL,  
  `sexo` varchar(2) NOT NULL,  
  `padres` varchar(100) NOT NULL,  
  `anio_ingreso` varchar(4) NOT NULL,  
  PRIMARY KEY (`carnet`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1;  
  
--  
-- Volcar la base de datos para la tabla `estudiantes`  
--
```



```
INSERT INTO `estudiantes` (`carnet`, `nombres`, `apellidos`, `direccion`, `telefono`,
`fecha_nacimiento`, `sexo`, `padres`, `anio_ingreso`) VALUES
('11-2501-2011', 'Lucía Marian', 'Pérez Sáenz', 'Chinadega, Restaurante Anton-Che 130 metros al
oeste', '2341-4271', 'Chiandega, 02 de mayo 2006', 'F', 'Angel Renier Pérez Delgado, Erika del Socorro
Sáñez Ramírez', 'Lunes, 28 enero 2011'),
('11-2502-2011', 'Sandra del Carmen', 'Delgado Rivas', 'Chinadega, de donde fue norori 3 C. al norte',
'2341-4160', 'Chiandega, 29 de octubre 2007', 'F', 'Donald Ramon Delgado, Martha Juia Rivas', 'Lunes,
28 enero 2011'),
('11-2503-2011', 'Erick Andres', 'Flores Alegría', 'Chinadega, parque las rosas 3 C. al este', '2341-3817',
'Chiandega, 10 de noviembre 2006', 'M', 'Heriberto Andrés Flores Sarmiento, Mariana de la Concepción
Alegría Peralta', 'Lunes, 28 enero 2011'),
('11-2505-2011', 'Sara Isabel', 'Caballero Guevara', 'El Viejo, de donde fue norori 3 C. al norte', '2341-
4160', 'Chiandega, 29 de octubre 2007', 'F', 'Victor Antonio Caballero Medina, Teresa de Jesús
Guevara Guevara', 'Martes, 29 enero 2011'),
('11-2506-2011', 'Veronica Gabriela', 'Carrillo Palma', 'El Viejo, puente central 3 C. al norte', '2340-
5676', 'El Viejo, 13 de diciembre 2005', 'F', 'Manuel Antonio Carrillo Castro, Ana Ninosca Palma
Fonseca', 'Martes, 29 enero 2011'),
('11-2507-2011', 'Gloria Maritza', 'Fernández Fernandez', 'El Viejo, Basilica Concepción de María 3 C. al
este', '2343-2208', 'Managua, 19 de junio 2006', 'F', 'Terencio Gregorio Fernández Gómez, Norma
Lucía Fernández Olivares', 'Martes, 29 enero 2011'),
('11-2508-2011', 'Glenda Virginia', 'Trewin Oconor', 'Chinandega, Panadería Santeliz 130 aras al norte',
'2340-5906', 'Chinandega, 13 de enero 2007', 'F', 'Isaura Yolanda Trewin Oconor', 'Martes, 29 enero
2011');
```

-- -----

--

-- Estructura de tabla para la tabla `aulas`

--

```
DROP TABLE IF EXISTS `aulas`;
CREATE TABLE IF NOT EXISTS `aulas` (
  `codigo` varchar(5) NOT NULL,
  `nombre` varchar(20) NOT NULL,
  `ubicacion` varchar(20) NOT NULL,
  `dpto_a_cargo` varchar(50) NOT NULL,
  `tamano` varchar(9) NOT NULL,
  `capacidad_estudiantes` varchar(20) NOT NULL,
  PRIMARY KEY (`codigo`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

--

-- Volcar la base de datos para la tabla `aulas`

--

```
INSERT INTO `aulas` (`codigo`, `nombre`, `ubicacion`, `dpto_a_cargo`, `tamano`,
`capacidad_estudiantes`) VALUES
('1001', 'Laboratorio1', 'Pabellón1', 'Español', '40 metros cuadrados', '50 alumnos'),
('1002', 'Laboratorio2', 'Pabellón1', 'Matemática', '40 metros cuadrados', '50 alumnos'),
('1003', 'Aditorio1', 'Pabellón1', 'Ciencias Sociales', '30 metros cuadrados', '40 alumnos'),
('1004', 'Aditorio2', 'Pabellón1', 'Ciencias Sociales', '30 metros cuadrados', '40 alumnos'),
('1005', 'Leyenda1', 'Pabellón2', 'Música', '120 metros cuadrados', '80 alumnos'),
('1006', 'Leyenda2', 'Pabellón2', 'Dibujo y Artes Plásticas', '120 metros cuadrados', '40 alumnos'),
('1007', 'Biblioteca', 'Pabellón2', 'Matemática', '60 metros cuadrados', '50 alumnos'),
('1008', 'Salón Virtual', 'Pabellón2', 'Geografía', '40 metros cuadrados', '5a alumnos');
```

-- -----


```
DROP TABLE IF EXISTS `usuarios`;
CREATE TABLE IF NOT EXISTS `usuarios` (
  `login` varchar(20) NOT NULL,
  `clave` varchar(10) NOT NULL,
  `tipo` varchar(13) NOT NULL,
  PRIMARY KEY (`login`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```
--
-- Volcar la base de datos para la tabla `usuarios`
--
```

```
INSERT INTO `usuarios` (`login`, `clave`, `tipo`) VALUES
('aldomartinez', 'aldo123', 'Docente'),
('areniery', 'angel123', 'Estudiante'),
('smolina', 'santi123', 'Administrador');
```

- Crear las páginas PHP del sitio Web (no se han creado aún), a continuación se proporcionan las páginas que se deben crear para proporcionar al sitio la funcionalidad necesaria, los valores deben ser enviados al servidor utilizando POST:

Página index.php:

```
<?php
// capturamos las variables del formulario ingresar.html
$login = $_POST['-----']; # Capturando el Nombre de Usuario.
$clave = $_POST['-----']; # Capturando el Contraseña de Usuario.
$tipo = $_POST['-----']; # Capturando el Tipo de Usuario (Administrador, Docente o Estudiante).

//Realizamos la conexion con el servidor localhost
$conexion = conexión con mysql ("servidor", "usuario", "contraseña");

//Seleccinamos la Base de dato trabajo en la conexion
Selección de bd mysql("basededatos", variable que contiene la conexión a la base de datos);

# seleccioname a los Operadores cuyo login, contraseña y tipo sea igual al valor contenidos en la
base de datos
$cadena_u = "seleccionar de la tabla usuarios donde login='-----' y clave='----' y tipo='-----';

//verificamos si la consulta se ejecuto en la conexion con mysql_query
$consulta_u = consulta mysql (-----) de lo contrario ( función error de mysql ());

//le asignamos un valor numerico consultado el numero de filas de la consulta
$nfilas = numero de filas de mysql (-----);

if ( evaluar si $nfilas tiene algún valor ) {
    #Determinar el tipo de usuario
    if (tipo de usuario == 'Administrador'){
        header("página correspondiente");
    }

    if (tipo de usuario == 'Docente'){
        header ("página correspondiente");
    }

    if (tipo de usuario == 'Estudiante'){
```

```

        header ("página correspondiente");
    }

    else {
        cargar("página correspondiente");
    }
?>

```

- Si el tipo de usuario es Administrador y su login y contraseña se encuentran registrados en la base de datos, debe cargarse la página `modulos.html`, la cual tiene las siguientes funciones, se puede:
 - Ver los horarios de clases.
 - Ver los grupos de clases.
 - Buscar estudiantes.
 - Buscar docentes.
 - Agregar estudiantes.
 - Agregar docentes.
 - Agregar usuarios (hay tres tipos, Administrador, Docente, Estudiante).
 - Volver a la página principal.

Esta es la página `modulos.html`

Sistema de Gestion Académicos

Horario de clases Grupos Buscar Estudiantes Buscar Docentes Agregar Estudiantes

Agregar Docentes Agregar Usuario Principal

Horario de Clases

Lunes	Martes	Miercoles	Jueves	Viernes	Sabado	Hora
Aula A de Derecho						7-9
						9-11
	Laboratorio de Alcala					11-1
						1-3
						3-5
						5-7

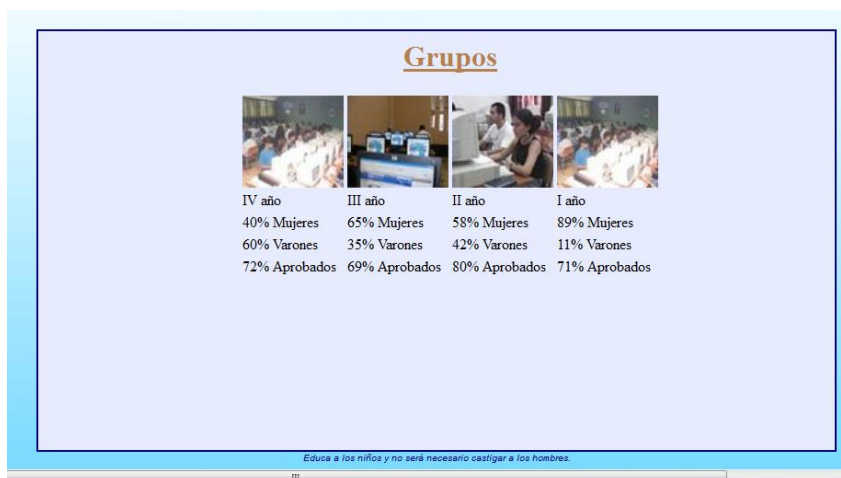
Lunes	Martes	Miercoles	Jueves	Viernes	Sabado	Hora
Aula B de Derecho						7-9
						9-11
	Laboratorio de Alcala					11-1
						1-3
						3-5
						5-7

Educa a los niños y no será necesario castigar a los hombres.

Ilustración 0.5 Página `modulos.html`

Esta página solo puede ser accedida si el usuario es de tipo Administrador, para que esta página funcione correctamente se debe hacer una copia del archivo `index.html` y modificar únicamente el contenido, de los `div`, cada botón habilita un `div` diferente, el `id="Bienvenido"` debe contener el horario de clases a como se muestra en la ilustración de arriba, a continuación se presenta el contenido que debe tener cada `div` subsiguiente:

El div id="Vision", debe contener en general información generalizada de cada uno de los grupos de estudiantes. Este div se hace visible cuando el usuario haga clic en el botón Grupos del menú.



			
IV año	III año	II año	I año
40% Mujeres	65% Mujeres	58% Mujeres	89% Mujeres
60% Varones	35% Varones	42% Varones	11% Varones
72% Aprobados	69% Aprobados	80% Aprobados	71% Aprobados

Educa a los niños y no será necesario castigar a los hombres.

Ilustración 0.6 Div Visión, información general de cada grupo.

El div id="Objetivos", debe contener un formulario con un cajetín y un botón, este formulario hace la llamada a la página buscarestudiante.php, la cual debe hacer la consulta en la base de datos y si encuentra el registro en la base de datos lo mostrará en pantalla.



