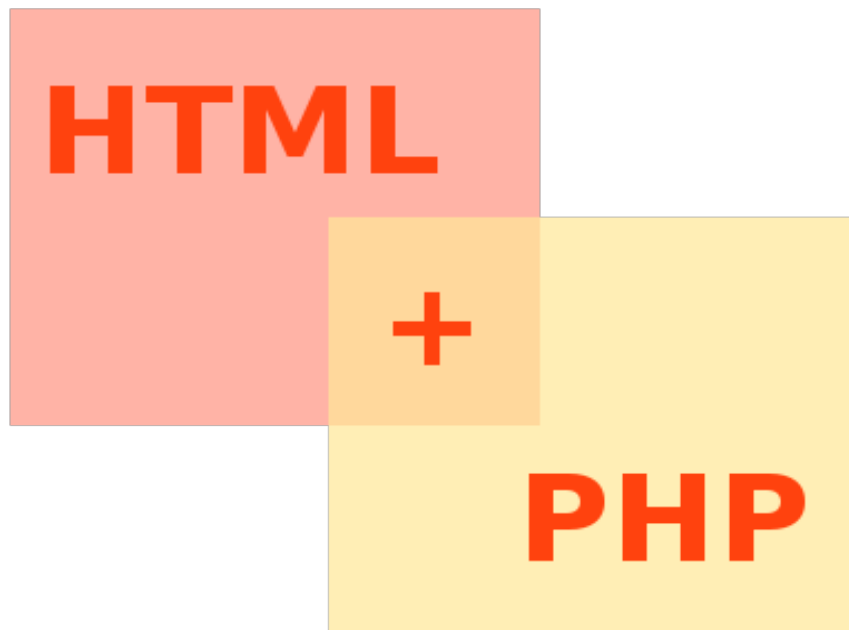


Fundamentos Básicos de



Versión 1.0 - Julio 12, 2008

Wil A. Alvarez R.

Fundamentos Básicos de HTML + PHP

Copyright (c) 2008 Wil Alejandro Alvarez Rodríguez.

Permiso para copiar, distribuir y/o modificar este documento bajo los términos de la GNU Free Documentation License, Version 1.2 o cualquier versión posterior publicada por la Free Software Foundation; sin secciones invariantes, sin textos de portada y sin textos de contraportada.

Se incluye una copia de la licencia en la sección titulada “Licencia de Documentación Libre GNU”, sino puede escribir a la Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA o visite la página <http://www.gnu.org/licenses/fdl.html>.

Historial de Cambios

| | | |
|----------------|-------------|-----------------------------------|
| Julio 12, 2008 | Version 1.0 | Publicación inicial del documento |
|----------------|-------------|-----------------------------------|

Índice de contenido

| | |
|--|----|
| Consideraciones Previas..... | 7 |
| ¿A quién va dirigida la guía?..... | 7 |
| ¿Cómo está estructurada la guía?..... | 7 |
| Convenciones..... | 7 |
| ¿Dónde comenzar? | 8 |
| Instalar Apache y PHP en Windows..... | 8 |
| Instalar Apache y PHP en GNU/Linux..... | 8 |
| Probando... probando..... | 9 |
| Capítulo 1. El comienzo..... | 10 |
| Capítulo 2. Características Básicas..... | 11 |
| Elementos HTML..... | 11 |
| Estructura básica..... | 12 |
| Sintaxis HTML..... | 14 |
| Capítulo 3. Texto..... | 16 |
| Párrafos..... | 16 |
| Sección de Titulares..... | 16 |
| Cursiva..... | 16 |
| Negrita..... | 16 |
| Subrayado..... | 16 |
| Tachado..... | 17 |
| Nueva Línea (Retorno de carro)..... | 17 |
| Cita Textual..... | 17 |
| Abreviatura..... | 17 |
| Acrónimo..... | 17 |
| Definiciones..... | 17 |
| Cita Externa..... | 18 |
| Texto Preformateado..... | 18 |
| Fragmentos de Código..... | 18 |
| Capítulo 4. Hiperenlaces..... | 21 |
| Enlaces relativos y absolutos..... | 21 |
| Otros enlaces..... | 23 |
| Capítulo 5. Listas..... | 25 |
| Listas no numeradas..... | 25 |
| Listas numeradas..... | 26 |
| Listas de definición..... | 27 |
| Capítulo 6. Imágenes..... | 30 |
| Capítulo 7. Tablas..... | 32 |
| Tablas Básicas..... | 32 |
| Capítulo 8. Estructuras (Layouts)..... | 36 |
| Capítulo 9. Comentarios..... | 37 |
| Capítulo 10. Formularios..... | 38 |
| Formulario..... | 38 |
| Controles de formulario..... | 38 |
| Cuadro de texto..... | 39 |

| | |
|---|----|
| Cuadro de contraseña..... | 39 |
| Checkbox..... | 40 |
| Radiobutton..... | 40 |
| Botón de envío de datos..... | 41 |
| Botón de restablecimiento de formulario..... | 41 |
| Botón Estándar..... | 42 |
| Botón de imagen | 42 |
| Archivos adjuntos | 43 |
| Campo oculto | 43 |
| Ejemplo completo..... | 43 |
| Área de texto..... | 45 |
| Lista desplegable..... | 45 |
| Capítulo 11. Introducción a PHP..... | 49 |
| ¿Qué es PHP?..... | 49 |
| ¿Cómo funciona PHP?..... | 49 |
| ¿Qué es un archivo PHP?..... | 50 |
| ¿Por qué PHP?..... | 50 |
| Capítulo 12. Sintaxis de PHP..... | 51 |
| Sintaxis Básica de PHP | 51 |
| Comentarios en PHP | 51 |
| Capítulo 13. Variables en PHP..... | 52 |
| Variables..... | 52 |
| Variables predefinidas..... | 52 |
| Constantes..... | 53 |
| Tipos de datos..... | 53 |
| Manipulación de tipos de datos..... | 54 |
| Capítulo 14. Echo..... | 55 |
| Echo con comillas..... | 56 |
| Echo con variables..... | 56 |
| Echo con variables y cadenas de caracteres..... | 56 |
| Capítulo 15. Cadenas de caracteres..... | 58 |
| strlen()..... | 58 |
| strpos()..... | 59 |
| strcmp()..... | 59 |
| substr()..... | 59 |
| str_replace()..... | 60 |
| Capítulo 16. Operadores en PHP..... | 62 |
| Operadores aritméticos..... | 62 |
| Operadores de asignación..... | 62 |
| Operadores de comparación..... | 63 |
| Operadores lógicos..... | 63 |
| Operador de cadenas de caracteres..... | 63 |
| Capítulo 17. If...else...elseif..... | 64 |
| Sentencia if..... | 64 |
| Una sentencia if verdadera..... | 64 |
| Una sentencia if falsa..... | 65 |

| | |
|--|-----|
| Sentencia if...else..... | 65 |
| Una sentencia if...else verdadera..... | 65 |
| Una sentencia if...else falsa..... | 66 |
| Sentencia elseif..... | 66 |
| Una sentencia elseif..... | 67 |
| Capítulo 18. Switch...case..... | 69 |
| Capítulo 19. Bucles..... | 73 |
| While..... | 73 |
| Do...while..... | 74 |
| Ejemplo con while()..... | 74 |
| Ejemplo con do...while()..... | 74 |
| For..... | 75 |
| Capítulo 20. Funciones..... | 77 |
| Creando una función..... | 77 |
| Usando una función..... | 77 |
| Funciones y parámetros de entrada..... | 78 |
| Funciones y valores de retorno..... | 79 |
| Capítulo 21. Manejo de formularios..... | 81 |
| La variable \$_GET..... | 81 |
| La variable \$_POST..... | 81 |
| PHP, formularios y ¡acción!..... | 82 |
| Capítulo 22. Caracteres Especiales..... | 88 |
| Apéndice A. Tabla de etiquetas..... | 90 |
| Apéndice B. Respuestas..... | 92 |
| Licencia de Documentación Libre GNU..... | 104 |
| Referencias Bibliográficas..... | 110 |

Consideraciones Previas

¿A quién va dirigida la guía?

Este documento va dirigido a todos aquellos que deseen iniciarse en la programación de páginas web.

Este material no pretende de ninguna manera ser una guía de contenidos extensos y detallados sobre HTML o PHP, por el contrario, ha sido concebido como un manual rápido en el cual el lector encontrará los tópicos claves para comenzar a programar en HTML y PHP.

Por esta razón se recomienda encarecidamente al lector complementar el contenido de esta guía con otras fuentes de información más detalladas y explícitas (referencias y manuales oficiales, tutoriales en línea, bibliografía especializada, entre otros) y por supuesto, ¡con mucha práctica!

¿Cómo está estructurada la guía?

Esta guía básica de programación web es la primera de las tres entregas que conforman todo el curso. La segunda entrega estudia las características básicas del lenguaje JavaScript y las Hojas de Estilo en Cascada (o Cascade Style Sheet en inglés), y su aplicación para controlar el aspecto visual de una página web; mientras que la última entrega abarca un estudio de PHP en un nivel intermedio y su integración con bases de datos MySQL para emplearlas en aplicaciones web.

Esta entrega consiste en una introducción al lenguaje XHTML (aunque el título diga lo contrario) y sus elementos principales; así como una introducción al lenguaje PHP y sus funciones básicas. Está compuesta de 22 capítulos y se divide en 3 partes.

La primera parte, comprendida entre los capítulos 1 y 10, abarca la sintaxis XHTML, las etiquetas más comunes, los efectos de texto, listas, tablas, hiperenlaces y formularios.

La segunda parte, que va desde el capítulo 11 al capítulo 20, explica los principios básicos de PHP, su sintaxis, variables, manejo de cadenas de caracteres, condicionales, bucles y funciones.

Por último, los capítulos 21 y 22 corresponden a la interacción entre XHTML y PHP, en otras palabras, al procesamiento de formularios. Además se explican algunas consideraciones que deben tomarse en cuenta con los caracteres especiales cuando se desarrollan páginas web.

Convenciones

En esta guía se usan diferentes artificios para mejorar la comprensión del contenido.

Los cuadros de código tienen un fondo de color gris claro y una fuente monospace, se emplean para distinguir los fragmentos de código HTML/PHP y diferenciarlos del texto normal.

Nota: Los cuadros de notas encierran consejos, trucos o advertencias que el usuario debe tomar en cuenta con mayor atención.

Ejercicio 0-0. Cuadros de ejercicios

Los cuadros de ejercicios indican una actividad propuesta para el lector referente al capítulo en curso

¿Dónde comenzar?

Antes de continuar debemos preparar nuestro entorno de desarrollo; eso incluye el servidor web Apache y el interprete de PHP. El servidor web es el encargado de atender las peticiones HTTP de los clientes y entregar las respuestas en HTML y el interprete de PHP se encarga de procesar los scripts y entregar los resultados HTML que el servidor web devolverá a los clientes.

Instalar Apache y PHP en Windows

Si desea instalar Apache visite <http://httpd.apache.org/download.cgi> y descargue la última versión de los binarios para Win32, ejecute y siga las instrucciones.

Para instalar PHP vaya a <http://www.php.net/downloads.php> y descargue la última versión de los binarios para Windows, ejecute y siga las instrucciones.

Nota: Haciendo una instalación individual de cada paquete la carpeta de documentos web de Apache queda localizada generalmente en **C:\directorio_de_instalación\Apache\htdocs**. Algunas versiones pueden variar y cambiar el nombre de la carpeta **htdocs** por **www**.

Existen instaladores que empaquetan todas estas herramientas en un solo archivo y permiten instalar y configurar todos estos servicios en poco más de un minuto. El paquete recomendado para seguir esta guía es el AppServ (<http://www.appservnetwork.com/>).