Buscar Estudiantes

Código

Educa a los niños y no será necesario castigar a los hombres.

Ilustración 0.7 Formulario de búsqueda de estudiante por medio de su Código (Carnet).

El código de la página buscarestudiante.php es el siguiente:

```
<?php
// capturamos las variables del formulario ingresar.html
$carnet = $_POST['-----']; # Código con el que fue registrado en la BD.

//Realizamos la conexión con el servidor localhost
$conexion = conexión con mysql ("servidor", "usuario", "contraseña");

//Seleccinamos la Base de dato trabajo en la conexión
Selección de la bd de mysql ("base de datos", variable de conexión);
```

```
# seleccioname a los Operadores cuyo login, contrasena y tipo sea igual al valor que hay en la base de datos
$cadena_u = "seleccionar de la tabla estudiantes donde carnet='-----';

//verificamos si la consulta se ejecuto en la conexion con mysql_query
$consulta_u = consulta mysql (-----) de lo contrario ( función error de mysql ());

//le asignamos un valor numerico consultado el numero de filas de la consulta
$nfilas = numero de filas mysql (-----);

if (evaluar si $nfilas tiene algún valor ) {
    //presentamos los datos buscados en la base de datos
    //echo "<h3><strong>Estas son todas las coincidencias encontradas : "
    "</strong></h3><br><br>";
    $cadena1 = "seleccionar de la tabla estudiantes donde carnet='-----'";
    $consulta1 = consulta mysql (-----) de lo contrario (error mysql ());
    $filas = numero de filas mysql (variable consulta1);
    echo "<center>";
    echo "<table border=1>";
    echo "<caption>Coincidencias encontradas </caption>";
    for($i=valor de inicio;$i<valor de filas;$i incremento)
    {
        $fila = mysql_fetch_assoc(variable consulta1);
        echo "<tr><td> Nombres </td><td>Apellidos </td><td>Direccion
        </td><td>Telefono</td></tr>";
        echo "<tr><td> ".$fila['-----'].</td><td> ".$fila['-----'].</td><td>
        ".$fila['-----'].</td><td> ".$fila['-----'].</td></tr>";
    }
    echo "</table>";      echo "</center>";
}

else
{
    echo "Lo sentimos, el estudiante no est&aacute; registrado";
}

?>
```

Agregar un vinculo que te devuelva a la página [modulos.html](#).

El div id="Profesores", debe contener un formulario con un cajetín y un botón, este formulario hace la llamada a la página [buscardocente.php](#), la cual debe hacer la consulta en la base de datos y si encuentra el registro en la base de datos lo mostrará en pantalla.



Ilustración 0.8 Formulario de búsqueda de docente por medio de su Código (Cedula)



El código de la página buscadorcentro.php es el siguiente:

```
<?php
// capturamos las variables del formulario ingresar.html
$cedula = $_POST['-----']; # Código con el que fue registrado en la BD.

//Realizamos la conexión con el servidor localhost
$conexion = conexión con mysql ("servidor", "usuario", "contraseña");

//Seleccionamos la Base de datos trabajo en la conexión
Selección de la bd de mysql ("base de datos", variable de conexión);

# seleccioname a los Operadores cuyo login, contraseña y tipo sea igual al valor de las variables registrados en
la base de datos
$cadena_u = "seleccionar de la tabla profesores donde cedula='-----'";

//verificamos si la consulta se ejecuto en la conexión con mysql_query
$consulta_u = consulta mysql (-----) de lo contrario (error mysql ());

//le asignamos un valor numerico consultado el numero de filas de la consulta
$nfilas = numero de filas mysql (-----);

if ( si la variable $nfilas tiene algún valor ) {
    //presentamos los datos buscados en la base de datos
    //echo "<h3><strong>Estas son todas las coincidencias encontradas : "
    "</strong></h3><br><br>";
    $cadena1 = "seleccionar de la tabla profesores donde cedula='-----'";
    $consulta1 = consulta mysql (-----) de lo contrario (erro de mysql ());
    $filas = numero de filas de mysql (variable consulta1);
    echo "<center>";
    echo "<table border=1>";
    echo "<caption>Coincidencias encontradas </caption>";
    for($i=valor inicial;$i<cantidad total de filas;$i incremento)
    {
        $fila = mysql_fetch_assoc(variable consulta1);
        echo "<tr><td> Nombres </td><td>Apellidos </td><td>Direccion
        </td><td>Telefono</td></tr>";
        echo "<tr><td> ".$fila['-----'].</td><td> ".$fila['-----'].</td><td> ".$fila['-
        -----'].</td><td> ".$fila['-----'].</td></tr>";
    }
    echo "</table>";
    echo "</center>";
}

else
{
    echo "Lo sentimos, el estudiante no est&aacute; registrado";
}

?>
```

Agregar un vinculo que te devuelva a la página modulos.html.


```
echo "<tr><td align=center>Todos los datos se enviaron al servidor con  
&eacute;xito</td></tr></table>";  
}  
else{  
    echo "No se pudo realizar la inserci&oacute;n de los datos en la Base de Datos";  
}  
//cerramos la conexi&oacute;n  
mysql_close(-----);  
?>  
Agregar un v&iacute;nulo que te devuelva a la p&eacute;gina modulos.html.
```

El div id="Estudiantes", debe contener un formulario como el que se muestra en la ilustración 0.10, en este formulario se recoge la información pertinente para agregar a un docente a la base de datos, este formulario también hace la llamada a la pégina `agregardocente.php`, la cual debe hacer la inserción de los datos en la base de datos. El campo de la Edad Actual no se guardaré en la base de datos pero debe ser calculada desde una función de JavaScript.



Ilustración 0.10 Formulario para agregar Docentes.

El c&odigo de la pégina `agregarestudiante.php` es el siguiente:

```
<?php  
//capturando los valores de las variables del formulario Agregar Docente  
$nom= $_POST["-----"];  
$ape= $_POST["-----"];  
$cedula= $_POST["-----"];  
$pasaporte= $_POST["-----"];  
$dir= $_POST["-----"];  
$tel= $_POST["-----"];  
$fecha= $_POST["-----"];  
$sex= $_POST["-----"];  
$anioexp= $_POST["-----"];  
  
//Realizamos la conexi&oacute;n con el servidor en este caso es localhost
```

```
$conexion = conexión con mysql ("servidor", "usuario", "contraseña");
//Seleccinamos la Base de datos academia en la conexion
mysql_select_db(-----);
//Realizamo la consulta para insertar datos en la tabla estudiantes
$cadena = "insertar en la tabla profesores los valores
('-----','-----','-----','-----','-----','-----','-----','-----','-----','-----');";
//verificamos si la consulta se ejecuto en la conexion con mysql_query asignado un valor a la variable num
$num = consulta mysql (-----) de lo contrario (error de mysql ());
// si num es mayor que 0 , la consulta se ejecuto bien y se envia un mensaje enterando la inserción
if($num es mayor que cero) {
    echo "<table border=1 width=300>";
    echo "<tr><th bgcolor=#66CC66>Insertando datos de Docentes</th></tr>";
    echo "<tr><td align=center>Todos los datos se enviaron al servidor con
    &eacute;xito</td></tr></table>";
}
else{
    echo "No se pudo realizar la inserci&ocute;n de los datos en la Base de Datos";
}
//cerramos la conexion
mysql_close(-----);
?>
Agregar un vinculo que te devuelva a la página modulos.html.
```

El div id="Asignaturas", debe contener un formulario como el que se muestra en la ilustración 0.11, en este formulario se recoge la información pertinente para agregar a un usuario del sistema a la base de datos, este formulario también hace la llamada a la página `agegarusuario.php`, la cual debe hacer la inserción de los datos en la base de datos.

Ilustración 0.11 Formulario de inserción de datos de usuarios del sistema.



El código de la página agregarusuario.php es el siguiente:

```
<?php
//capturando los valores de las variables del formulario Agregar Usuario
$login= $_POST["-----"];
$clave= $_POST["-----"];
$tipo= $_POST["-----"];

//Realizamos la conexion con el servidor en este caso es localhost
$conexion = conexión con mysql ("servidor", "usuario", "contraseña");
//Seleccionamos la Base de datos academia en la conexion
mysql_select_db(-----);
//Realizamos la consulta para insertar datos en la tabla estudiantes
$cadena = "insertar en la tabla usuarios los valores ('-----','-----','-----)";
//verificamos si la consulta se ejecuto en la conexion con mysql_query asignado un valor a la variable num
$num = consulta mysql (-----) de lo contrario (error de mysql ());
// si num es mayor que 0 , la consulta se ejecuto bien y se envia un mensaje enterando la inserción
if( $num es mayor que cero) {
    echo "<table border=1 width=300>";
    echo "<tr><th bgcolor=#66CC66>Insertando datos de Usuarios</th></tr>";
    echo "<tr><td align=center>Todos los datos se enviaron al servidor con  
&eacute;xito</td></tr></table>";
}
else{
    echo "No se pudo realizar la inserci&oacute;n de los datos en la Base de Datos";
}
//cerramos la conexion
mysql_close(-----);
?>
```

Agregar un vinculo que te devuelva a la página modulos.html.

El último botón del menú, es un enlace para volver a la página index.html

- Si el tipo de usuario es Docente y su login y contraseña se encuentran registrados en la base de datos, debe cargarse la página grupos.html, la cual únicamente tiene tres botones en su menú, uno para visualizar los grupos, los horarios y el otro es para volver a la página principal:

Sistema de Gestion Académicos

[Horarios](#) [Grupos](#) [Principal](#)

Horario de Clases

Lunes	Martes	Miercoles	Jueves	Viernes	Sabado	Hora
Aula A de Derecho						7-9
						9-11
	Laboratorio de Alcala					11-1
						1-3
						3-5
						5-7

Lunes	Martes	Miercoles	Jueves	Viernes	Sabado	Hora
Aula B de Derecho						7-9
						9-11
	Laboratorio de Alcala					11-1
						1-3
						3-5
						5-7

Educa a los niños y no será necesario castigar a los hombres.

Ilustración 0.12 Página grupos.html

- Si el tipo de usuario es Estudiante y su login y contraseña se encuentran registrados en la base de datos, debe cargarse la página horarios.html, la cual únicamente tiene un botón en su menú, para volver a la página principal:

Sistema de Gestion Académicos

Principal

Horario de Clases

Lunes	Martes	Miercoles	Jueves	Viernes	Sabado	Hora
Aula A de Derecho						7-9
						9-11
	Laboratorio de Alcala					11-1
						1-3
						3-5
						5-7

Lunes	Martes	Miercoles	Jueves	Viernes	Sabado	Hora
Aula B de Derecho						7-9
						9-11
	Laboratorio de Alcala					11-1
						1-3
						3-5
						5-7

Educa a los niños y no será necesario castigar a los hombres.