Puede descargarlo desde <http://prdownloads.sourceforge.net/appserv/appserv-win32-2.5.10.exe?download>

Nota: Haciendo la instalación con AppServ la carpeta de documentos web de Apache generalmente se localiza en **C:\AppServ\www**.

Instalar Apache y PHP en GNU/Linux

Para instalar estos paquetes en Debian y en distribuciones basadas en Debian (Ubuntu, Knoppix, etc) basta con ejecutar este comando con privilegios de root en un terminal:

```
# apt-get install apache2 php5
```

Para otras distribuciones consulte la documentación específica de su distribución o visite <http://httpd.apache.org/download.cgi> para obtener el código fuente y las instrucciones de compilación e instalación.

Probando... probando

Luego de instalar el servidor web Apache es necesario probar que esta completamente operativo. Para eso abrimos nuestro navegador y colocamos en la barra de direcciones: <http://localhost>. Eso debería cargarnos una página con la frase: **It Works!**

Nota: La URL **http://localhost** apunta al directorio local de los documentos web de Apache.

En el caso de Windows apunta a **C:\directorio_de_instalación\Apache\htdocs** cuando se instala Apache de manera individual o **C:\AppServ\www** cuando se instala con AppServ.

En el caso de GNU/Linux **http://localhost** apunta hacia el directorio **/var/www**.

Capítulo 1. El comienzo

HTML significa **HyperText Markup Language** o Lenguaje de Etiquetas de Hipertexto.

HTML no es un lenguaje de programación como Java, C/C++ o Python. Es un sistema de códigos usado para describir la estructura y el contenido en forma de texto (principalmente páginas web), así como para complementar el texto con objetos multimedia.

Un navegador web es un programa que le permite al usuario visualizar documentos de hipertexto (HTML) alojados en servidores web alrededor del mundo o en una red local.

El lenguaje HTML está basado en etiquetas y atributos que el navegador interpreta y las despliega en la pantalla. En ocasiones puede incluir un script (por ejemplo Javascript) que puede alterar el comportamiento de la página o del navegador.

Por convención, los archivos de formato HTML usan la extensión .htm o .html.

Capítulo 2. Características Básicas

Elementos HTML

Una **etiqueta** (tag, en inglés) es la “*instrucción*” que define el HTML para **marcar** los diferentes elementos que componen una página. Deben estar rodeadas por paréntesis angulares “< >” para que el navegador sea capaz de identificarlas y no confundirlas con el contenido de la página. Existen etiquetas de apertura y etiquetas de cierre.

<etiqueta>: Etiqueta de apertura

</etiqueta>: Etiqueta de cierre

Los **atributos** se incluyen dentro de la etiqueta de apertura para añadir información adicional sobre el elemento. Se expresan de la forma: atributo=“valor”

Nota: el valor del atributo **siempre** va entre comillas

Un **elemento HTML** está compuesto por una *etiqueta de apertura*, los *atributos*, el *contenido* y la *etiqueta de cierre*. Estos elementos se aplican al contenido del documento para cambiar la forma en que el navegador interpretará esa información en la pantalla.

```
<etiqueta atributo="valor_atributo">contenido</etiqueta>
```

Nota: Algunos elementos solo tienen etiqueta de apertura, sin atributos, contenido o etiqueta de cierre. Estos elementos son conocidos como *etiquetas vacías*.

Aunque cada etiqueta HTML define sus propios atributos, algunos son comunes a todas las etiquetas. A continuación se listan los **atributos básicos** para todos los elementos HTML:

| Atributo | Tipo | Descripción |
|-----------------|------------|--|
| id = “value” | Texto | Representa el identificador único de cada elemento dentro de la página HTML |
| class = “value” | Texto | Establece la clase de estilos CSS que se aplicará al texto |
| style = “value” | Estilo CSS | Establece el estilo CSS del elemento de forma directa |
| title = “value” | Texto | Establece el título de un elemento (mejora la accesibilidad y los navegadores lo muestran cuando el usuario pasa el ratón por encima del elemento) |

Los elementos HTML también responden a eventos, por lo que además de los atributos básicos también disponemos de **atributos de eventos**. Estos atributos permiten definir la acción a realizar cuando se produce un evento de ratón o teclado sobre el elemento y solo son válidos al usarlos con JavaScript. Debido a que el curso está orientado hacia HTML/PHP no entraremos en muchos detalles al respecto y bastará con describirlos brevemente.

| Atributo | Descripción |
|-------------|---|
| onclick | Pulsar y soltar un botón del ratón |
| ondblclick | Hacer doble clic |
| onmousedown | Pulsar un botón del ratón (sin soltar) |
| onmouseup | Soltar el botón del ratón que estaba pulsado |
| onmouseover | El ratón “entra” en un elemento (pasa por encima) |
| onmousemove | Mover el ratón |
| onmouseout | El ratón “sale” del elemento (pasa por encima de otro elemento) |
| onkeypress | Pulsar y soltar una tecla |
| onkeydown | Pulsar una tecla (sin soltar) |
| onkeyup | Soltar la tecla que estaba pulsada |

Estructura básica

Una página web está compuesta por dos secciones bien diferenciadas: el **encabezado** y el **cuerpo**. El encabezado contiene información sobre la propia página (como por ejemplo su título y su idioma) y es invisible al usuario (a excepción de título). El cuerpo de la página incluye todo el contenido que el usuario verá en pantalla, como párrafos de texto e imágenes.

La estructura básica de un documento HTML se puede ilustrar con el siguiente ejemplo:

```
<html>

<head>
<title>Mi primer documento HTML</title>
</head>

<body>
Este es el cuerpo de la página. Aquí se visualiza todo el contenido.
</body>

</html>
```

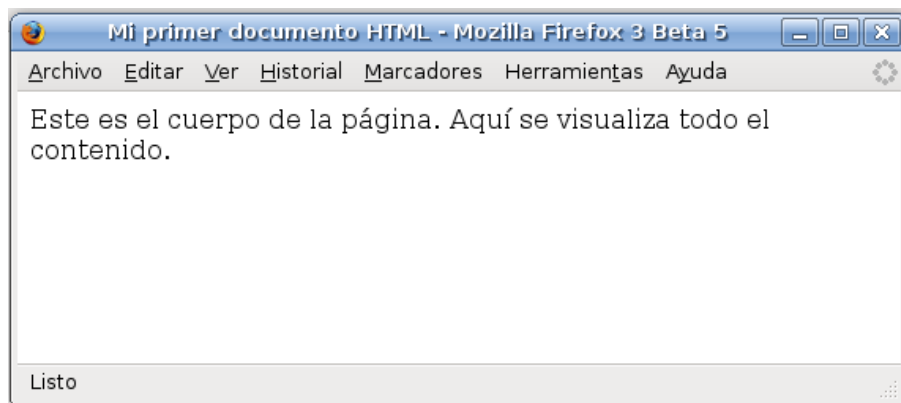


Figura 1. Primer documento HTML.

Se pueden apreciar claramente las etiquetas y los elementos básicos que componen la página. Un

documento HTML comienza con la etiqueta <html> y termina con </html>. El encabezado está delimitado por las etiquetas <head> y </head> y el cuerpo está delimitado por las etiquetas <body> y </body>.

Nota: Todo lo que esté fuera de las etiquetas <html> será ignorado.

Es importante aclarar que todos los efectos que se desean aplicar al texto se hacen a través de elementos HTML. Esto quiere decir que los espacios, tabulaciones y retornos de carro que se introduzcan en el archivo fuente no tendrán ningún efecto a la hora de la presentación del documento en el navegador.

```
<html>

<head>
<title>Mi primer documento HTML</title>
</head>

<body>
Texto escrito      con
      tabulaciones,  retornos de carro
y
espacios

Texto escrito sin tabulaciones, retornos de carro o espacios.
</body>

</html>
```

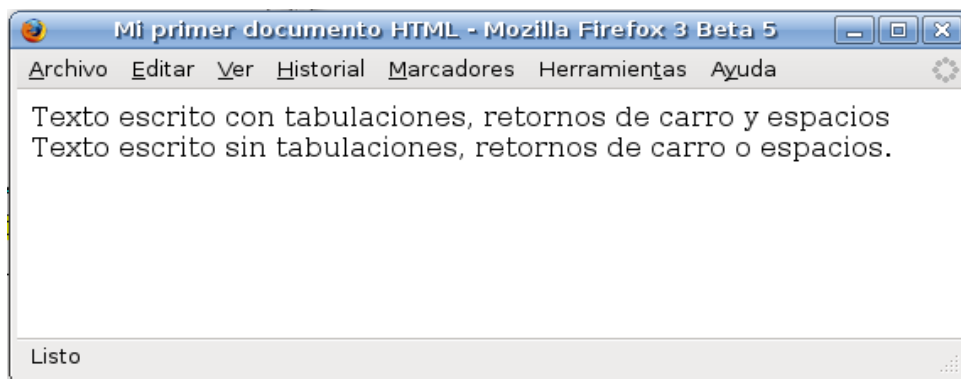


Figura 2. Ejemplo de tabulaciones, espacios y retornos de carro

En la Figura 2 se observa la salida del navegador para el código anterior. Se puede apreciar con facilidad que los espacios, tabulaciones y retornos de carro insertados en el documento HTML son ignorados por el navegador al momento de presentar la página. Esto es porque HTML establece una definición especial para éste tipo de caracteres.

Nota: Los espacios, tabulaciones y retornos de carro que se introduzcan en el archivo fuente de un documento HTML serán ignorados por el navegador al momento de presentar la página.

La siguiente tabla resume las etiquetas básicas de una página web:

| Etiqueta | Función |
|----------|---------------------------------|
| <html> | Empieza el documento HTML |
| <head> | Empieza la zona de encabezado |
| <title> | Empieza la zona de título |
| </title> | Termina la zona de título |
| </head> | Termina la zona de encabezado |
| <body> | Empieza el cuerpo del documento |
| </body> | Termina el cuerpo del documento |
| </html> | Termina el documento HTML |

Nota: Todas estas etiquetas son indispensables para que se visualice correctamente el documento HTML y para cumplir con el estándar XHTML.

Sintaxis HTML

El lenguaje HTML original es muy permisivo con su sintaxis y aunque esto pueda parecer algo bueno no lo es, pues trae como resultado páginas desordenadas, difíciles de mantener y poco profesionales.

Las últimas evoluciones del HTML (como HTML 4.01, XHTML 1.0 o XHTML 1.1) han definido cinco normas para la sintaxis de los elementos HTML. Es importante aprender y aplicar esas normas, no solo para estar acorde con los estándares sino para que nuestras páginas tengan un aspecto más elegante y profesional.

Las normas se listan a continuación y se explican brevemente con un ejemplo:

1. Las etiquetas se cierran en el mismo orden en que se abren

Incorrecto: <p>Éste es un párrafo y éste es un <a>enlace</p>

Correcto: <p>Éste es un párrafo y éste es un <a>enlace</p>

2. Los nombres de las etiquetas y sus atributos siempre se escriben en minúsculas.

Incorrecto: <p>Éste es un párrafo y éste es un enlace</p>

Correcto: <p>Éste es un párrafo y éste es un enlace</p>

3. El valor de los atributos siempre se encierra entre comillas.

Incorrecto: `<p>Éste es un párrafo y éste es un enlace</p>`

Correcto: `<p>Éste es un párrafo y éste es un enlace</p>`

4. Los atributos no se pueden comprimir.

Incorrecto: `<dl compact>...</dl>`

Correcto: `<dl compact="compact">...</dl>`

5. Todas las etiquetas deben cerrarse siempre.

En HTML existen etiquetas especiales llamadas **etiquetas vacías** que no encierran ningún contenido y no necesitan ser cerradas. Un ejemplo de esas etiquetas es `
` que se utiliza para iniciar una nueva línea y nunca encierra ningún contenido, tal y como se verá más adelante.