Ilustración 0.13 Página horarios.html



Práctica N^a 4

Implementando Ajax en nuestra aplicación.

Implementación de Ajax.

Instrucciones de la práctica:

- Haber leído el capítulo 5 y 6 de la monografía.
- Se aconseja consultar al profesor sobre cualquier aspecto que no se comprenda sobre la práctica.

Objetivos:

- Adquirir experiencia en la validación de formularios con Ajax.
- Familiarizarse con la sintaxis y peticiones Ajax asíncronas del cliente al servidor.
- Dotar nuestras páginas de código PHP, con las técnicas necesarias para incorporar el funcionamiento del sitio web con Ajax.
- Vincular las páginas web programadas del lado del cliente con las programadas del lado del servidor.

La práctica:

Se pide que el estudiante se realice una búsqueda por medio del campo “apellidos”, de la base de datos para que al momento de realizar la búsqueda de un estudiante se haga utilizando PHP con Ajax del lado del servidor, que mantenga la interacción con la base de datos.

Desde PHP en el servidor, se devolverán al usuario todas las coincidencias a medida que el usuario va tecleando caracteres en el formulario de búsqueda. Se debe modificar la página `modulos.html` y agregarle el código php que sea necesario, antes de agregar el código necesario a esta página, se debe renombrar a `modulos.php`.

Para cumplir con esta práctica se pide crear o modificar las siguientes páginas:

- **modulos.php:**
- **ajax.php:** es la página que debe ser creada en el servidor para contestar las peticiones del formulario de búsqueda.
- **Reg.js:** Este script es el mismo que creo en la práctica 2 donde se utilizó para validar el cajetín de ingreso mostrado en la página principal. Dentro de este se debe agregar el código necesario para que las contestaciones del servidor sean mostradas en la página.
- **jquery-1.2.1.pack.js:** El código de este script debe ser copiado sin modificación. Este archivo debe ser incluido en la página **modulos.php**
- **auto.css:** esta es una hoja de estilo utilizado para mostrar el cajetín en el cual se muestran los apellidos de los estudiantes. La cual se debe incorporar en la página **modulos.php**.

Requerimientos:

- ✓ Un editor de texto plano (Notepad++, Bloc de Notas, etc.).
- ✓ Un Navegador Web. (Explorer, Mozilla, Opera, etc.)
- ✓ Software que nos permita visualizar nuestras páginas desde un servidor, para el caso nuestro hemos escogido XAMPP que incorpora Apache, PhpMyadmin, etc.

Código de la página modulos.php:

```
<?php
    Referenciar el archive index_top.php
?>
<html>

<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>Sistema de Gestion de Academicos</title>

<link href="Estilo.css" rel="stylesheet" type="text/css">

<script language="javascript" type="text/javascript">
    function OcultarDivs() {
        document.obtener element por Id("-----").style.visibility = 'oculto';
        document.obtener element por Id ("-----").style.visibility = ' oculto ';
        document.obtener element por Id ("-----").style.visibility = ' oculto ';
        document.obtener element por Id ("-----").style.visibility = ' oculto ';
        document.obtener element por Id ("-----").style.visibility = ' oculto ';
        document.obtener element por Id ("-----").style.visibility = ' oculto ';
        document.obtener element por Id ("-----").style.visibility = ' oculto ';
    }

    function cambiar visibilidad(argumento) {
        document.obtener element por Id (argumento).style.visibility = 'visible';
        return true;
    }
</script>

</head>

<body onload="cambiarvisibilidad('-----');">

<div id="-----">
    <div id="logo">
        </div>

    <h1 class="Banner">Sistema de Gestion Acad&eacute;micos</h1>
    <div id="atop" ><a class="ColBan" href="#" onClick=" hideAllDivs();
changeVisibility('CrearUsuario')"></a>
    <a class="ColBan" href="#" onclick=" OcultarDivs(); cambiarvisibilidad('-----') " ></a></div></div>

<div id="top_nav">
    <a class="boton" href="#" onClick=" OcultarDivs(); cambiarvisibilidad('-----') " >Horario de
    clases</a>
    <a class="boton" href="#" onClick=" OcultarDivs(); cambiarvisibilidad('-----') " >Grupos</a>
    <a class="boton" href="#" onClick=" OcultarDivs(); cambiarvisibilidad('-----') " >Buscar
    Estudiantes</a>
    <a class="boton" href="#" onClick=" OcultarDivs(); cambiarvisibilidad('-----') " >Buscar
    Docentes</a>
    <a class="boton" href="#" onClick=" OcultarDivs(); cambiarvisibilidad('-----') " >Agregar
    Estudiantes</a><br /><br />
    <a class="boton" href="#" onClick=" OcultarDivs(); cambiarvisibilidad('-----') " >Agregar
    Docentes</a>
```



```
<a class="boton" href="#" onClick=" OcultarDivs(); cambiarvisibilidad('-----') " >Agregar
Usuario</a>
<a class="boton" href="index.html" >Principal</a>
</div>
<br> <br>
```

```
<div id="container">
  <div id="page_content">
    <div id="-----" class="topsup"><br /><br />
      <h1 class="Titulos">Horario de Clases</h1>
      <center><table border=2>
        <tr>
          <th>Lunes</th><th>Martes</th><th>Miercoles</th><th>Jueves</th><th>Viernes</th><th>Sabado</th><th>Hor
          a</th> </tr>
          <tr> <td>Aula A de
          Derecho</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>7-
          9</td></tr>
          <tr>
            <td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>9-
            11</td></tr>
            <tr> <td>&nbsp;</td><td>Laboratorio de Alcala
            </td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>11-1</td></tr>
            <tr>
              <td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>1-
              3</td></tr>
              <tr>
                <td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>3-
                5</td></tr>
                <tr>
                  <td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td></td><td>5-7</td></tr>
                  </tr>
                </table>

                <table border=2>
                  <tr>
                    <th>Lunes</th><th>Martes</th><th>Miercoles</th><th>Jueves</th><th>Viernes</th><th>Sabado</th><th>Hor
                    a</th> </tr>
                    <tr> <td>Aula B de
                    Derecho</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>7-
                    9</td></tr>
                    <tr>
                      <td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>9-
                      11</td></tr>
                      <tr> <td>&nbsp;</td><td>Laboratorio de Alcala
                      </td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>11-1</td></tr>
                      <tr>
                        <td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>1-
                        3</td></tr>
                        <tr>
                          <td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>3-
                          5</td></tr>
                          <tr>
                            <td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td><td></td><td>5-7</td></tr>
                            </tr>
                          </table></center>
```

```
<br >

</div><br>
<div id="-----" class="topsup" ><br /><br />
    <h1 class="Titulos">Grupos</h1>
        <center><table border="0" width="30%">
            <tr>
                <td></td>
                <td></td>
                <td></td>
                <td></td>
            </tr><tr>
                <td> IV a&ntilde;o </td>
                <td> III a&ntilde;o </td>
                <td> II a&ntilde;o </td>
                <td> I a&ntilde;o </td>
            </tr><tr>
                <td> 40% Mujeres </td>
                <td> 65% Mujeres </td>
                <td> 58% Mujeres </td>
                <td> 89% Mujeres </td>
            </tr><tr>
                <td> 60% Varones </td>
                <td> 35% Varones </td>
                <td> 42% Varones </td>
                <td> 11% Varones </td>
            </tr><tr>
                <td> 72% Aprobados </td>
                <td> 69% Aprobados </td>
                <td> 80% Aprobados </td>
                <td> 71% Aprobados </td>
            </tr>
        </table></center>

</div>

<div id="-----" class="topsup"><br /><br />
    <h1 class="Titulos">Buscar Estudiantes</h1>
        <form name="estudiantes" method="post" action="buscarestudiente.php">

            <div class="espac"><label for="apellidos">Teclee el apellido *:</label>
                <input type="text" name="apellido" id="inputString" size="30"
                onkeyup="lookup(this.value);" onblur="fill();" />
                <input type="submit" value="Buscar">
            </div>

            <div class="suggestionsBox" id="suggestions" style="display: none;">
                <div class="suggestionList" id="autoSuggestionsList">

            </div>
        </div>

    </form>

</div>

<div id="-----" class="topsup"><br /><br />
    <h1 class="Titulos">Buscar Docentes</h1>
        <form name="docentes" method="post" action="buscardocente.php">
```



Edad Actual:


```
<input type="text" name="Sexo" size="1" />

</div>

<!-- Padres -->
<div class="espac">
    <label for="padres">Padres: *</label>
    <input type="text" name="padres" size="30" maxlength="32"/>
</div>

<!-- Carnet -->
<div class="espac">
    <label for="txtCarnet">Carnet: *</label>
    <input type="text" name="carnet" size="8" maxlength="12"/>
</div>

<!-- Anio de ingreso -->
<div class="espac">
    <label for="txtAnyoin">A&ntilde;o de Ingreso: *</label>
    <input type="text" name="anyoin" size="4" maxlength="4"/>
</div><br />

<center><input type="submit" value="Aceptar" ></center>

</form>

</div>

<div id="-----" class="topsup"><br /><br />
    <h1 class="Titulos">Agregar Docentes</h1>
    <form name="profesores" method="post" action="agregardocentes.php">
        <div class="espac" >
            <label for="Nombre">Nombres *</label>
            <input type="text" name="nombres" size="15" maxlength="20" >
        </div>

        <div class="espac" >
            <label for="Apellido">Apellidos *</label>
            <input type="text" nombre="apellidos" size="15" maxlength="20" >
        </div>

        <div class="espac">
            <label for="cod">Cedula *</label>
            <input type="text" name="cedula" size="16" >
        </div>

        <div class="espac">
            <label for="Pas">Pasaporte:*</label>
            <input type="text" name="pas" size="14">
        </div>

        <div class="espac">
            <label for="Dir" >Direcci&ocute;n *</label>
            <input type="text" name="dir" size="20">
        </div>
    </form>
</div>
```

[illegible]

```
<form name="profesores" method="post" action="agregarusuario.php">
    <div class="espac" >
        <label for="Nombre">Login*</label>
        <input type="text" name="login" size="15" maxlength="20" >
    </div>

    <div class="espac" >
        <label for="Apellido">Clave*</label>
        <input type="password" name="clave" size="10">
    </div>

    <div class="espac">
        <label for="PostG">Tipo de Usuario*</label>
        <select name="tipo" >
            <option selected/>Administrador
            <option />Docente
            <option />Estudiante
        </select>
    </div>
    <br /><br/>
    <center><input type="submit" value="Agregar"></center>
</form>
</div>
</div>
</div>
</div>
<div id="-----" > Educa a los niños y no ser necesario castigar a los hombres. </div>
</div>
</body>
</html>
```

Modificar Código del script reg.js:

```
function lookup(inputString) {
    if(inputString.length == 0) {
        $('#suggestions').hide();
    } else {
        $.post("rpc.php", {queryString: ""+inputString+""}, function(data){
            if(data.length >0) {
                $('#suggestions').show();
                $('#autoSuggestionsList').html(data);
            }
        });
    }
}

function fill(thisValue) {
    $('#inputString').val(thisValue);
    setTimeout($('#suggestions').hide();, 200);
}
```



Código de la página **ajax.php**:

```
<?php
// mysqli('localhost', 'Usuario', 'Password', 'Base de Dato');
$db = new mysqli('_____', '_____', '_____', '_____');

if(!$db) {
    // Mostrar un error si no nos conectamos.
    echo 'ERROR: No se pudo conectar a la base de datos.';
} else {
    // Si hay una petición o cadena de consulta?
    if(isset($_POST['queryString'])) {
        $queryString = $db->real_escape_string($_POST['queryString']);
        // Si la longitud de la consulta es mayor que 0?
        if(strlen(_____) > 0) {
            // Ejecutar la consulta: Nosotros estamos usando '$queryString%'
            // Usted aquí solo necesita alterar la consulta con los datos de su Base de Datos.
            // ejemplo: SELECT NombreColumna FROM nombreTabla WHERE NombreColumna
            // sea '$queryString%' hasta un máximo de 10
            $query = $db->query("SELECT _____ FROM estudiantes WHERE _____
            LIKE '$queryString%' LIMIT 10");
            if(_____) {
                // Si la consulta se ejecuta, ejecutamos nuestro ciclo while.
                while ($result = $query -> fetch_object()) {
                    // Para el formato de los resultados, usamos <li> para las listas, puedes
                    // cambiarlo si lo deseas.
                    // La función onClick mostrará las filas de texto con los resultados.
                    // Usted debe cambiar: $result->apellidos por $result->NombreColumna

                    echo '<li onClick="fill(\''.$result->_____.'\'),">'.$result->_____. '</li>';
                }
            } else {
                echo 'ERROR: Ocurrió un problema con la consulta.';
            }
        } else {
            // No hacer ninguna otra cosa.
        }
    } else {
        echo 'No debe haber acceso directo a este script!';
    }
}

?>
```

Codigo del script **jquery-1.2.1.pack.js:**

```
eval(function(p,a,c,k,e,r){e=function(c){return(c<a?"":e(parseInt(c/a)))+(c=c%a)>35?String.fromCharCode(c+29):c.toString(36)};if(!".replace(/\\/,String)){while(c-->r[e(c)]=k[c]]e(c);k=[function(e){return r[e]};e=function(){return'\\w+'};c=1;while(c-->r[e(c)]=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return p}('G(){9(1m E!="W")H w=E;H E=18.15=G(a,b){I 6 7u E?6.5N(a,b):1u E(a,b);9(1m $!="W")H D=$;18.$=E;H u=/^[^<]*(.\\s+)>[^>]*$/#(\\w+)$;/E.1b=E.3A={5N:G(c,a){c=c||U;9(1m c=="1M")}{H m=u.2S(c);9(m&&(m[1]||a)){9(m[1])c=E.4D([m[1]],a);J{H b=U.3S(m[3]);9(b)9(b.22!=m[3])I E().1Y(c);J{6[0]=b;6.K=1;I 6J c=[]}J I 1u E(a).1Y(c)J 9(E.1n(c))I 1u E(U)[E.1b.2d?"2d":"39"](c);I 6.6v(c.1c==1B&&c||(c.4c|c.K&&c!=18&&c.1y&&c[0]!=W&&c[0].1y)&&E.2h(c)||[c]),4c:"1.2.1",7Y:G(){I 6.K,K:0,21:G(a){I a==W?E.2h(6):6[a],2o:G(a){H b=E(a);b.4Y=6;I b},6v:G(a){6.K=0;1B.3A.1a.16(6,a);I 6},N:G(a,b){I E.N(6,a,b),4l:G(a){H b=-1;6.N(G(i){9(6==a)b=i});I b},1x:G(f,d,e){H c=f;9(f.1c==3X)9(d==W)I 6.K&&E[e]"1x"](6[0],f)||W;J{c={};c[f]=d}I 6.N(G(a){L(H b 1i c)E.1x(e?6.R:6,b,E.1e(6,c[b],e,a,b))}},17:G(b,a){I 6.1x(b,a,"3C")},2g:G(e){9(1m e!="5i"&&e!=S)I 6.4n().3g(U.6F(e));H t="";E.N(e)[6,G(){E.N(6.3j,G(){9(6.1y!=8)t+=6.1y!=1?6.6x:E.1b.2g([6]))});I t},5m:G(b){9(6[0])E(b,6[0].3H).6u().3d(6[0]).1X(G){H a=6;1W(a.1w)a=a.1w;I a},3g(6);I 6},8m:G(a){I 6.N(G){E(6).6q().5m(a))},8d:G(a){I 6.N(G){E(6).5m(a))},3g:G(){I 6.3z(1q,Q,1,G(a){6.58(a)}),6j:G(){I 6.3z(1q,Q,-1,G(a){6.3d(a,6.1w)}),6g:G(){I 6.3z(1q,P,1,G(a){6.12.3d(a,6)}),50:G(){I 6.3z(1q,P,-1,G(a){6.12.3d(a,6.2q)}),2D:G(){I 6.4Y||E([ ]),1Y:G(t){H b=E.1X(6,G(a){I E.1Y(t,a)};I 6.2o(/^[^>+][^>+]/.14(t)||t.1g("...")>-1?E.4V(b):b),6u:G(e){H f=6.1X(G){I 6.67?E(6.67)[0]:6.4R(Q)};H d=f.1Y("****").4O().N(G){9(6[F]=W)6[F]=S};9(e===Q)6.1Y("****").4O().N(G(i){H c=E.M(6,"2P");L(H a 1i c)L(H b 1i c[a])E.1j.1f(d[i],a,c[a][b],c[a][b].M));I f},1E:G(t){I 6.2o(E.1n(t)&&E.2W(6,G(b,a){I t.16(b,[a]))||E.3m(t,6)}),5V:G(t){I 6.2o(t.1c==3X&&E.3m(t,6,Q)||E.2W(6,G(a){I(t.1c==1B||t.4c)?E.2A(a,t)<0:a!=t})),1f:G(t){I 6.2o(E.1R(6.21),t.1c==3X?E(t).21():t.K!=W&&(t.11||E.11(t,"2Y"))?t:[t])),3t:G(a){I a?E.3m(a,6).K>0:P},7c:G(a){I 6.3t("."+a)},3i:G(b){9(b==W)9(6.K){H c=6[0];9(E.11(c,"24")){H e=c.4Z,a=[],Y=c.Y,2G=c.O=="24-2G";9(e<0)I S;L(H i=2G?e:0,33=2G?e+1:Y.K;i<33;i++){H d=Y[i];9(d.26){H b=E.V.1h&&!d.9V["1Q"].9L?d.2g:d.1Q;9(2G)I b;a.1a(b)}I a}J I 6[0].1Q.1p(/\\r/g,""))}J I 6.N(G){9(b.1c==1B&&4k[5j/.14(6.O))6.2Q=(E.2A(6.1Q,b)>=0||E.2A(6.2H,b)>=0);J 9(E.11(6,"24")){H a=b.1c==1B?b[b];E("9h",6).N(G){6.26=(E.2A(6.1Q,a)>=0||E.2A(6.2g,a)>=0)};9(!a.K)6.4Z=-1}J 6.1Q=b});4o:G(a){I a==W?(6.K?6[0].3O:S):6.4n().3g(a),6H:G(a){I 6.50(a).28()},6E:G(i){I 6.2J(i,i+1)},2J:G(){I 6.2o(1B.3A.2J.16(6,1q)),1X:G(b){I 6.2o(E.1X(6,G(a,i){I b.2O(a,i,a)})),4O:G(){I 6.1f(6.4Y)},3z:G(f,d,g,e){H c=6.K>1,a;I 6.N(G){9(!a){a=E.4D(f,6.3H);9(g<0)a.8U()}H b=6;9(d&&E.11(6,"1l")&&E.11(a[0],"4m"))b=6.4l("1K")[0]||6.58(U.5B("1K"));E.N(a,G){H a=c?6.4R(Q):6;9(!5A(0,a))e.2O(b,a))}};G 5A(i,b){H a=E.11(b,"1J");9(a){9(b.3k)E.3G({1d:b.3k,3e:P,1V:"1J"});J E.5f(b,2g)||b.6s||b.3O[""]};9(b.12)b.12.3b(b)}J 9(b.1y==1)E("1J",b).N(5A);I a}E.1k=E.1b.1k=G(){H c=1q[0]||{;a=1,2c=1q.K,5e=P;9(c.1c==8o){5e=c;c=1q[1]||{;9(2c==1){c=6;a=0}H b;L;(a<2c;a++)9((b=1q[a])!=S)L(H i 1i b){9(c==b[i])6r;9(5e&&1m b[i])=\\5i\\&&c[i])E.1k(c[i],b[i]);J 9(b[i]!=W)c[i]=b[i]}I c};H F="15"+(1u 3D()).3B(),6p=0,5c={;E.1k({8a:G(a){18.$=D;9(a)18.15=w;I E},1n:G(a){!!a&&1m a!="1M"&&a.11&&a.1c!=1B&&/G/i.14(a+"")},4a:G(a){I a.2V&&!a.1G||a.37&&a.3H&&a.3H.1G},5f:G(a){a=E.36(a);9(a){9(18.6l)18.6l(a);J 9(E.V.1N)18.56(a,0);J 3w.2O(18,a)},11:G(b,a){I b.11&&b.11.27()==a.27(),1L:{},M:G(c,d,b){c=c==18?5c:c;H a=c[F];9(!a)a=c[F]=++6p;9(d&&!E.1L[a])E.1L[a]={;9(b!=W)E.1L[a][d]=b;I d?E.1L[a][d]:a},30:G(c,b){c=c==18?5c:c;H a=c[F];9(b){9(E.1L[a]){2E E.1L[a][b];b="";L(b 1i E.1L[a])1T;9(!b)E.30(c)}J{2a[2E c[F]]29(e){9(c.53)c.53(F)}2E E.1L[a]},N:G(a,b,c){9(c){9(a.K==W)L(H i 1i a)b.16(a[i],c);J L(H i=0,48=a.K;i<48;i++)9(b.16(a[i],c)==P)1T}J{9(a.K==W)L(H i 1i a)b.2O(a[i],i,a[i]);J L(H i=0,48=a.K,3i=a[0];i<48&&b.2O(3i,i,3i)!=P;3i=a[++i]}I a},1e:G(c,b,d,e,a){9(E.1n(b))b=b.2O(c,[e]);H f=/z-?4l|7T-?7Q|1r|69|7P-?1H/i;I b&&b.1c==4W&&d=="3C"&&f.14(a)?b+"2T":b},1o:{1f:G(b,c){E.N((c||"").2l(/\\s+/),G(i,a){9(!E.1o.3K(b.1o,a))b.1o+= (b.1o?" ":"")a}},28:G(b,c){b.1o=c!=W?E.2W(b.1o.2l(/\\s+/),G(a){I!E.1o.3K(c,a)}).66(" ":""),3K:G(t,c){I E.2A(c,(t.1o||t).3s().2l(/\\s+/))>-1},2k:G(e,o,f){L(H i 1i o){e.R["3r"+i]=e.R[i];e.R[i]=o[i]}f.16(e,[]);L(H i 1i o)e.R[i]=e.R["3r"+i]},17:G(e,p){9(p=="1H")p=="2N"}H b={;42,41,d=["7J","7I","7G","7F"];E.N(d,G(){b["7C"+6]=0;b["7B"+6+"5Z"]=0});E.2k(e,b,G){9(E(e).3t('\\:3R\\')){42=e.7A;41=e.7w}J{e=E(e.4R(Q)).1Y("4k").5W("2Q").2D().17({4C:"1P",2X:"4F",19:"2Z",7o:"0",1S:"0"}).5R(e.12)[0];H a=E.17(e.12,"2X")||"3V";9(a=="3V")e.12.R.2X="7g";42=e.7e;41=e.7b;9(a=="3V")e.12.R.2X="3V";e.12.3b(e)});I p=="1H"?42:41}I E.3C(e,p)},3C:G(h,j,i){H g,2w=[],2k=[];G 3n(a){9(!E.V.1N)I P;H
```