Sin embargo, el estándar obliga a cerrar todas las etiquetas abiertas, es decir que en el caso anterior la secuencia debería ser: `
</br>`. Afortunadamente XHTML permite escribir de una manera abreviada una etiqueta que se abre y se cierra consecutivamente. Solo basta con colocar `
` en lugar de `
</br>`.

Incorrecto: `
`

Correcto: `
`

Capítulo 3. Texto

En este capítulo se estudiarán las etiquetas más importantes que define HTML para estructurar y marcar el texto de un documento. Se explicarán brevemente las definiciones de cada etiqueta y se reforzará la explicación con un ejemplo.

Párrafos

| | |
|----------------------|--|
| <p> | Permite definir los párrafos que conforman un texto. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A |

Sección de Titulares

| | |
|--------------------------------|---|
| <h1>...<h6> | Define los títulos de mayor importancia de la página, enumerados en forma decreciente del 1 al 6. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A |

Cursiva

| | |
|----------------------|--|
| | Permite aplicar un énfasis cursivo al texto marcado. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A |

Negrita

| | |
|-----------------------|---|
| | Permite aplicar un efecto de negritas al texto marcado. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A |

Subrayado

| | |
|----------------------|---|
| <ins> | Se emplea para marcar una inserción de texto en el contenido original de la página web. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● cite - Indica la URL de la página donde se puede obtener más información sobre el motivo de la modificación del contenido.● datetime - Especifica la fecha y la hora en la que se realizó la modificación. |

Tachado

| | |
|----------------------|---|
| | Se emplea para marcar un borrado de texto en el contenido original de la página web. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● cite - Indica la URL de la página donde se puede obtener más información sobre el motivo de la modificación del contenido.● datetime - Especifica la fecha y la hora en la que se realizó la modificación. |

Nueva Línea (Retorno de carro)

| | |
|----------------------|---|
| | Inserta una nueva línea en la página web. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | Etiqueta vacía |

Cita Textual

| | |
|---------------------------|---|
| <blockquote> | Se emplea para indicar que el contenido marcado es una cita textual de otro contenido externo. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● cite - Indica la URL de la página original de donde se extrae la cita. |

Abreviatura

| | |
|----------------------|--|
| <abbr> | Se emplea para marcar las abreviaturas del texto y proporcionar su significado. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● title - Define el significado completo de la abreviatura. |

Acrónimo

| | |
|------------------------|--|
| <acronym> | Se emplea para marcar acrónimos o siglas y proporcionar su significado. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● title - Define el significado del acrónimo o de las siglas. |

Definiciones

| | |
|----------------------|---|
| <dfn> | Se emplea para marcar las definiciones de ciertos términos y proporcionar su significado. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● title - Indica el significado del termino. |

Cita Externa

| | |
|----------------------|--|
| <cite> | Se emplea para marcar una cita o referencia a otras fuentes. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A |

Texto Preformateado

| | |
|----------------------|--|
| <pre> | Muestra el texto tal cual como está escrito (respetando los espacios en blanco). |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A |

Fragmentos de Código

| | |
|----------------------|---|
| <code> | Se emplea para delimitar un fragmento de texto considerado código fuente. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A |

El siguiente código muestra a manera de ejemplo, como aplicar las etiquetas para dar formato a un texto determinado.

```
<html>
<head>
<title>Efectos de texto</title>
</head>
<body>

<h1>Titular de nivel 1</h1>
<h2>Titular de nivel 2</h2>
<h3>Titular de nivel 3</h3>
<h4>Titular de nivel 4</h4>
<h5>Titular de nivel 5</h5>
<h6>Titular de nivel 6</h6>

<p>
Este es el primer párrafo de este documento. El lenguaje <acronym title="Hyper Text Markup Language">HTML</acronym> permite aplicar efectos de <em>cursivas</em> y <strong>negritas</strong>, así como de <ins>subrayado</ins> y <del>tachado</del>. También permite insertar nuevas líneas<br/> y hacer citas textuales:
<blockquote>Oigo y olvido, veo y recuerdo, hago y aprendo</blockquote>
</p>

</body>
</html>
```

En la Figura 3 siguiente se puede apreciar el resultado del código anterior.

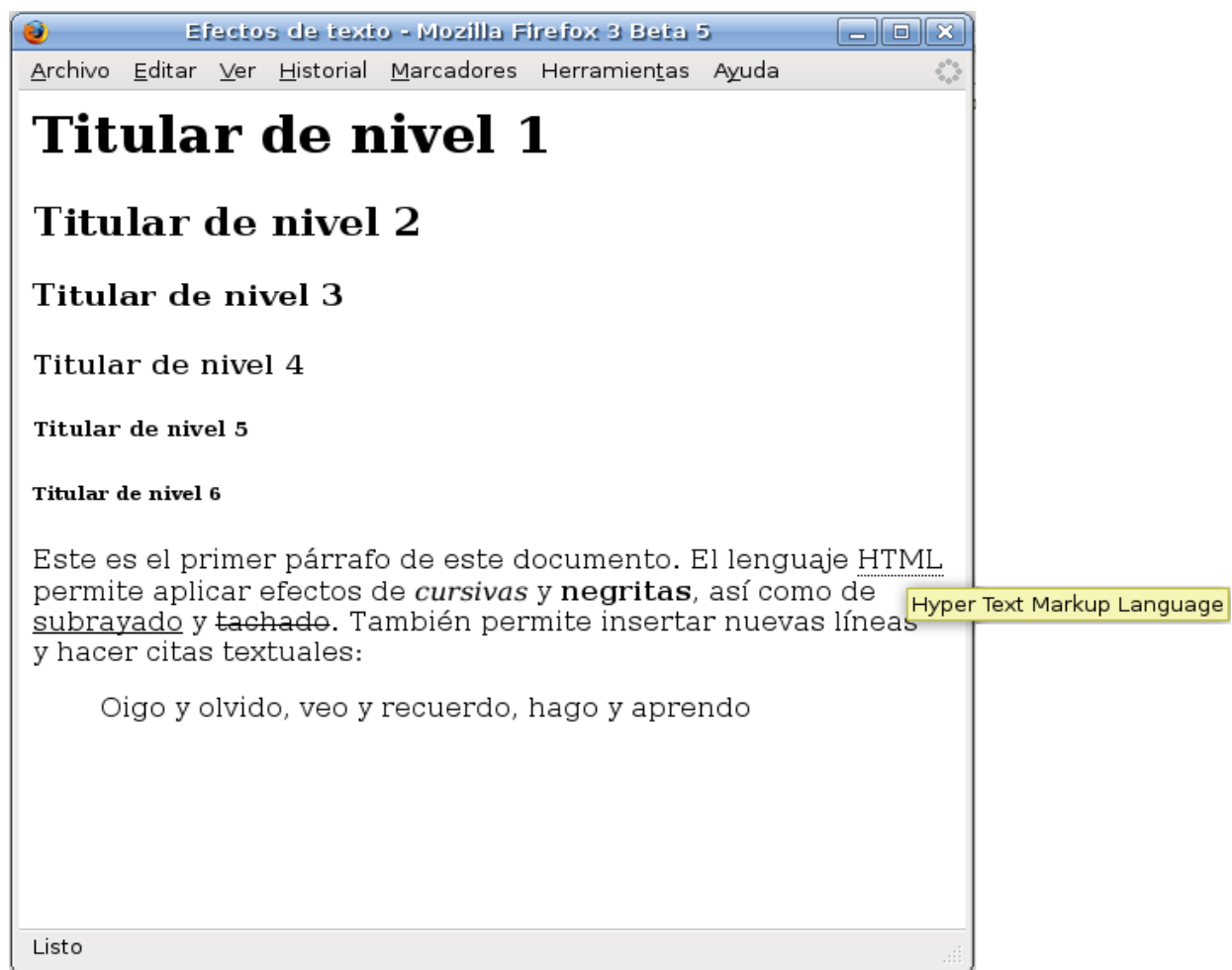


Figura 3. Resultado del código para el marcado de texto

Ejercicio 3-1. Marcado de Texto

Generar el código HTML que le permita obtener una página similar a la que se observa en la Figura 4.



Figura 4. Resultado deseado para el ejercicio de marcado de texto

Capítulo 4. Hiperenlaces

Los enlaces se utilizan para establecer relaciones entre dos recursos. Como HTML es un lenguaje de Hipertexto entonces los enlaces son denominados comúnmente **Hiperenlaces**. La mayoría de enlaces relacionan páginas web, pero también es posible enlazar otros recursos como imágenes, documentos y archivos.

Antes de empezar a incluir enlaces en las páginas HTML, es necesario comprender y dominar el concepto de URL. El acrónimo URL (del inglés Uniform Resource Locator) hace referencia al identificador único de cada recurso disponible en Internet.

Un URL se construye de la siguiente manera: **protocolo://máquina/ruta**. Donde **protocolo** puede ser http://, https://, ftp://, etc., la **máquina** es la dirección del equipo donde se encuentra el recurso y la **ruta** es el camino a seguir dentro de los directorios de la máquina para llegar hasta el recurso.

Un ejemplo típico de una URL sería:

```
http://www.miservidor.com/documentos/fisica/relatividad.html
```

Podemos apreciar en el ejemplo que el protocolo es **http://**, la máquina es **www.miservidor.com** y la ruta corresponde a **/documentos/fisica/relatividad.html**.

Conviene que nos detengamos momentáneamente en la estructuración habitual de los ficheros en un servidor web. Para empezar, siempre hay una página de bienvenida (home page o index) que podría compararse con la portada de un periódico o revista; la portada es lo primero que vemos.

La home page o index de cualquier servidor web, se carga con solo escribir en el navegador la dirección del servidor. Por ejemplo, para acceder a la página de bienvenida en nuestro ejemplo anterior basta con ir a la dirección <http://www.miservidor.com/>, para acceder a la página de bienvenida de la NASA habría que contactar con <http://www.nasa.gov/>.

Nota: La dirección del index (o dirección del servidor) se conoce como **URL de origen**

Enlaces relativos y absolutos

Un enlace relativo es aquel que enlaza documentos dentro de la misma máquina (locales) y se forma omitiendo algunas partes del URL. Si en una URL omitimos la dirección del servidor y solo nos quedamos con la ruta hacia el recurso obtendremos un enlace relativo.

Para que un enlace relativo tenga éxito es necesario que previamente el navegador conozca la **URL de origen** de forma tal que pueda realizar la búsqueda del recurso a partir de ese punto.

Un enlace absoluto es aquel que enlaza un documento de forma única y sin omitir información sobre su ubicación.

Ahora, supongamos que tenemos la siguiente página publicada en internet:

```
http://www.miservidor.com/mipagina1.html
```

Si queremos crear un enlace a otra página llamada *mipagina2.html* tenemos dos posibles resultados:

```
URL absoluta: http://www.miservidor.com/mipagina2.html
URL relativa: mipagina2.html
```

Al colocar la URL relativa el navegador buscará en la URL de origen; como habíamos accedido anteriormente a <http://www.miservidor.com/mipagina1.html> entonces el navegador tomará como URL de origen <http://www.miservidor.com/> y buscará en ese servidor una página llamada *mipagina2.html* para hacer el enlace. Si no se encuentra la página en ese servidor el navegador retornará el error 404 Not Found. Si por el contrario colocamos la URL absoluta entonces el navegador enlazará directamente el documento sin importar en que sitio este ubicado.

Ahora bien, si en el ejemplo anterior el navegador no ha cargado ningún documento del servidor <http://www.miservidor.com/> e intentamos realizar un enlace a *mipagina2.html* tendremos dos resultados. Para el URL relativo, el navegador nos dará el error 404 Not Found también, indicando que no pudo encontrar el servidor o la página web. Para el URL absoluto, el navegador cargará la página sin ningún inconveniente.