293

m=g.2S(t);9(m){H p=m[1].27();L(H i=0;d[i];i++)L(H
c=d[i].1w;c;c=2q)9(c.1y==1&&(p=="*")|c.11.27()==p.27()))r.1a(c);d=r;t=t.1p(g,"");9(t.1g(""
")=0)6r;I=Q}J{g=/(^(>+~])\s*(\w*)/i;9((m=g.2S(t))!=S){r=;H p=m[2],1R={;m=m[1];L(H j=0,31=d.K;j<31;j++){H
n=m=="~"||m=="~"?d[j].2q;d[j].1w;L(;n;n=n.2q)9(n.1y==1){H
h=E.M(n);9(m=="~"&&1R[h])1T;9(!p||n.11.27()==p.27()){9(m=="~")1R[h]=Q;r.1a(n)}9(m=="~")1T}}d=r;t=E.36(t.1
p(g,"");I=Q}}9(t&&!){9(!t.1g("","")){9(o==d[0])d.44();2f=E.1R(2f,d);r=d=[o];t=" "+t.68(1,t.K)}J{H k=6i;H
m=k.2S(t);9(m){m=[0,m[2],m[3],m[1]]}J{k=6h;m=k.2S(t)}m[2]=m[2].1p(/\\w/g,"");H f=d[d.K-
1];9(m[1]=="#"&&f&&f.3S&&!E.4a(f)){H q=f.3S(m[2]);9((E.V.1h||E.V.34)&&q&&1m
q.22=="1M"&&q.22!=m[2])q=E('[@22="\"'+m[2]+'\"'],f)[0];d=r=q&&(!m[3]||E.11(q,m[3]))?{q:[]}{J{L(H i=0;d[i];i++){H
a=m[1]=="#"&&m[3]?m[3]:m[1]!="~"||m[0]=="~"?m[2];9(a=="*"&&d[i].11.2p()=="5i")a="3a";r=E.1R(r,d[i].4l(a))}9(
m[1]=="~")r=E.4X(r,m[2]);9(m[1]=="~")}{H e=;L(H
i=0;r[i];i++)9(r[i].4p("22")==m[2]){e=r[i];1T;r=e}d=r;t=t.1p(k,"")}9(t){H
b=E.1E(t,r);d=r=b;r;t=E.36(b,t)}9(t)d=;9(d&&o==d[0])d.44();2f=E.1R(2f,d);I 2f;4X:G(r,m,a){m=" "+m+" ";H
c=;L(H i=0;r[i];i++){H b=" "+r[i].1o+" ").1g(m)>=0;9(!a&&b|a&&!b)c.1a(r[i])I c};1E:G(t,r,h){H
d;1W(t&&t!=d){d=t;H p=E.6a,m;L(H
i=0;p[i];i++){m=p[i].2S(t);9(m){t=t.7O(m[0].K);m[2]=m[2].1p(/\\w/g,"");1T}}9(!m)1T;9(m[1]=="~"&&m[2]=="5V")r=E.1
E(m[3],r,Q).r;J 9(m[1]=="~")r=E.4X(r,m[2],h);J 9(m[1]=="~")}{H g=,O=m[3];L(H i=0,31=r.K;i<31;i++){H
a=r[i],z=a[E.5o[m[2]]|m[2]];9(z==S||/6C|3k|26/.14(m[2]))z=E.1x(a,m[2])||\\';9((O=="&&!z|O=="&&z==m[5])|O
=="!&&z!=m[5]|O=="^&&z&&!z.1g(m[5])|O=="\$&&z.68(z.K-
m[5].K)==m[5])|(O=="~&&z.1g(m[5])>=0)^h)g.1a(a)}r=g}J 9(m[1]=="~"&&m[2]=="2l-46")}{H
e={,g=,14=/(\\d*)n\\+?(\\d*)/.2S(m[3]=="6f"&&"2n"||m[3]=="6e"&&"2n+1"||/\\D/.14(m[3])&&"n"+m[3]|m[3]),3v=
(14[1]|1)-0,d=14[2]-0;L(H i=0,31=r.K;i<31;i++){H j=r[i],12=j.12,22=E.M(12);9(!e[22]){H c=1;L(H
n=12.1w;n;n=n.2q)9(n.1y==1)n.4U=c++;e[22]=Q}H b=P;9(3v==1){9(d==0||j.4U==d)b=Q}J
9((j.4U+d)%3v==0)b=Q;9(b^h)g.1a(j)}r=g}J{H f=E.55[m[1]];9(1m f!="1M")f=E.55[m[1]]|m[2];f=3w("P||G(a,i){I
"+f+"");r=E.2W(r,f,h)}{r:r,t:t};4e:G(b,c){H d=;H a=b[c];1W(a&&a!=U){9(a.1y==1)d.1a(a);a=a[c]}I
d},2l:G(a,e,c,b){e=e||1;H d=0;L(;a;a=a[c])9(a.1y==1&&a++d==e)1T;I a},5d:G(n,a){H
r=;L(;n;n=n.2q){9(n.1y==1&&!a||n!=a)}r.1a(n)}I
r});E.1j={1f:G(g,e,c,h){9(E.V.1h&&g.4j!=W)g=18;9(!c.2u)c.2u=6.2u++;9(h!=W){H d=c;c=G(){I
d.16(6,1q)};c.M=h;c.2u=d.2u}H i=e.2l(" ");e=i[0];c.O=i[1];H b=E.M(g,"2P")||E.M(g,"2P",{});H f=E.M(g,"2t",G){H
a;9(1m E=="W"||E.1j.4T)I a;a=E.1j.2t.16(g,1q);I a};H j=b[e];9(!j){j=b[e]=;9(g.4S)g.4S(e,f,P);J
g.7N("43"+e,f)}c.2u=c;6.1Z[e]=Q;2u:1,1Z:;28:G(d,c,b){H e=E.M(d,"2P"),2L,4l;9(1m c=="1M")}{H
a=c.2l(" ");c=a[0]}9(e){9(c&&c.O){b=c.4Q;c=c.O}9(!c){L(c 1i e)6.28(d,c)}J 9(e[c]){9(b)2E e[c]|b.2u};J L(b 1i
e[c])9(!a[1])|e[c]|b.O==a[1])2E e[c]|b;L(2L 1i e[c])1T;9(!2L){9(d.4P)d.4P(c,E.M(d,"2t"),P);J
d.7M("43"+c,E.M(d,"2t"));2L=S;2E e[c]}L(2L 1i
e)1T;9(!2L){E.30(d,"2P");E.30(d,"2t")}};1F:G(d,b,e,c,f){b=E.2h(b||);9(!e){9(6.1Z[d])E(" ").1f([18,U]).1f(d,b)}J{H
a,2L,1b=E.1n(e[d])|S,4N=!b[0]||b[0].2M;9(4N)b.4w(6.4M({O:d,2m:e}));b[0].O=d;9(E.1n(E.M(e,"2t"))a=E.M(e,"2t"
").16(e,b);9(!1b&&e["43"+d]&&e["43"+d].16(e,b)==P)a=P;9(4N)b.44();9(f&&f.16(e,b)==P)a=P;9(1b&&c!=P&
&a!=P&&!(E.11(e,\"a\")&&d=="4L")){6.4T=Q;e[d][0]6.4T=P}I a},2t:G(d){H a;d=E.1j.4M(d||18.1j||);H
b=d.O.2l(" ");d.O=b[0];H c=E.M(6,"2P")&&E.M(6,"2P")[d.O],3q=1B.3A.2J.2O(1q,1);3q.4w(d);L(H j 1i
c){3q[0].4Q=c[j];3q[0].M=c[j].M;9(!b[1]|c[j].O==b[1])H
e=c[j].16(6,3q);9(a!=P)a=e;9(e==P){d.2M();d.3p()}}9(E.V.1h)d.2m=d.2M=d.3p=d.4Q=d.M=S;I a},4M:G(c){H
a=c;c=E.1k({,a);c.2M=G(){9(a.2M)a.2M();a.7L=P};c.3p=G(){9(a.3p)a.3p();a.7K=Q};9(!c.2m&&c.65)c.2m=c.65;9(
E.V.1N&&c.2m.1y==3)c.2m=a.2m.12;9(!c.4K&&c.4J)c.4K=c.4J==c.2m?c.7H:c.4J;9(c.64==S&&c.63!=S){H
e=U.2V,b=U.1G;c.64=c.63+(e&&e.2R||b.2R||0);c.7E=c.7D+(e&&e.2B||b.2B||0)}9(!c.3Y&&(c.61|c.60))c.3Y=c.61|
|c.60;9(!c.5F&&c.5D)c.5F=c.5D;9(!c.3Y&&c.2r)c.3Y=(c.2r&1?1:(c.2r&2?3:(c.2r&4?2:0)));I
c};E.1b.1k({3W:G(c,a,b){I c=="5Y"?6.2G(c,a,b):6.N(G){E.1j.1f(6,c,b|a,b&&a)}},2G:G(d,b,c){I
6.N(G){E.1j.1f(6,d,G(a){E(6).5X(a);I(c|b).16(6,1q)},c&&b)}},5X:G(a,b){I 6.N(G){E.1j.28(6,a,b)}},1F:G(c,a,b){I
6.N(G){E.1j.1F(c,a,6,Q,b)}},7x:G(c,a,b){9(6[0])I E.1j.1F(c,a,6[0],P,b)},25:G(){H a=1q;I
6.4L(G(e){6.4H=0==6.4H?1:0;e.2M();I a[6.4H].16(6,[e])|P}},7v:G(f,g){G 4G(e){H
p=e.4K;1W(p&&p!=6)2a{p=p.12}29(e){p=6};9(p==6)I P;I(e.O=="4x"?f:g).16(6,[e])I
6.4x(4G).5U(4G);2d:G(f){5T();9(E.3T)f.16(U,[E]);J E.3l.1a(G){I f.16(6,[E])};I
6});E.1k({3T:P,3l:;2d:G(){9(!E.3T){E.3T=Q;9(E.3l){E.N(E.3l,G){6.16(U)};E.3l=S}9(E.V.35||E.V.34)U.4P("5S",E
.2d,P);9(!18.7t.K)E(18).39(G){E("#4E").28()}}};E.N(("7s,7r,39,7q,6n,5Y,4L,7p,"+7n,7m,7l,4x,5U,7k,24,"+51,
7j,7i,7h,3U").2l(" "),G(i,o){E.1b[o]=G(f){I f6.3W(o,f):6.1F(o)}},H x=P;G
5T){9(x)I;x=Q;9(E.V.35||E.V.34)U.4S("5S",E.2d,P);J 9(E.V.1h){U.7f("<7d"+"7y 22=4E 7z=Q
"+"3k=//:><\\V1J>");H a=U.3S("4E");9(a).62=G(){9(6.2C!="1l")I;E.2d();a=S}J
9(E.V.1N)E.4B=4j(G){9(U.2C=="5Q"||U.2C=="1l"){4A(E.4B);E.4B=S;E.2d();10};E.1j.1f(18,"39",E.2d)}E.1b.1k({



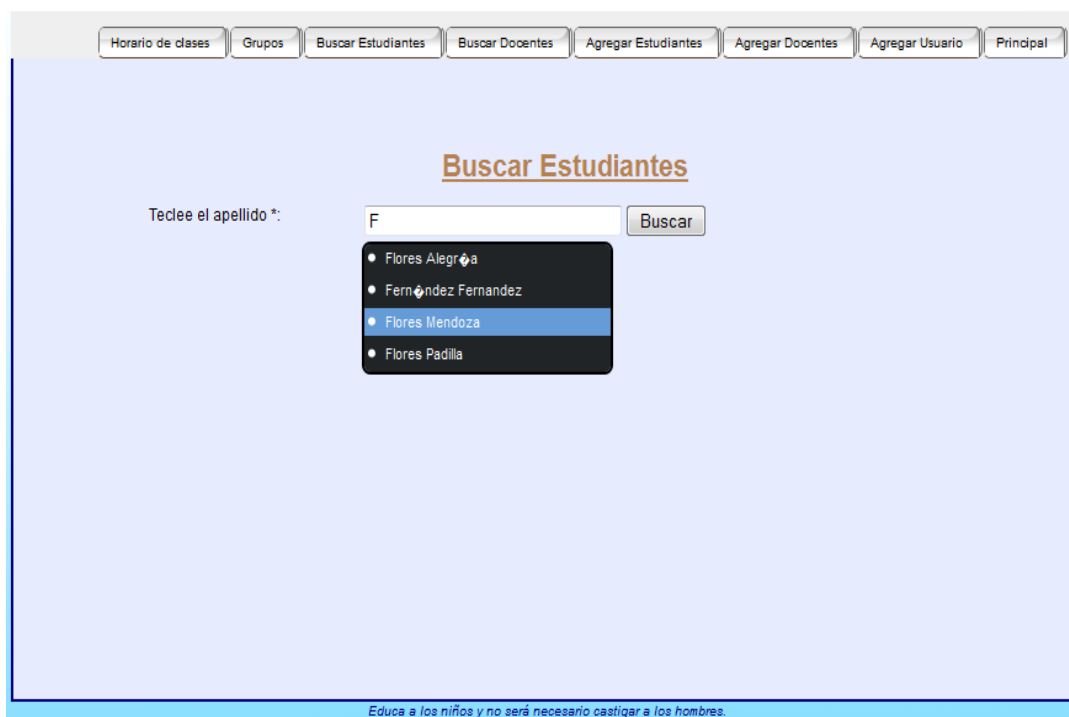
39:G(g,d,c){9(E.1n(g))l 6.3W("39",g);H e=g.1g(" ");9(e>=0){H i=g.2J(e,g.K);g=g.2J(0,e)}c=c||G(){};H
f="4z";9(d)9(E.1n(d)){c=d;d=S}J{d=E.3a(d);f="5P"}H
h=6;E.3G({1d:g,O:f,M:d,1l:G(a,b){9(b=="1C"||b=="5O")h.4o(i?E("<1s/>").3g(a.40.1p(/<1J(.\\s)*?\\V1J>/g,")).1Y(i
E.11(6,"2Y")?E.2h(6.79:6)}.1E(G()l 6.2H&&6.3c&&(6.2Q||/24|6b/i.14(6.11)||/2g|1P|52/i.14(6.O)))}.1X(G(i,c){H
b=E(6).3i();l
b==S?S:b.1c==1B?E.1X(b,G(a,i){l{2H:c.2H,1Q:a}}:{2H:c.2H,1Q:b}}.21()));E.N("5L,5K,6t,5J,5I,5H".2l(", "),G(i,o)
{E.1b[o]=G(f)l 6.3W(o,f)});H B=(1u 3D).3B();E.1k({21:G(d,b,a,c){9(E.1n(b)){a=b;b=S}
E.3G({O:"4z",1d:d,M:b,1C:a,1V:c}},78:G(b,a){l E.21(b,S,a,"1J")},77:G(c,b,a){l
E.21(c,b,a,"45")},76:G(d,b,a,c){9(E.1n(b)){a=b;b=f}}}
E.3G({O:"5P",1d:d,M:b,1C:a,1V:c}},75:G(a){E.1k(E.59,a)},59:{1Z:Q,O:"4z",2z:0,5G:"74/x-73-2Y-
72",6o:Q,3e:Q,M:S},49:{},3G:G(s){H f,2y=/=\\/?%3F)/g,1v,M;s=E.1k(Q,s,E.1k(Q,{},E.59,s));9(s.M&&s.6o&&1m
s.M!="1M")s.M=E.3a(s.M);9(s.1V=="4b"){9(s.O.2p()!="21"){9(!s.1d.1t(2y))s.1d+=(s.1d.1t(/\\/?)?&:"?")+ (s.4b||"5
E")+"="?"}}J
9(!s.M||!s.M.1t(2y))s.M=(s.M?s.M+"&:"")+ (s.4b||"5E")+"="?"s.1V="45"}9(s.1V=="45"&&(s.M&&s.M.1t(2y))|s.1d.1t
(2y)))f="4b"+B++;9(s.M)s.M=s.M.1p(2y,"="+f);s.1d=s.1d.1p(2y,"="+f);s.1V="1J";18[f]=G(a){M=a;1C();1l();18[f]=
W;2a{2E
18[f]}29(e){}}9(s.1V=="1J"&&s.1L==S)s.1L=P;9(s.1L===P&&s.O.2p()!="21")s.1d+=(s.1d.1t(/\\/?)?&:"?")+ "57=
"+(1u
3D());3B();9(s.M&&s.O.2p()!="21"){s.1d+=(s.1d.1t(/\\/?)?&:"?")+s.M;s.M=S}9(s.1Z&&!E.5b++)E.1j.1F("5L");9(!
s.1d.1g("8g")&&s.1V=="1J"){H h=U.4l("9U")l[0];H g=U.5B("1J");g.3k=s.1d;9(!f&&(s.1C||s.1l)){H
j=P;g.9R=g.62=G()l{9(!j&&(6.2C||6.2C=="5Q"||6.2C=="1l"))j=Q;1C();1l();h.3b(g)}h.58(g);l}H k=P;H i=18.6X?1u
6X("9P.9O"):1u 6W();i.9M(s.O,s.1d,s.3e);9(s.M)i.5C("9J-9l",s.5G);9(s.5y)i.5C("9H-5x-9F",E.49[s.1d]||"9D, 9C 9B
9A 5v:5v:5v 9z");i.5C("X-9x-9v",6W");9(s.6U)s.6U(i);9(s.1Z)E.1j.1F("5H",[i,s]);H
c=G(a){9(!k&&i.2C==4||a=="2z")k=Q;9(d){4A(d);d=S}1v=a=="2z"&&"2z"||!E.6S(i)&&"3U"||s.5y&&E.6R(i,s.1
d)&&"5O"||"1C";9(1v=="1C"){2a{M=E.6Q(i,s.1V)}29(e){1v="5k"}9(1v=="1C"){H b;2a{b=i.5s("6P-
5x")}}29(e){9(s.5y&&b)E.49[s.1d]=b;9(!f)1C()}J E.5r(s,i,1v);1l();9(s.3e)=S};9(s.3e){H
d=4j(c,13);9(s.2z>0)56(G){9(i){i.9q();9(!k)c("2z")},s.2z)}2a{i.9o(s.M)}29(e){E.5r(s,i,S,e)}9(!s.3e)c();l i;G
1C){9(s.1C)s.1C(M,1v);9(s.1Z)E.1j.1F("5l",[i,s])G 1l(){9(s.1l)s.1l(i,1v);9(s.1Z)E.1j.1F("6t",[i,s]);9(s.1Z&&!--
E.5b)E.1j.1F("5K")},5r:G(s,a,b,e){9(s.3U)s.3U(a,b,e);9(s.1Z)E.1j.1F("5J",[a,s,e]),5b:0,6S:G(r){2a{!r.1v&&9n.9l=
="54:"||r.1v>=6N&&r.1v<9j)||r.1v==6M||E.V.1N&&r.1v==W}29(e){l P},6R:G(a,c){2a{H b=a.5s("6P-5x");l
a.1v==6M||b==E.49[c]||E.V.1N&&a.1v==W}29(e){l P},6Q:G(r,b){H c=r.5s("9i-O");H
d=b=="6K"||!b&&c&c.1g("6K")>=0;H
a=d?r.9g:r.40;9(d&&a.2V.37=="5k")6G"5k";9(b=="1J")E.5f(a);9(b=="45")a=3w((" "+a+""));l a},3a:G(a){H
s=[];9(a.1c==1B||a.4c)E.N(a,G){s.1a(3f(6.2H)+"="+3f(6.1Q))};J L(H j 1i
a)9(a[j]&&a[j].1c==1B)E.N(a[j],G){s.1a(3f(j)+"="+3f(6))};J s.1a(3f(j)+"="+3f(a[j]));l
s.66("6").1p(/%20/g,"+"))};E.1b.1k({1A:G(b,a){l
b?6.1U({1H:"1A",2N:"1A",1r:"1A"},b,a):6.1E("1P").N(G){6.R.19=6.3h?6.3h:"";9(E.17(6,"19")=="2s")6.R.19="2Z"
}).2D(),1z:G(b,a){l
b?6.1U({1H:"1z",2N:"1z",1r:"1z"},b,a):6.1E("3R").N(G){6.3h=6.3h||E.17(6,"19");9(6.3h=="2s")6.3h="2Z";6.R.19
="2s"}).2D(),6J:E.1b.25,25:G(a,b){l
E.1n(a)&&E.1n(b)?6.6J(a,b):a?6.1U({1H:"25",2N:"25",1r:"25"},a,b):6.N(G){E(6)[E(6).3t("1P")?"1A":"1z"]())},9c:
G(b,a){l 6.1U({1H:"1A"},b,a)},9b:G(b,a){l 6.1U({1H:"1z"},b,a)},99:G(b,a){l 6.1U({1H:"25"},b,a)},98:G(b,a){l
6.1U({1r:"1A"},b,a)},96:G(b,a){l 6.1U({1r:"1z"},b,a)},95:G(c,a,b){l 6.1U({1r:a},c,b)},1U:G(k,i,h,g){H j=E.6D(i,h,g);l
6j.3L==P?"N":"3L"}(G){j=E.1k({},j);H f=E(6).3t("1P"),3y=6;L(H p 1i k){9(k[p]=="1z"&&f||k[p]=="1A"&&f)
E.1n(j.1l)&&j.1l.16(6);9(p=="1H"||p=="2N")j.19=E.17(6,"19");j.2U=6.R.2U}9(j.2U!=S)6.R.2U="1P";j.3M=E.1k({,
k);E.N(k,G(c,a){H e=1u E.2j(3y,j,c);9(/25|1A|1z/.14(a))e[a=="25"?f?"1A":"1z":a](k);J{H b=a.3s().1t(/^([+ -
])?(\\d+-.]+)(.*)\$/),1O=e.2b(Q)||0;9(b){H
d=3l(b[2]),2i=b[3]||"2T";9(2i=="2T"){3y.R[c]=(d||1)+2i;1O=((d||1)/e.2b(Q))*1O;3y.R[c]=1O+2i}9(b[1])d=((b[1]=="-
"?-1:1)*d)+1O;e.3N(1O,d,2i)}J e.3N(1O,a,""));l Q}},3L:G(a,b){9(E.1n(a)){b=a;a="2j"}9(!a||(1m a=="1M"&&b))l
A(6[0],a);l 6.N(G){9(b.1c==1B)A(6,a,b);J{A(6,a).1a(b);9(A(6,a).K==1)b.16(6)}},9f:G(){H a=E.32;l 6.N(G){L(H
i=0;i<a.K;i++)9(a[i].T==6)a.6l(i--,1)}.5n());H A=G(b,c,a){9(!b);H
q=E.M(b,c+"3L");9(!q||a)q=E.M(b,c+"3L",a?E.2h(a):[]);l q};E.1b.5n=G(a){a=a||"2j";l 6.N(G){H
q=A(6,a);q.44():9(q.K)q[0].16(6)};E.1k({6D:G(b,a,c){H
d=b&&b.1c==8Z?b:1l:c||c&&a||E.1n(b)&&b.2e:b,3J:c&&a||a&&a.1c==8Y&&a;d.2e=(d.2e&&d.2e.1c==4W?d.2e:
{8X:8W,8V:6N}[d.2e])||8T;d.3r=d.1l;d.1l=G()E(6).5n();9(E.1n(d.3r))d.3r.16(6);l d},3J:{6B:G(p,n,b,a){l
b+a*p},5q:G(p,n,b,a){l((-