Nota: Los enlaces relativos son más cortos y fáciles de escribir pero se debe estar seguro de que el recurso existe en el servidor mencionado. Los enlaces absolutos enlazan de forma inequívoca los recursos pero son más largos y tediosos para escribir.

Para crear el enlace en el documento HTML usamos la siguiente etiqueta:

| | |
|----------------------|--|
| <a> | Se emplea para enlazar todo tipo de recursos. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● name - Permite nombrar al enlace con un identificador para que pueda ser accedido desde otros enlaces● href - Indica la URL del elemento que se quiere enlazar● target - Indica el comportamiento del navegador al enlazar el recurso. Puede tomar los siguientes valores:<ul style="list-style-type: none">○ <i>_blank</i> - Abrirá el recurso en una ventana nueva del navegador.○ <i>_parent</i> - Abrirá el recurso en la ventana padre si existe.○ <i>_self</i> - Abrirá el recurso en la misma ventana (default).○ <i>_top</i> - Abrirá el recurso en una ventana superior.○ <i>nombre_de_frame</i> - Abrirá el recurso en el frame (marco) cuyo nombre sea <i>nombre_de_frame</i>. |

Esta etiqueta puede usarse también para crear enlaces a correos electrónicos e incluso a un archivo que puede ser descargado por los usuarios. Para ello:

```
<a href="mailto:nombre@direccion.com" title="Dirección de email"> Enlace a email </a>

<a href="http://www.ejemplo.com/ruta/archivo.zip" title="Archivo a descargar"> Descarga
un ZIP con todos los contenidos </a>
```

Otros enlaces

Hasta ahora los enlaces estudiados necesitan la interacción del usuario (clic) para cargar los recursos. HTML define dos etiquetas (<script> y <link>) para recursos que se deben cargar automáticamente sin la intervención del usuario. A continuación se explican cada una de ellas:

| | |
|-----------------------|---|
| <script> | Se emplea para enlazar o definir un bloque de código (generalmente JavaScript). |
| Atributos Comunes | N/A |
| Atributos Especiales | <ul style="list-style-type: none">● src - Indica la dirección o URL del archivo que contiene el código● type - Permite avisar al navegador sobre el tipo de contenido a ejecutar (generalmente código JavaScript). |

Un ejemplo sencillo del uso de esta etiqueta sería:

```
<head>
  <script type="text/javascript" src="http://www.ejemplo.com/js/archivo.js"></script>
</head>
```

| | |
|----------------------|---|
| <link> | Se emplea para enlazar y establecer relaciones entre el documento y otros recursos |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● Etiqueta vacía.● href - Indica la URL del elemento que se quiere enlazar● rel - Describe la relación entre el documento actual y el contenido enlazado● type - Permite avisar al navegador sobre el tipo de contenido a ejecutar (generalmente código JavaScript). |

Un ejemplo de como usar esta etiqueta sería:

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

La etiqueta link tiene un uso muy particular. En muchas ocasiones podemos observar que algunas páginas disponen de un pequeño icono en la barra de direcciones. Ese icono se puede enlazar usando la etiqueta link. Para hacerlo basta con escribir lo siguiente:

```
<link rel="shortcut" href="http://www.servidor.com/nombre_icono.ico" type="image/ico"/>
```

Ejercicio 4-1. Prueba de enlaces

Determine el código HTML para generar una página con enlaces como la que se observa a continuación:

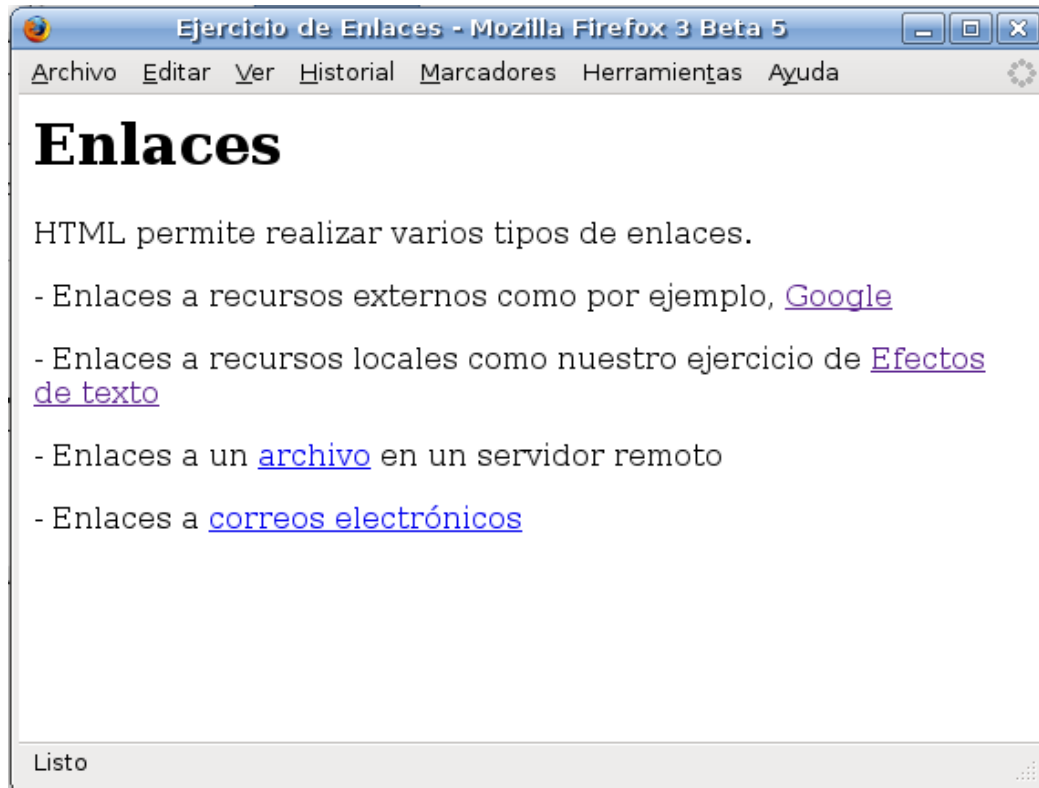


Figura 5. Resultado deseado para el ejercicio de enlaces

1. El primer enlace debe ir a la página de Google (www.google.co.ve)
2. El segundo debe ir a la página creada en el ejercicio anterior (efectos de texto)
3. El tercer enlace apuntará a un archivo indicado en el curso
4. El cuarto debe ir enlazado a la dirección de correo del estudiante

Capítulo 5. Listas

El lenguaje HTML define tres tipos diferentes de listas para agrupar los elementos: listas no numeradas, listas numeradas y listas de definición.

Listas no numeradas

Es el tipo más simple y común de las listas. Está conformado por un conjunto de elementos relacionados entre sí pero sin un orden o una secuencia determinada. Las etiquetas empleadas para este tipo de listas son: `` y ``

| | |
|--------------------------------|--|
| <code></code> | Se emplea para definir el inicio y el fin de una lista no ordenada |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A. |

| | |
|--------------------------------|--|
| <code></code> | Se emplea para definir cada elemento de lista. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A. |

Este es un ejemplo de una lista no numerada:

```
<html>
<head><title>Ejemplo de lista no numerada</title></head>
<body>
<h1>Lista de verduras</h1>

<ul>
  <li>Tomates</li>
  <li>Papas</li>
  <li>Lechugas</li>
</ul>

</body>
</html>
```



Figura 6. Ejemplo de lista no numerada

Listas numeradas

Son casi idénticas a las no numeradas, excepto que en este caso los elementos llevan un orden numerado y creciente. Las etiquetas para este tipo de listas son: `` y ``

| | |
|--------------------------------|---|
| <code></code> | Se emplea para definir el inicio y el fin de una lista ordenada |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A. |

| | |
|--------------------------------|--|
| <code></code> | Se emplea para definir cada elemento de lista. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A. |

Este es un ejemplo de una lista numerada:

```
<html>
<head><title>Ejemplo de lista numerada</title></head>
<body>

<h1>Instrucciones</h1>

<ol>
  <li>Enchufar correctamente</li>
  <li>Comprobar conexiones</li>
  <li>Encender el aparato</li>
</ol>

</body>
</html>
```

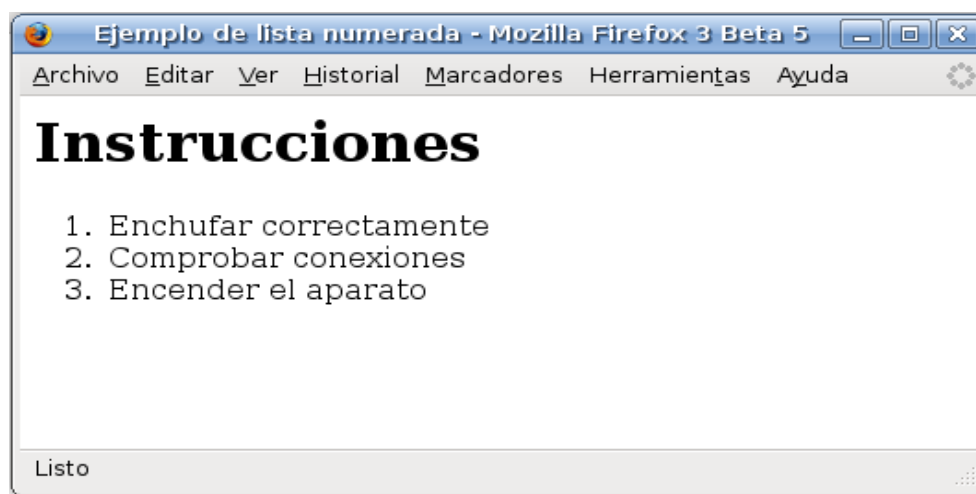


Figura 7. Ejemplo de una lista numerada

Listas de definición

Es un tipo de lista muy poco común y desconocido por muchos creadores de páginas web. Su funcionamiento es similar al de un glosario, ya que sus elementos están formados por un conjunto de términos y definiciones. Las etiquetas utilizadas son: <dl>, <dt> y <dd>

| | |
|----------------------|--|
| <dl> | Se emplea para definir el inicio y el fin de una lista de definiciones |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A. |

| | |
|----------------------|--|
| <dt> | Se emplea para definir los términos de una lista de definición |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A. |

| | |
|----------------------|---|
| <dd> | Se emplea para indicar las definiciones de los términos de la lista |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A. |

Este es un ejemplo de una lista de definiciones:

```
<html>

<head>
<title>Ejemplo de lista de definiciones</title>
</head>

<body>

<h1>Definiciones</h1>

<dl>
  <dt>Perro (<i>n. masc.</i>)</dt>
  <dd>Animal de cuatro patas que ladra.
  <dt>Gato (<i>n. masc.</i>)</dt>
  <dd>Animal de cuatro patas que maúlla y se lleva muy mal con el perro.
  <dt>Pescadilla (<i>n. fem.</i>)</dt>
  <dd>Animal que vive en el mar y está recubierto de escamas.
</dl>

</body>
</html>
```

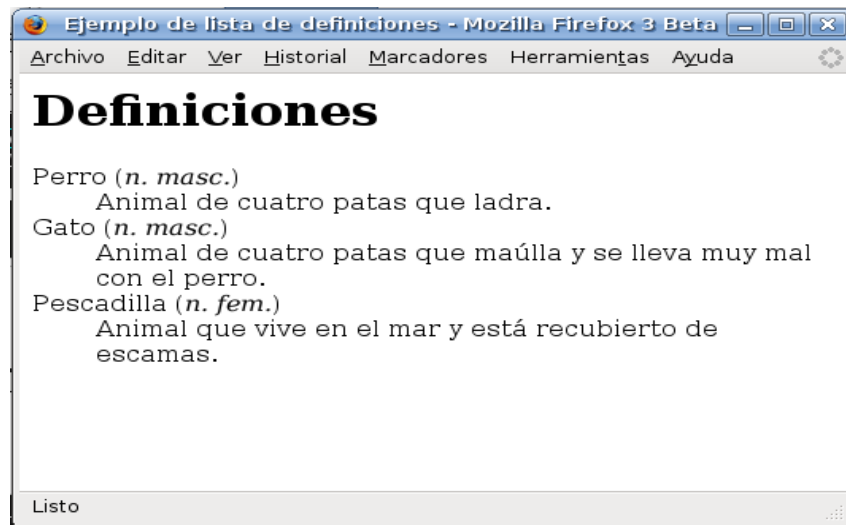


Figura 8. Ejemplo de lista de definiciones

Ejercicio 5-1. Listas anidadas

Genere una página que contenga las listas que indican en la imagen.

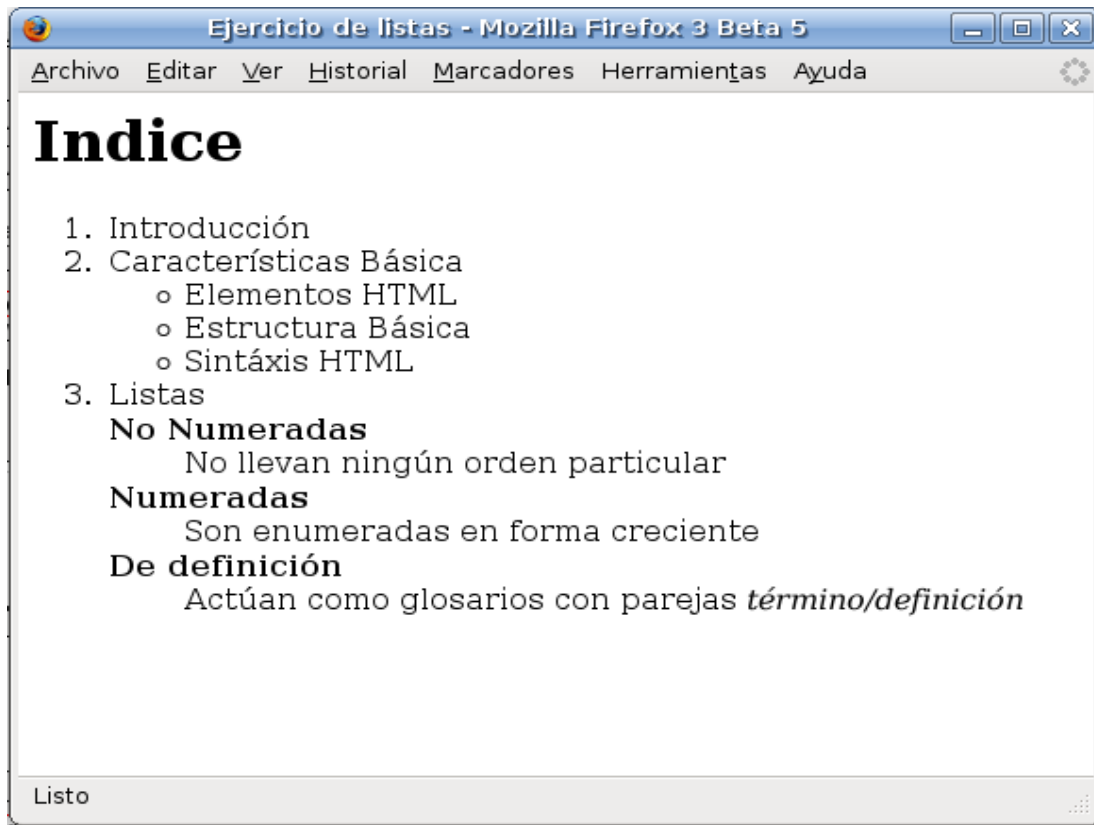


Figura 9. Resultado del ejercicio de listas

Pista: Use listas anidadas

Capítulo 6. Imágenes

Las imágenes son uno de los elementos más importantes de las páginas web. De hecho, prácticamente todas las páginas web contienen alguna imagen y la mayoría incluyen decenas de ellas. Dentro de las imágenes que se pueden incluir en una página HTML se deben distinguir dos tipos: las imágenes de contenido y las imágenes de adorno.

Las imágenes de contenido son las que proporcionan información y complementan el texto. Las imágenes de adorno son las que se utilizan para hacer bordes, para mostrar pequeños iconos en las listas de elementos, para mostrar fondos de página, etc.

Las imágenes de contenido se incluyen directamente en el código HTML mediante la etiqueta `` y las imágenes de adorno no se deberían incluir en el código HTML, sino que deberían emplearse hojas de estilos CSS para mostrarlas.

| | |
|---------------------------------|--|
| <code></code> | Se emplea para incluir imágenes en los documentos HTML. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● Etiqueta vacía● src - Indica la URL de la imagen que se muestra● alt - Descripción alternativa de la imagen● name - Nombre del elemento● height - Representa la altura con la que se debe mostrar la imagen● width - Representa el ancho con la que se debe mostrar la imagen● align - Indica la alineación de la imagen. Puede ser <i>left</i>, <i>center</i> o <i>right</i> |

Los dos atributos requeridos para que se muestre correctamente una imagen son **src** y **alt**. La URL indicada en *src* puede ser absoluta o relativa.

El atributo *alt* proporciona un texto alternativo a la imagen. Este texto aparecerá cuando se esté usando un navegador sin capacidades gráficas. Conviene utilizarlo cuando los gráficos sirven como origen a hiperenlaces, porque si la imagen no está disponible o el navegador no posee capacidades gráficas entonces se mostrará el texto en lugar de la imagen. En el atributo *alt* se puede describir muy brevemente la imagen a la que hace referencia (la descripción debe ser menor que 1.024 caracteres).

Para incluir una imagen basta con escribir un código como éste:

```

```

HTML no impone ninguna restricción sobre el formato gráfico que se puede utilizar en las imágenes, por lo que en principio la etiqueta `` puede incluir cualquier formato gráfico existente. Sin embargo, si la imagen utiliza un formato poco habitual, los navegadores no serán capaces de mostrar esa imagen.

Nota: Los formatos recomendados para cargar imágenes son: **GIF, JPG y PNG**. Estos formatos son manejados por todos los navegadores.

Los atributos *width* y *height* pueden indicarse usando números enteros o porcentajes. Si se utiliza un número entero se sobreentiende que la unidad de medida es el **píxel**. Si se utiliza porcentajes entonces se hace referencia a la altura/anchura del elemento en el que está contenida la imagen. Si la imagen no se encuentra contenida dentro de ningún elemento entonces se hace referencia a la altura/anchura total de la página.

```
<div>
  
</div>
```

El ejemplo anterior mezcla los dos tipos de medidas que se pueden utilizar, para indicar que la foto tiene una anchura igual al 55% de la anchura del elemento `<div>` que la contiene y una altura de 350 píxeles.

La anchura/altura con la que se muestra una imagen no tiene que coincidir obligatoriamente con la anchura/altura real de la imagen. Sin embargo, cuando estos valores no son proporcionales, las imágenes se muestran deformadas y el aspecto final es desagradable.

Nota: Es una buena práctica guardar todas las imágenes de contenido en un directorio especial independiente del resto de los contenidos HTML. Comúnmente este directorio es llamado "imagenes" o "images" (en inglés).

Capítulo 7. Tablas

Las tablas en HTML utilizan los mismos conceptos de filas, columnas, cabeceras y títulos que se utilizan en cualquier otro entorno de publicación de documentos. Pueden contener elementos simples, agrupaciones de filas y de columnas, cabeceras y pies de tabla, subdivisiones y otros elementos más complejos.

El uso de las tablas en HTML está destinado únicamente (aunque parezca obvio) a la presentación de información tabular. El problema con las tablas empezó porque desde hace unos años han sido usadas para definir la estructura completa de las páginas web. Esto ciertamente simplifica el diseño y permite ubicar los contenidos de una forma más rápida pero da como resultado una página lenta, tosca y con un acabado muy poco profesional.

Aunque algunos diseñadores siguen utilizando tablas para definir la estructura completa de las páginas web, se trata de una técnica obsoleta y nada recomendable. El motivo es que se complica en exceso el código HTML y su mantenimiento es muy complejo. La solución correcta para definir la estructura de las páginas consiste en la utilización de hojas de estilos CSS y estructuras (layouts).

Nota: Es una mala práctica utilizar tablas para definir la estructura de una página web, pues como resultado se obtiene una página lenta, complicada y muy difícil de mantener.

Tablas Básicas

Las tablas más sencillas de HTML se definen con tres etiquetas: `<table>`, `<tr>` y `<td>`.

La etiqueta `<table>` encierra todas las filas y columnas de la tabla, la etiqueta `<tr>` (del inglés *table row*) definen cada fila de la tabla y por último, la etiqueta `<td>` (del inglés *table data cell*) define cada una de las columnas de las filas, aunque realmente HTML no define columnas sino celdas de datos.

Normalmente, algunas de las celdas de la tabla se utilizan como cabecera de las demás celdas de la fila o de la columna. En este caso, HTML define la etiqueta `<th>` (del inglés *table header cell*) para indicar que una celda es cabecera de otras celdas.

Al definir una tabla, se debe pensar en primer lugar en las filas que la forman y a continuación en las columnas. El motivo es que HTML procesa primero las filas y por eso las etiquetas `<tr>` aparecen antes que las etiquetas `<td>`.

Por otra parte, HTML define la etiqueta `<caption>` para establecer la leyenda o título de una tabla. La etiqueta debe colocarse inmediatamente después de la etiqueta `<table>` y cada tabla sólo puede incluir una etiqueta `<caption>`.

| | |
|-----------------------------------|--|
| <code><table></code> | Se emplea para definir una tabla de datos |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● summary - Permite describir el contenido de la tabla (lo utilizan los buscadores y las personas discapacitadas) |

| | |
|----------------------|---|
| <tr> | Se emplea para definir cada fila de la tabla de datos |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A |

| | |
|----------------------|---|
| <td> | Se emplea para definir cada una de las celdas de datos de la tabla, es decir las columnas |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none"> ● headers - Indica la lista de las celdas que actuarán como cabeceras para esta celda ● scope - Indica las celdas para las que esta celda será su cabecera. Puede tomar los valores <i>col</i>, <i>row</i>, <i>colgroup</i>, <i>rowgroup</i>. ● colspan - Define el número de columnas que ocupará ésta celda ● rowspan - Define el número de filas que ocupará ésta celda ● abbr - Permite definir el contenido de la celda |

| | |
|----------------------|---|
| <th> | Se emplea para definir las celdas que son cabeceras de una fila o una columna en la tabla. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none"> ● headers - Indica las celdas que actuarán como cabeceras para esta celda ● scope - Indica las celdas para las que esta celda será su cabecera. Puede tomar los valores <i>col</i>, <i>row</i>, <i>colgroup</i>, <i>rowgroup</i>. ● colspan - Define el número de columnas que ocupará ésta celda ● rowspan - Define el número de filas que ocupará ésta celda ● abbr - Permite definir el contenido de la celda |

| | |
|------------------------|---|
| <caption> | Define el título o leyenda de una tabla |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A |