38.9s(p*38.8R)/2)+0.5)*a+b}},32:[,2j:G(b,c,a){6.Y=c;6.T=b;6.1e=a;9(!c.3P)c.3P={}}};E.2j.3A={4r:G(){9(6.Y.2F)6.Y.2F.16(6.T,[6.2v,6]);(E.2j.2F[6.1e])|E.2j.2F.6z(6);9(6.1e=="1H"||6.1e=="2N")6.T.R.19="2Z"},2b:G(a){9(6.T[6.1e])!=S&&6.T.R[6.1e]==S)1.6.T[6.1e];H.r=3!(E.3C(6.T,6.1e,a));l.r&&r>-8O?:r.3!(E.17(6.T,6.1e))|0},3N:G(c,b,e){6.5u=(1u3D()).3B();6.1O=c;6.2D=b;6.2i=e||6.2i||"2T";6.2v=6.1O;6.4q=6.4i=0;6.4r();H.f=6;G.t(){f.2F()}.t.T=6.T;E.32.1a(t);9(E.32.K==1){H.d=4j(G){H.a=E.32;L(H.i=0;i<a.K;i++)9(!a[i])a.6l(i--,1);9(!a.K)4A(d)},13}},1A:G(){6.Y.3P[6.1e]=E.1x(6.T.R,6.1e);6.Y.1A=Q;6.3N(0,6.2b());9(6.1e=="2N"||6.1e=="1H")6.T.R[6.1e]="8N";E(6.T.1A()),1z:G(){6.Y.3P[6.1e]=E.1x(6.T.R,6.1e);6.Y.1z=Q;6.3N(6.2b(),0)},2F:G(){H.t=(1u3D()).3B();9(t>6.Y.2e+6.5u){6.2v=6.2D;6.4q=6.4i=1;6.4r();6.Y.3M[6.1e]=Q;H.a=Q;L(H.i.1i6.Y.3M)9(6.Y.3M[i]==Q)a=P;9(a){9(6.Y.19!=S){6.T.R.2U=6.Y.2U;6.T.R.19=6.Y.19;9(E.17(6.T,"19")=="2s")6.T.R.19="2Z"}9(6.Y.1z)6.T.R.19="2s";9(6.Y.1z||6.Y.1A)L(H.p.1i6.Y.3M)E.1x(6.T.R,p,6.Y.3P[p])}9(a&&E.1n(6.Y.1l))6.Y.1l.16(6.T);l.P}J{H.n=t-6.5u;6.4i=n/6.Y.2e;6.4q=E.3J[6.Y.3J]||(E.3J.5q?"5q":"6B")}(6.4i,n,0,1,6.Y.2e);6.2v=6.1O+((6.2D-6.1O)*6.4q);6.4r()}J}Q}};E.2j.2F={2R:G(a){a.T.2R=a.2v},2B:G(a){a.T.2B=a.2v},1r:G(a){E.1x(a.T.R,"1r",a.2v)},6z:G(a){a.T.R[a.1e]=a.2v+a.2i}};E.1b.6m=G(){H.c=0,3E=0,T=6[0],5t;9(T)8L(E.V){H.b=E.17(T,"2X")=="4F",1D=T.12,23=T.23,2K=T.3H,4f=1N&&3x(4s)<8J;9(T.6V){5w=T.6V();1f(5w.1S+38.33(2K.2V.2R,2K.1G.2R),5w.3E+38.33(2K.2V.2B,2K.1G.2B));9(1h){H.d=E("4o").17("8H");d=(d=="8G"||E.5g&&3x(4s)>=7)&&2||d;1f(-d,-d)}}J{1f(T.5l,T.5z);1W(23){1f(23.5l,23.5z);9(35&&^t[d|h]\$/i.14(1D.37)||4f)d(23);9(4f&&!b&&E.17(23,"2X")=="4F")b=Q;23=23.23}1W(1D.37&&!^1G|4o\$/i.14(1D.37)){9(!^8D|1l-9S.*\$/i.14(E.17(1D,"19")))1f(-1D.2R,-1D.2B);9(35&&E.17(1D,"2U")!="3R")d(1D);1D=1D.12}9(4f&&b)1f(-2K.1G.5l,-2K.1G.5z)}5t={3E:3E,1S:c}}1.5t;G.d(a){1f(E.17(a,"9T"),E.17(a,"8A"))}G.1f(l,t){c+=3x(l)||0;3E+=3x(t)||0}});',62,616,'|||||this|||if|||||function|var|return|else|length|for|data|each|type|false|true|style|null|elem|document|browser|undefined||options|||nodeName|parentNode||test|jQuery|a|pply|css|window|display|push|fn|constructor|url|prop|add|indexOf|msie|in|event|extend|complete|typeof|is|function|className|replace|arguments|opacity|div|match|new|status|firstChild|attr|nodeType|hide|show|Array|success|parent|filter|trigger|body|height|table|script|tbody|cache|string|safari|start|hidden|value|merge|left|break|animate|dataType|while|map|find|global||get|id|offsetParent|select|toggle|selected|toUpperCase|remove|catch|try|cur|all|ready|duration|done|text|makeArray|unit|fx|swap|split|target||pushStack|toLowerCase|nextSibling|button|none|handle|guid|now|stack|tbody|s|re|timeout|inArray|scrollTop|readyState|end|delete|step|one|name|nth|slice|doc|ret|prev|entDefault|width|call|events|checked|scrollTop|exec|px|overflow|documentElement|grep|position|form|block|removeData|rl|timers|max|opera|mozilla|trim|tagName|Math|load|param|removeChild|disabled|insertBefore|async|encodeURIComponent|append|oldblock|val|childNodes|src|readyList|multiFilter|color|defaultView|stopPropagation|args|old|toString|is|last|first|eval|parseInt|self|domManip|prototype|getTime|curCSS|Date|top||ajax|ownerDocument|parseFloat|easing|has|queue|curAnim|custom|innerHTML|orig|currentStyle|visible|getElementById|isReady|error|static|bind|String|which|getComputedStyle|responseText|oWidth|oHeight|on|shift|json|child|RegExp|ol|lastModified|isXMLDoc|jsonp|jQuery|previousSibling|dir|safari2|el|styleFloat|state|setInterval|radio|getElementsById|tagName|tr|empty|html|getAttribute|pos|update|version|input|float|runtimeStyle|unshift|mouseover|getPropertyValue|GET|clearInterval|safariTimer|visibility|clean|_ie_init|absolute|handleHover|lastToggle|index|fromElement|relatedTarget|click|fix|evt|andSelf|removeEventListener|handler|cloneNode|addEventListener|triggered|nodeIndex|unique|Number|classFilter|prevObject|selectedIndex|after|submit|password|removeAttribute|file|expr|setTimeou|t|_appendChild|ajaxSettings|client|active|win|sibling|deep|globalEval|boxModel|cssFloat|object|checkbox|parser|error|offsetLeft|wrapAll|dequeue|props|lastChild|swing|handleError|getResponseHeader|results|startTime|00|box|Modified|ifModified|offsetTop|evalScript|createElement|setRequestHeader|ctrlKey|callback|metaKey|contentType|ajaxSend|ajaxSuccess|ajaxError|ajaxStop|ajaxStart|serializeArray|init|notmodified|POST|loaded|appendTo|DOMContentLoaded|bindReady|mouseout|not|removeAttr|unbind|unload|Width|keyCode|charCode|onreadystatechange|clientX|pageX|srcElement|join|outerHTML|substr|zoom|parse|textarea|reset|image|odd|even|before|quickClass|quickID|prepend|quickChild|execScript|offset|scroll|processData|uuid|contents|continue|textContent|ajaxComplete|clone|setArray|webkit|nodeValue|fil_default|100|linear|href|speed|eq|createTextNode|throw|replaceWith|splice|_toggle|xml|colgroup|304|200|alpha|Last|httpData|httpNotModified|httpSuccess|fieldset|beforeSend|getBoundingClientRect|XMLHttpRequest|ActiveXObject|col|br|abbr|pixelLeft|urlencoded|www|application|ajaxSetup|post|getJSON|getScript|elements|serialize|clientWidth|hasClass|scr|clientHeight|write|relative|keyup|keypress|keydown|change|mousemove|mouseup|mousedown|right|dblclick|resize|focus|blur|frames|instanceof|hover|offsetWidth|triggerHandler|ipt|defer|offsetHeight|border|padding|clientY|pageY|Left|Right|toElement|Bottom|Top|cancelBubble|returnValue|detachEvent|attachEvent|substring|line|weight|animated|header|font|enabled|innerText|contains|only|size|gt|lt|uFFFF|u0128|417|inner|Height|toggleClass|removeClass|addClass|replaceAll|noConflict|in