A continuación se muestra el código HTML de una tabla sencilla:

```
<html>
<head><title>Ejemplo de tabla sencilla</title></head>
<body>

<h1>Lista de cursos</h1>
```

```

<table>
<tr>
  <td><strong>Curso</strong></td>
  <td><strong>Horas</strong></td>
  <td><strong>Fecha</strong></td>
  <td><strong>Horario</strong></td>
</tr>

<tr>
  <td>HTML+PHP Básico</td>
  <td>20</td>
  <td>14 y 15 de junio 2008</td>
  <td>08:00 - 18:00</td>
</tr>

<tr>
  <td>PHP+MySQL</td>
  <td>20</td>
  <td>21 y 22 de junio 2008</td>
  <td>08:00 - 18:00</td>
</tr>

<tr>
  <td>CSS y JavaScript</td>
  <td>20</td>
  <td>28 y 29 de junio 2008</td>
  <td>08:00 - 18:00</td>
</tr>
</table>

</body>
</html>

```

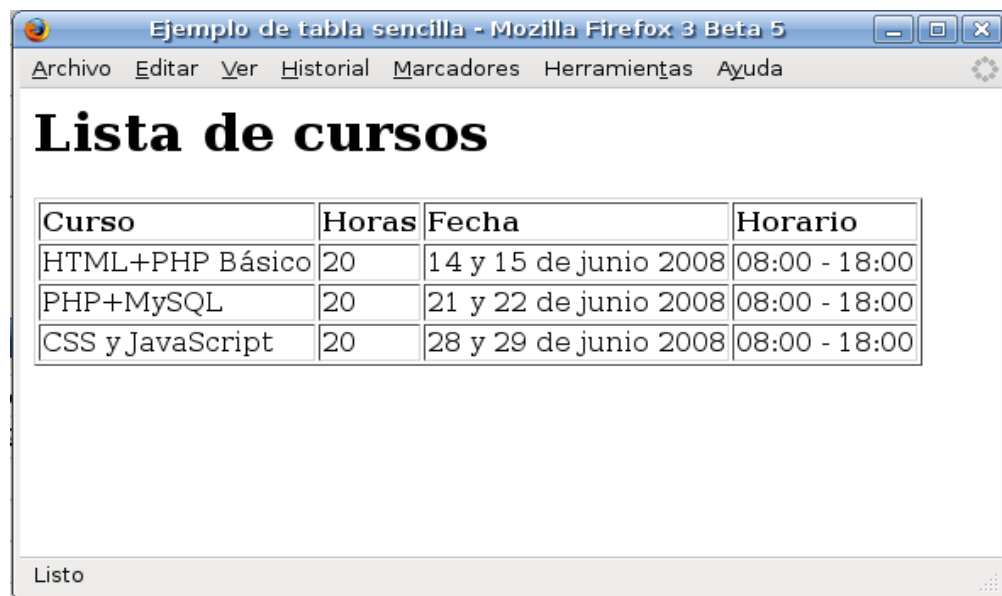
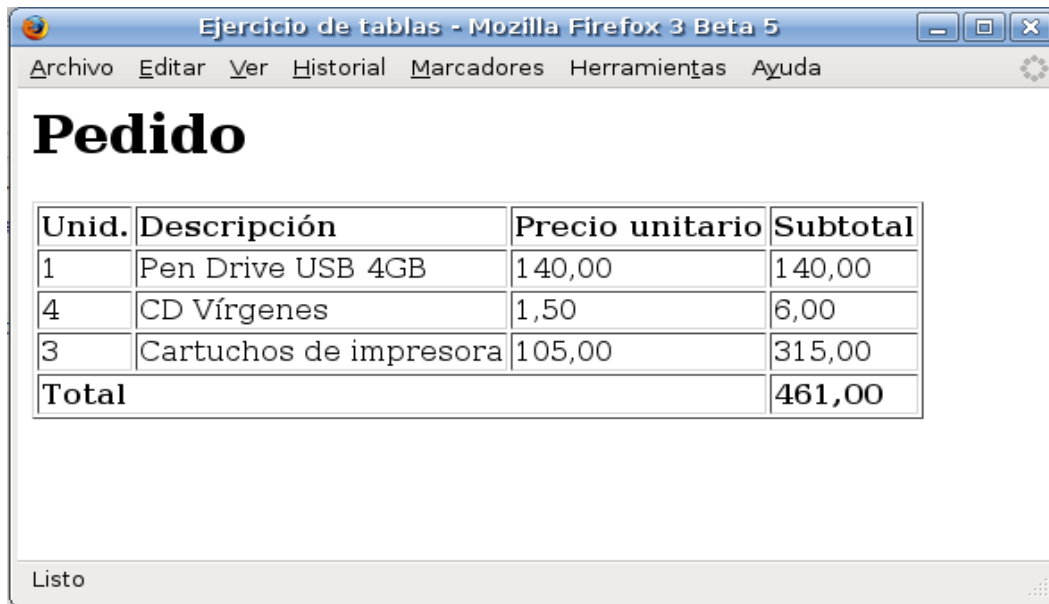


Figura 10. Ejemplo de una tabla HTML sencilla

Nota: HTML procesa primero las filas y luego las celdas (o columnas). Es por eso que se debe pensar primero en filas y luego en columnas.

Ejercicio 7-1. Tablas

Diseñe una tabla HTML similar a la que se muestra a continuación.



The screenshot shows a web browser window titled "Ejercicio de tablas - Mozilla Firefox 3 Beta 5". The browser's menu bar includes "Archivo", "Editar", "Ver", "Historial", "Marcadores", "Herramientas", and "Ayuda". The main content area displays the word "Pedido" in a large, bold font. Below it is a table with four columns: "Unid.", "Descripción", "Precio unitario", and "Subtotal". The table contains four data rows and a final "Total" row. The status bar at the bottom of the browser window shows the word "Listo".

| Unid. | Descripción | Precio unitario | Subtotal |
|-------|------------------------|-----------------|----------|
| 1 | Pen Drive USB 4GB | 140,00 | 140,00 |
| 4 | CD Vírgenes | 1,50 | 6,00 |
| 3 | Cartuchos de impresora | 105,00 | 315,00 |
| Total | | | 461,00 |

Figura 11. Resultado deseado para el ejercicio de tablas

Pista: Use los atributos **scope** y **colspan**

Capítulo 8. Estructuras (Layouts)

Hasta ahora, se han definido y marcado con HTML los diferentes elementos individuales que forman las páginas web (tablas, listas, enlaces, párrafos, imágenes, etc.). Sin embargo las páginas web suelen tener una estructura compleja que albergan cientos de elementos .

Utilizando exclusivamente HTML no es posible crear estas estructuras complejas. Es necesario emplear HTML junto con hojas de estilos CSS.

Para aplicar una hoja de estilos es necesario agrupar los contenidos de la página. La estrategia que se sigue es la de dividir la página en zonas según su finalidad, por ejemplo: la zona de la cabecera, la zona de contenidos, una zona lateral para el menú y otras secciones menores, la zona del pie de página, etc.

Para agrupar los elementos que forman cada zona de la página se utiliza la etiqueta `<div>`

| | |
|---------------------------------|---|
| <code><div></code> | Define zonas o divisiones en una página HTML. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | N/A |

Un ejemplo del uso de la etiqueta `<div>` sería:

```
<div id="cabecera">
...
</div>

<div id="menu">
...
</div>

<div id="pie">
...
</div>
```

No se va a profundizar en el proceso de diseñar una página web mediante `<div>`, ya que no es posible realizar ésta tarea sin utilizar hojas de estilos CSS. Dejaremos esos detalles para el siguiente módulo.

Capítulo 9. Comentarios

En la mayoría de los lenguajes (bien sea de programación o marcado) podemos incluir comentarios dentro del código fuente. HTML no es la excepción.

Normalmente, los diseñadores y programadores incluyen comentarios para marcar el comienzo y el final de las secciones de las páginas, para incluir avisos y notas para otros diseñadores, para incluir explicaciones sobre la forma en la que se ha creado el código HTML, etc.

Estos comentarios son invisibles en la página web, es decir no aparecen en pantalla, pero sí aparecen en el código HTML. Es por esto que debemos tener cuidado con la información que se incluye en esos comentarios.

Nota: Nunca debe colocarse información sensible o confidencial en los comentarios HTML

Para poner un comentario en un documento HTML se encierra el texto que formará el comentario entre los símbolos `<!--` y `-->`. Por ejemplo:

```
<html>

<head>
<!-- Aquí va el título -->
<title>Mi primer documento HTML</title>
</head>

<body>
<!-- Los comentarios HTML pueden ocupar
tantas líneas como sea necesario. Siempre
y cuando esté encerrado entre
los símbolos correspondientes -->
Este es el cuerpo de la página. Aquí se visualiza todo el contenido.
</body>

</html>
```

Nota: Los comentarios no se muestran por pantalla y por lo tanto no influyen en la forma en la que se ven las páginas.

Capítulo 10. Formularios

HTML es un lenguaje de marcado cuyo propósito principal es el de estructurar y formatear los contenidos de los documentos y páginas web. Sin embargo, dispone de elementos que permiten crear formularios para interactuar con el usuario. En este capítulo estudiaremos cada uno de estos elementos.

Formulario

Para crear un formulario se utiliza la etiqueta `<form>`. Esta etiqueta encierra todos los contenidos del formulario (botones, cuadros de texto, listas desplegables, etc.)

| | |
|----------------------|---|
| <form> | Define el comienzo y el fin de un formulario. |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● action - Indica la URL que se encargará de procesar los datos del formulario● method - Método HTTP empleado para enviar la información del formulario hacia la aplicación indicada en <i>action</i>. Puede ser <i>POST</i> o <i>GET</i>.● enctype - Tipo de codificación empleada al enviar el formulario al servidor (sólo se indica de forma explícita en los formularios que permiten adjuntar archivos) . Puede ser <i>application/x-www-form-urlencoded</i> o <i>multipart/form-data</i>● accept - Lista separada por comas de todos los tipos de archivos aceptados por el servidor (sólo para los formularios que permiten adjuntar archivos)● Otros: <i>accept-charset</i>, <i>onsubmit</i>, <i>onreset</i> |

Los atributos más empleados en un formulario son **action** y **method**. El atributo **action** indica la URL de la aplicación que se encargará de procesar los datos introducidos por los usuarios. Esta aplicación también se encarga de generar la respuesta que muestra el navegador y puede estar escrita en diferentes lenguajes de programación. En nuestro caso trabajaremos con PHP.

El atributo **method** establece la forma en la que se envían los datos del formulario al servidor. Los valores que puede tomar este atributo son **GET** y **POST**. La diferencia principal entre ellos es que GET añade los datos enviados al final de la URL y por lo tanto son visibles en la ventana del navegador, mientras que POST los envía como parte de las cabeceras de la solicitud HTTP y no se pueden ver tan fácilmente.

Existe una regla general que ayuda a los diseñadores de páginas web a elegir entre un método y otro. La regla dice que el método GET se debe utilizar en los formularios que no modifican la información y el método POST se debería utilizar cuando el formulario modifica la información original.

Otro atributo de la etiqueta `<form>` que se utiliza ocasionalmente es **enctype**. Este atributo es imprescindible en los formularios que permiten adjuntar archivos, pues deben indicar que su codificación es *"multipart/form-data"*.

Controles de formulario

HTML define varios tipos de controles (cuadros de texto, listas desplegables, áreas de texto, etc).

Es lógico pensar que existe una etiqueta para cada uno de ellos, pero no es así. La etiqueta `<input>` se utiliza para crear diez tipos de controles diferentes junto con otras etiquetas como `<select>`, `<option>` y `<textarea>`.

Por esta razón, estudiaremos la etiqueta `<input>` con todos sus atributos y luego se expondrán los ejemplos de cómo crear los controles.

| | |
|-----------------------------------|--|
| <code><input></code> | Se emplea para insertar un control en un formulario |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● Etiqueta vacía.● type - Indica el tipo de control que se insertará en el formulario.● name - Asigna un nombre único al control (es imprescindible para que el servidor pueda procesar el formulario).● maxlength - Indica el máximo número de caracteres permitido por el control (solo para los cuadros de texto).● checked="checked" - Permite indicar que opción aparecerá preseleccionada al cargar el elemento (solo para los controles checkbox y radiobutton)● disabled="disabled" - Deshabilita el control y su contenido no se envía a la aplicación web.● src - Indica la URL de la imagen que se emplea para el botón (solo para los botones de imágenes)● alt - Descripción del control● readonly="readonly" - Evita que el contenido del control pueda ser modificado● value - Valor inicial del control |

Cuadro de texto

Es el elemento más utilizado en los formularios y consiste de un cuadro en el cual el usuario puede escribir una línea de texto.

```
Nombre: <input type="text" name="nombre" value="" />
```

Nombre:

Figura 12. Cuadro de texto

Nota: Los cuadros de texto sirven para escribir una línea de texto en un formulario

Cuadro de contraseña

Es similar al cuadro de texto, con la diferencia que el texto escrito en el cuadro no se ve en pantalla sino que es reemplazado por asteriscos o círculos. Este control se obtiene asignando al atributo `type` el valor de `password`.

```
Contraseña: <input type="password" name="nombre" value="" />
```

Contraseña:

Figura 13. Cuadro de contraseña

Nota: Los cuadros de contraseña son ideales para escribir información confidencial o sensible

Checkbox

Los checkbox o "casillas de verificación" son controles de formulario que permiten al usuario seleccionar opciones individualmente. Generalmente, se utilizan cuando el usuario puede activar y desactivar varias opciones relacionadas pero no excluyentes.

En este caso el valor del atributo *value* no es el valor inicial que se muestra en pantalla. El valor de *value*, junto con el valor del atributo *name*, es la información que se envía al servidor. Además el atributo *type* debe tener el valor *checkbox*.

Si no se añade un texto al lado de la etiqueta `<input>` del checkbox, el usuario sólo verá un pequeño cuadrado sin ninguna información referente a ese checkbox por lo que es recomendable agregar un texto a manera de título para el elemento.

Si se quiere mostrar al inicio alguna opción seleccionada por defecto, se utiliza el atributo *checked="checked"*.

```
Hobbies<br/>
<input name="beber" type="checkbox" value="beber"/>Beber con los amigos<br/>
<input name="basket" type="checkbox" value="basket"/>Jugar basket<br/>
<input name="musica" type="checkbox" value="musica"/>Escuchar música<br/>
```

Hobbies

- ☐ Beber con los amigos
- ☐ Jugar basket
- ☐ Escuchar música

Figura 14. Casillas de verificación

Nota: Los checkbox se emplean cuando el usuario puede seleccionar más de una opción a la vez

Radiobutton

Los controles de tipo radiobutton son similares a los controles de tipo checkbox, a excepción de que son mutuamente excluyentes. En otras palabras, solo una de las opciones presentadas puede estar seleccionada a la vez. Cada vez que se selecciona una opción, automáticamente se deselecta la opción que estaba seleccionaba anteriormente.

El valor del atributo *type* para estos controles de formulario es *radio*. La forma de indicar que varios radiobutton pertenecen al mismo grupo consiste en asignar el mismo valor al atributo *name* de todos esos radiobutton, tal y como se muestra en el código HTML siguiente.


```
Sexo <br/>
<input type="radio" name="sexo" value="hombre" checked="checked" /> Hombre <br/>
<input type="radio" name="sexo" value="mujer" /> Mujer <br/>
```

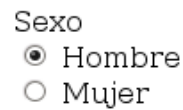


Figura 15. Radiobutton

Nota: Los radiobutton se utilizan cuando el usuario solamente puede escoger una opción entre las distintas opciones que se le presentan.

Botón de envío de datos

La mayoría de formularios disponen de un botón para enviar los datos introducidos al servidor y así obtener una respuesta adecuada. Este botón es conocido como botón “submit” o botón de envío.

El valor del atributo *type* para este control de formulario es *submit*. El navegador se encarga de enviar automáticamente los datos cuando el usuario pulsa sobre este tipo de botón. El valor del atributo *value* es el texto que muestra el botón. Si no se establece el atributo *value*, el navegador muestra un texto predefinido.

```
<input type="submit" name="enviar" value="Enviar" />
```



Figura 16. Botón de envío de datos

Nota: El botón de submit se encarga de enviar toda la información de los campos del formulario a la aplicación que gestionará los datos en el servidor.

Botón de restablecimiento de formulario

Este tipo de botón permite borrar todos los datos introducidos en el formulario. El valor del atributo *type* para este control de formulario es *reset*.

El navegador se encarga de borrar automáticamente toda la información cuando el usuario pulsa este tipo de botón. El atributo *value* se puede utilizar para cambiar el texto predefinido del botón.

```
<input type="reset" name="limpiar" value="Borrar datos del formulario" />
```

Borrar datos del formulario

Figura 17. Botón de restablecimiento de formulario

Nota: El botón de reset borra toda la información de los campos del formulario cuando es presionado

Botón Estándar

Además de los botones para enviar y borrar datos, existe otro tipo de botones que es muy útil cuando se programan aplicaciones web con JavaScript.

El valor del atributo *type* para este control de formulario es *button*. Si pruebas a pulsar un botón de este tipo, verás que el navegador no hace nada; no envía los datos al servidor y no borra los datos introducidos. Este tipo de botones sólo son útiles si se utilizan junto con JavaScript, de forma que se pueda definir la acción que se ejecuta cuando se pulsa el botón.

```
<input type="button" name="custom" value="Acción Personalizada"/>
```

Acción Personalizada

Figura 18. Botón estándar

Nota: Estos botones permiten definir acciones personalizadas en el formularios usando JavaScript.

Botón de imagen

Estos botones son similares a los botones estándares pero con la diferencia de que pueden mostrar imágenes en lugar de la forma tradicional del botón.

El valor del atributo *type* para este control de formulario es *image*. Este tipo de control permite cambiar por completo el aspecto con el que se muestran los botones del formulario, ya que los navegadores muestran la imagen cuya URL se indica en el atributo *src* en vez de los botones habituales.

Su principal ventaja es que permite personalizar por completo la estética de los botones y mostrarlos con un aspecto homogéneo en todos los navegadores. El principal inconveniente es que ralentiza la carga del formulario y que si se quiere modificar su valor, es necesario crear una nueva imagen.

```
<input type="image" name="enviar" src="accept.png" />
```



Figura 19. Botón de imagen

Nota: Los botones de imagen son ideales para personalizar el aspecto de la página y darle una apariencia acorde con la decoración de la misma.

Archivos adjuntos

Este control permite incluir archivos para subirlos al servidor. El valor del atributo *type* para este control de formulario es *file*. El navegador se encarga de mostrar un cuadro de texto donde aparece el nombre del archivo seleccionado y un botón que permite navegar por los directorios y archivos del ordenador del usuario.

```
Fichero adjunto <br/>
<input type="file" name="adjunto" />
```

Fichero adjunto




Figura 20. Control para adjuntar archivos

Nota: Cuando se usa este tipo de control es imprescindible añadir el atributo *enctype* al formulario, de forma que la etiqueta `<form>` debe contener el atributo `enctype="multipart/form-data"`, de lo contrario los archivos que se intenten adjuntar no se enviarán al servidor.

Campo oculto

Los campos ocultos incluyen información que se envía al servidor junto con el resto de datos del formulario pero no se muestran en pantalla, de forma que el usuario desconoce que el formulario los incluye. El valor del atributo *type* para este control de formulario es *hidden*. En general, los campos ocultos se utilizan para guardar información que necesita el servidor pero que no es necesario o no es posible que la establezca el usuario.

```
<input type="hidden" name="oculto" value="cualquier_valor" />
```

Ejemplo completo

El siguiente código ilustra como instanciar cada uno de los controles estudiados hasta ahora.

```
<html>
<head>
<title>Formularios</title>
</head>
<body>

<h3>Controles de formulario generados por input</h3>

<form>
Nombre: <input type="text" name="nombre" value="" /><br/><br/>

Contraseña: <input type="password" name="clave" value="" /><br/><br/>

Hobbies<br/>
<input name="beber" type="checkbox" value="beber"/>Beber con los amigos<br/>
<input name="basket" type="checkbox" value="basket"/>Jugar basket<br/>
<input name="musica" type="checkbox" value="musica"/>Escuchar música<br/><br/>
```

```

Sexo <br/>
<input type="radio" name="sexo" value="hombre" checked="checked" /> Hombre <br/>
<input type="radio" name="sexo" value="mujer" /> Mujer <br/><br/>

<input type="submit" name="enviar" value="Enviar" />
<input type="reset" name="limpiar" value="Borrar datos del formulario" />
<input type="button" name="custom" value="Acción Personalizada"/>
<input type="image" name="enviar" src="firefox.png" /><br/><br/>

Fichero adjunto <br/>
<input type="file" name="adjunto" /> <br/><br/>

<input type="hidden" name="oculto" value="cualquier_valor" />

</form>

</body>
</html>

```

La Figura 21 muestra el resultado del código anterior en la ventana del navegador.

Formularios - Mozilla Firefox 3 Beta 5

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Controles de formulario generados por input

Nombre:

Contraseña:

Hobbies

☐ Beber con los amigos


☐ Jugar basket

☐ Escuchar música

Sexo

☒ Hombre

☐ Mujer

 FIREFOX 2

Fichero adjunto

Listo

Figura 21. Vista previa de los controles creados con la etiqueta `<input>`.

Área de texto

Las áreas de texto son útiles cuando se debe introducir una gran cantidad de texto, ya que es mucho más cómodo de manejar la información que en un campo de texto normal. La etiqueta empleada para este tipo de controles es `<textarea>`.

| | |
|--------------------------------------|---|
| <code><textarea></code> | Se emplea para insertar áreas de texto en un formulario |
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● rows - Número de filas de texto que mostrará el textarea● cols - Número de caracteres que se muestran en cada fila del textarea● name - Asigna un nombre al control de texto (es imprescindible para que el servidor pueda procesar el formulario).● disabled="disabled" - Deshabilita el control y su contenido no se envía en el formulario● readonly="readonly" - Evita que el contenido del control pueda ser modificado |

La anchura del área de texto se controla mediante el atributo *cols*, que indica las columnas o número de caracteres que se podrán escribir como máximo en cada fila. La altura del área de texto se controla mediante *rows*, que indica directamente las filas de texto que serán visibles.

El principal inconveniente de éste control es que el lenguaje HTML no permite limitar el número máximo de caracteres que se pueden introducir, por lo que sólo es posible limitar el número de caracteres mediante su programación con JavaScript.

```
Descripción del producto<br/>
<textarea id="descripcion" name="descripcion" cols="40" rows="5"></textarea>
```

Descripción del producto



Figura 22. Cuadro de área de texto

Nota: Las áreas de texto son ideales cuando se necesita manejar grandes cantidades de texto

Lista desplegable

Existen tres tipos de lista desplegable: el primero (y más utilizado) está conformado por las listas que sólo muestran un valor y permiten seleccionar sólo un valor. El segundo tipo de lista sólo permite seleccionar un valor pero muestra varios a la vez y por último, el tercer tipo de lista desplegable es aquella que muestra varios valores y permite realizar selecciones múltiples.

La etiqueta `<select>` define la lista y encierra todas las opciones que muestra la lista. Cada una

de las opciones de la lista se define mediante la etiqueta `<option>`.