sertAfter|prependTo|wrap|contentWindow|contentDocument|http|iframe|children|siblings|prevAll|nextAll|wrapInn
er|prev|Boolean|next|parents|maxLength|maxlength|readOnly|readonly|class|htmlFor|CSS1Compat|compatMod
e|compatible|borderTopWidth|ie|ra|inline|it|rv|medium|borderWidth|userAgent|522|navigator|with|concat|1px|100
00|array|ig|PI|NaN|400|reverse|fast|600|slow|Function|Object|setAttribute|changed|be|can|property|fadeTo|fade
Out|getAttributeNode|fadeIn|slideToggle|method|slideUp|slideDown|action|cssText|stop|responseXML|option|co
ntent|300|th|protocol|td|location|send|cap|abort|col|cos|tfoot|thead|With|leg|Requested|opt|GMT|1970|Jan|01|T
hu|area|Since|hr|If|Type|Content|meta|specified|open|link|XMLHTTP|Microsoft|img|onload|row|borderLeftWidth|
head|attributes'.split('').split(''),0,{}))

Una vez completado este ejercicio deberá funcionar de la siguiente manera:



The screenshot shows a web application interface with a navigation bar at the top containing buttons: "Horario de clases", "Grupos", "Buscar Estudiantes", "Buscar Docentes", "Agregar Estudiantes", "Agregar Docentes", "Agregar Usuario", and "Principal". The main content area has a light blue background and features the title "Buscar Estudiantes" in orange. Below the title, there is a label "Teclee el apellido *:" followed by a text input field containing the letter "F". To the right of the input field is a "Buscar" button. A dropdown menu is open below the input field, displaying a list of student names: "Flores Alegría", "Fernández Fernandez", "Flores Mendoza" (which is highlighted with a blue background), and "Flores Padilla". At the bottom of the page, there is a blue footer bar with the text "Educa a los niños y no será necesario castigar a los hombres."

La hoja de estilo que muestra el cajetín con las coincidencia tiene el nombre de **auto.css**:

```
body {  
    Tipo de Letra: Helvetica;  
    Tamaño: 11pixel;  
    Color de letra: #000;  
}  
  
h3 {  
    margen: 0px;  
    Relleno : 0px;  
}  
  
.suggestionsBox {  
    Posicion: relativo;  
    Izquierda: 200píxeles;  
    márgenes: 0px 0px 0px 0px;  
    ancho: 200px;  
    colo de fonder: #212427;  
    -moz-border-radius: 7px;  
    -webkit-border-radius: 7px;  
    bordes: 2px solid #000;  
    color de letra: #fff;  
}  
  
.suggestionList {  
    margen: 0pixel;  
    Relleno: 0pixel;  
}  
  
.suggestionList li {  
    margin: 0px 0px 3px 0px;  
    padding: 3px;  
    cursor: pointer;  
}  
  
.suggestionList li:hover {  
    Color de fondo: #659CD8;  
}
```



Práctica N^a 5

USO DE FRAMEWORKS

FRAMEWORKS.

Instrucciones de la práctica:

- Haber leído el capítulo 7 de la monografía.
- Se aconseja consultar al profesor sobre cualquier aspecto que no se comprenda sobre la práctica.

Objetivos:

- Aprender a utilizar frameworks para el desarrollo de aplicaciones web.
- Aumentar la rapidez para desarrollar un sitio web.
- Mejorar nuestro sitio, utilizando frameworks.

La práctica:

Se pretende que el estudiante se familiarice con la implementación de frameworks que le ayuden a la creación de sitios web de forma mas rápida, minimizando el tiempo de desarrollo de la interfaz gráfica del sitio.

Para cumplir con esta práctica se pide:

- Crear nuestro sitio pero utilizando de frameworks, aquí les dejamos algunos nombres:
 - Jaml CSS.
 - WYMstyle CSS.
 - YUI Grids CSS.
 - Boilerplate CSS.

[Registro](#) | [Contáctenos](#)

Sistema de Gestio de Académicos

Esperamos se comporte con decoro

Bienvenidos	Vision	Objetivos	profesores	Galeria de Títulos	Estudiantes	Asignatura	Aulas
Cursos Especiales Ingresar Login: Clave: Aceptar Cursos Especiales Pintura con acuarela Tocar el piano Matemática Poesía, escenografía y literatura. Add Text Remove Text	Bienvenido al sistema de gestión de académicos (SDGA) Este sistema le facilitara como profesor, conocer los diferentes horarios de clases, los grupos que tiene asignados, los días de la semana en los cuales tendrá afectación, las aulas en las que se deberán impartir las clases, etc. Para los alumnos, se pone a la disposición el nivel profesional y académico de sus docentes, así como también la ubicación de los recintos en los que se impartirán las asignaturas, ubicación de laboratorios, etc. Esperamos que disfrute su estancia en el sitio, esta información se ha facilitado para mejorar la comunicación del sector estudiantil, docente y administrativo. Rogamos se comporte con decoro. Add Text Remove Text					Vinculos de Interes. Vinculos de Interés Blog escolar Buscador de Google .com Daleya.com El nuevo diario.com Estudiantes Destacados Santiago Molina Denis Leopoldo Espinoza Aldo Martínez Alaro Mendiola Julio Medina Add Text Remove Text	

Footer with copyright notice and status information
Layout based on YAML

Ilustración 0.14 Ejemplo de como se ve nuestro sitio una vez que lo trabajamos con los frameworks.



Conclusiones

1. La elaboración de un documento teórico-práctico de apoyo para la asignatura Programación Orientada a la Web, será de mucha utilidad para los estudiantes y profesores de esta asignatura.
2. La ejemplificación de ejercicios en cada capítulo desarrollado, ayuda a asociar los conceptos teóricos con los aspectos prácticos.
3. La realización de un programa de prácticas guiadas que estén relacionadas con la teoría, ayudará en gran medida a los estudiantes a resolver dichas prácticas.

Recomendaciones

1. Incorporar una clase Electiva, que sea continuación de la asignatura Programación Orientada a la Web.
2. Aumentar la cantidad de horas clases de esta asignatura, en laboratorios de computación que presten las condiciones necesarias para que los estudiantes puedan concentrarse tanto en la parte teórica como en la práctica.
3. Impartir la asignatura Programación Orientada a la Web así como la Electiva a incorporar (que será continuación) en horas de la mañana, antes del medio día para garantizar que los alumnos tengan una mejor asimilación de todo el contenido que se imparte.
4. Proveer a los estudiantes de documentación en español, que les facilite el aprendizaje.
5. Estructurar las prácticas con POO.
6. Mejorar este trabajo, incorporando un capítulo completo del MVC con ejemplos e incorporarlo en la parte práctica.



Planificación para Autoestudio.

Capítulo	Nombre del Capítulo	Parte Teórica	Ejemplos	Solución de Práctica	Tiempo total de Estudio
Capítulo 1	HTML, CSS, HTML5.	3 horas	4 horas.	6 horas.	9 horas.
Capítulo 2	JavaScript.	4 hora.	6 hora.	6 horas	16.horas
Capítulo 3	DOM	4 horas.	3 horas.	4 horas.	11 horas.
Capítulo 4	PHP	6 horas.	10 horas.	10 horas.	26 horas.
Capítulo 5	XML.	2 horas	3 horas.	2 hora.	7 horas.
Capítulo 6	Ajax.	5 hora.	6 horas.	6 horas.	17 horas.
Capítulo 7	Frameworks.	1 hora.	2 horas.	1hora.	4 hora.
Total horas de estudio					90 horas



Planificación para el docente.

Nombre Clase	Numero del Capítulo	Nombre del Capítulo	Sábados por capítulo	Parte teórica Por sábado	Parte práctica Por sábado	Hora total del capítulo
POW1	Capítulo 1	HTML5, CSS.	4	5 horas.	5 horas.	10 horas.
	Capítulo 2	JavaScript.	5	7.5 horas.	5 horas.	12.5 horas
	Capítulo 3	DOM.	4	5 hora.	5 horas.	10 horas.
	Capítulo 4	XML.	3	5 horas.	2.5 hora.	7.5 horas.
POW 2	Capítulo 5	PHP.	8	12.5 horas.	7.5 horas.	20 horas.
	Capítulo 6	Ajax.	6	5 horas.	10 horas.	15 horas.
	Capítulo 7	Frameworks.	2	1 horas.	1 horas.	5 hora.
Total horas de clase						80 horas

Bibliografía

Libros:

Carles Mateu. Desarrollo de aplicaciones web.
Universitat Oberta de Catalunya. 2004.

Peter Lubbers, Brian Albers y Frank Salim. Pro HTML5 Programming
United States of America. 2010.

Ellie Quigley. JavaScript by Example.
Prentice hall. 2010.

Jeremy Keith. Dom Scripting.
United States of America. 2005.

Stoyan Stefanov. Object-Oriented JavaScript
Packt Publishing. 2008.

Jason Lengstorf. PHP for Absolute Beginners.
United States of America. 2009.

Matt Zandstra. PHP Objects Patterns, and Practice.
United States of America. 2010

David Powers. PHP Object-Oriented Solutions.
United States of America. 2008.

Bogdan Brinzarea-Iamandi., Cristian Darie., Audra Hendrix. AJAX y PHP.
Packt Publishing. 2009.

Lee Babin. Beginning Ajax with PHP.
United States of America. 2009.

Steven Holzner. Ajax Bible.
United States of America 2007.

Páginas Web:

1. www.w3school.com
2. <http://top12cssframeworksandhowtounderstandthem.htm>