El atributo *value* de cada opción es obligatorio, ya que es el dato que se envía al servidor cuando el usuario envía el formulario. Para seleccionar por defecto una opción al mostrar la lista, se añade el atributo *selected* a la opción deseada.

| <select> | Se emplea para insertar listas desplegables en un formulario |
|-----------------------|--|
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● size - Número de filas que se muestra en la lista (por defecto sólo se muestra una).● multiple="multiple" - Si se incluye, permite seleccionar más de un elemento● name - Asigna un nombre al control de texto (es imprescindible para que el servidor pueda procesar el formulario).● disabled="disabled" - Deshabilita el control y su contenido no se envía en el formulario |

| <option> | Se emplea para definir cada elemento de una lista desplegable |
|-----------------------|--|
| Atributos Comunes | Básicos y eventos (Ver Elementos HTML). |
| Atributos Especiales | <ul style="list-style-type: none">● selected="selected" - Si se incluye, permite seleccionar más de un elemento● value - El valor que se envía al servidor cuando el usuario elige esa opción● disabled="disabled" - Deshabilita el control y su contenido no se envía en el formulario |

A continuación se muestra un ejemplo de los tres tipos de listas y en la Figura 23 se puede observar el resultado en la ventana del navegador.

```
Sistema operativo (Lista 1)
<select id="so" name="so">
  <option value="" selected="selected">- selecciona -</option>
  <option value="linux">GNU/Linux</option>
  <option value="mac">Mac</option>
  <option value="windows">Windows</option>
  <option value="otro">Otro</option>
</select> <br/><br/>

<label for="so2">Sistema operativo (Lista 2)</label> <br/>
<select id="so2" name="so2" size="5">
  <option value="linux" selected="selected">GNU/Linux</option>
  <option value="mac">Mac</option>
  <option value="windows">Windows</option>
  <option value="otro">Otro</option>
</select> <br/><br/>

<label for="so3">Sistema operativo (Lista 3)</label> <br/>
<select id="so3" name="so3" size="5" multiple="multiple">
```

```
<option value="linux" selected="selected">GNU/Linux</option>
<option value="mac">Mac</option>
<option value="windows">Windows</option>
<option value="otro">Otro</option>
</select>
```

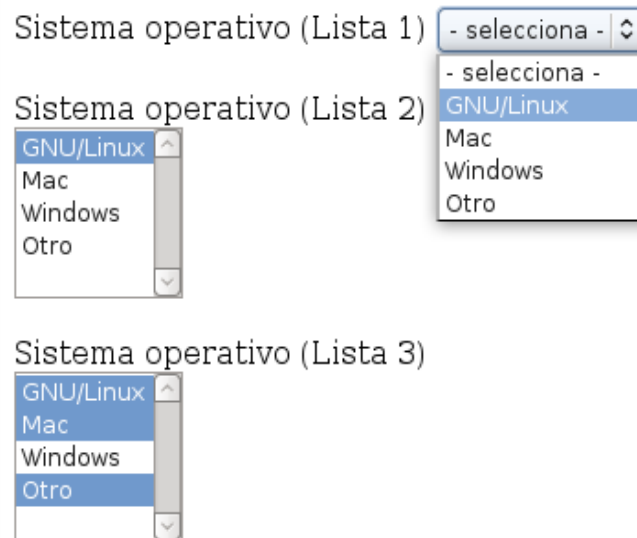


Figura 23. Distintos tipos de listas desplegables

Nota: Todos los controles HTML poseen el atributo **name** y deben tener asignado a ese atributo un nombre único que los identifique dentro del formulario. De lo contrario la información no podrá ser procesada por la aplicación.

Ejercicio 10-1. Formulario de registro

Diseñe un formulario para registrar los datos de un usuario, similar al que se muestra a continuación.



The image shows a screenshot of a web browser window titled "Ejercicio de Formularios - Mozilla Firefox 3 Beta 5". The browser's menu bar includes "Archivo", "Editar", "Ver", "Historial", "Marcadores", "Herramientas", and "Ayuda". The main content area displays a registration form with the heading "Bienvenido al sistema de registro". The form contains the following fields and controls:

- Nombre y apellido:** A single-line text input field.
- Usuario:** A single-line text input field.
- Contraseña:** A single-line text input field.
- Sexo:** A dropdown menu with the text "- selecciona -" and a small arrow icon.
- Tipo de conexión:** Three radio buttons with labels: "Cable/ADSL", "Dial-Up" (which is selected), and "No disponible".
- Avatar:** A single-line text input field followed by a button labeled "Examinar...".
- Subscription options:** Two checkboxes with labels: "Suscribirme a la lista de correos" and "Suscribirme al boletín de novedades".
- Buttons:** Two buttons at the bottom: "Enviar" and "Borrar datos del formulario".
- Status bar:** A grey bar at the very bottom of the window with the text "Listo" on the left and a small icon on the right.

Figura 24. Formulario de registro de datos

Capítulo 11. Introducción a PHP

¿Qué es PHP?

De una forma rápida y concisa podemos decir que PHP (acrónimo de **H**ypertext **P**reprocessor) es un lenguaje interpretado de **código abierto**, de alto nivel, embebido en páginas HTML y ejecutado en el servidor.

Detallando un poco más se puede decir que PHP es un lenguaje de programación tipo clásico; con variables, sentencias, condiciones, bucles y funciones, al igual que la mayoría de los lenguajes de programación; con una sintaxis derivada de lenguajes como C, Java o Perl y que soporta muchas bases de datos (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)

¿Cómo funciona PHP?

PHP se ejecuta del lado del servidor. Todo comienza cuando el navegador realiza la petición de una página web. El servidor procesa ésta solicitud, ejecuta el código PHP y la respuesta es enviada de regreso al navegador en forma de código HTML. La Figura 25 nos ilustra el proceso anterior.

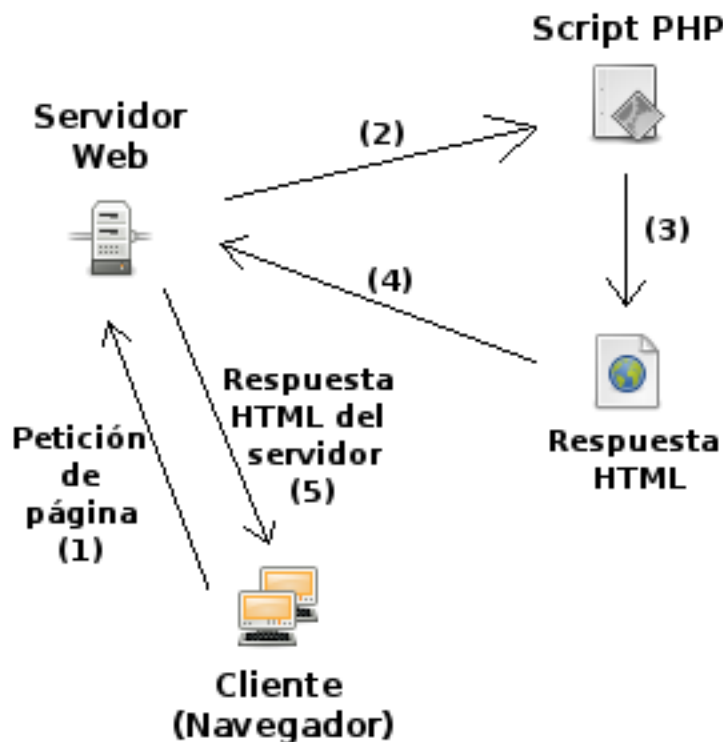


Figura 25. Funcionamiento básico de PHP

Al ser un lenguaje que se ejecuta en el servidor no es necesario que el navegador lo soporte; es independiente del navegador. Sin embargo el servidor donde están alojados los scripts debe tener instalado un interprete de PHP.

¿Qué es un archivo PHP?

Es un archivo plano que puede contener texto, etiquetas HTML y scripts de PHP. Son usados para almacenar el código fuente de los scripts PHP y tienen una extensión **.php**, .php5 o .phtml.

¿Por qué PHP?

El propósito de PHP es permitir que los desarrolladores web escriban páginas que puedan ser generadas dinámicamente de una forma segura, rápida y eficiente.

PHP puede correr en diferentes plataformas (Windows, Linux, Unix, etc.), es altamente eficiente, soporta muchos tipos de bases de datos y es compatible con casi todos los servidores web usados hoy en día (Apache, IIS, etc.). Además es fácil de aprender, es Open Source (OSS) y se puede descargar y usar sin costo alguno.

Nota: No es posible ver el código fuente PHP seleccionando la opción “Ver código fuente” en el navegador. Esto es porque el script es ejecutado en el servidor antes de enviar el resultado de vuelta al navegador. Solo es posible ver la salida del archivo PHP que es código HTML puro.

Capítulo 12. Sintaxis de PHP

Sintaxis Básica de PHP

Un bloque de código PHP siempre comienza con **<?php** y termina con **?>**. Estos bloques de código pueden estar situados en cualquier parte del documento y situados entre código HTML.

```
<?php
...
?>
```

Un archivo PHP normalmente contiene etiquetas HTML, al igual que un archivo HTML, y fragmentos de código PHP. El siguiente ejemplo muestra un código sencillo en PHP que envía el texto "Hola Mundo" al navegador:

```
<html>
<body>
<?php
    echo "Hola Mundo";
?>
</body>
</html>
```

Cada línea de código en PHP debe terminar con punto y coma (;). El punto y coma es un separador y se usa para distinguir entre una instrucción y otra.

Comentarios en PHP

En PHP se usan dos barras (slash) // para hacer comentarios de una línea o se encierra el texto entre /* y */ para hacer un comentario de múltiples líneas.

```
<html>
<body>

<?php

//Este es un comentario de una línea

/*
Este es un
comentario de
múltiples líneas
*/

?>

</body>
</html>
```

Capítulo 13. Variables en PHP

Variables

Las variables se usan para almacenar valores como números, cadenas de caracteres o resultados de funciones. Una vez que se ha inicializado la variable se puede usar una y otra vez a lo largo del script.

En PHP las variables se representan como un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

Los nombres de variables en PHP deben seguir ciertas normas para que se consideren válidos. Estas normas se pueden resumir como sigue:

- Un nombre de variable tiene que empezar con una letra o una raya baja "_" (underscore)
- Un nombre de variable solo puede contener caracteres alfanuméricos y underscores (a-z, A-Z, 0-9, y _)
- Un nombre de variable no debe contener espacios. Si el nombre de la variable es de más de una palabra entonces debería ser separado con underscore (\$my_var) o con capitalización (\$myVar)

La forma correcta de inicializar una variable en PHP es:

```
$var_name = valor;
```

Nota: Si olvida colocar el signo de dólar al inicio, la variable no será definida

Un pequeño ejemplo de cómo crear variables en PHP sería:

```
<?php  
$txt = "Hello World!";  
$number = 16;  
?>
```

Como se puede observar en el ejemplo anterior, en PHP las variables no necesitan ser declaradas antes de inicializarlas porque son declaradas automáticamente al momento de usarlas. En otros lenguajes más estrictos (como C/C++) es necesario declarar (definir) el tipo y el nombre de la variable antes de usarla.

En PHP tampoco es necesario definir el tipo de dato que almacenará variable, PHP automáticamente convierte la variable al tipo de dato correspondiente según el valor asignado.

Variables predefinidas

PHP proporciona una gran cantidad de variables predefinidas a cualquier script que se ejecute. Estas variables permiten obtener información sobre el servidor, el entorno y otros tópicos que serán discutidos más adelante. Muchas de esas variables dependen del servidor sobre el que se esté ejecutando PHP, la versión y configuración de dicho servidor, y otros factores.

A continuación una lista con una breve descripción de las variables predefinidas más utilizadas:

- **\$GLOBALS** : Contiene una referencia a cada variable disponible en el espectro de las variables del

script. Las llaves de esta matriz son los nombres de las variables globales.

- `$_SERVER` : Contiene información relacionada con el servidor o con el entorno en donde el script se esta ejecutando.
- `$_GET` : Contiene información de los datos enviados al script por medio de una petición HTTP GET.
- `$_POST` : Contiene información de los datos enviados al script por medio de una petición HTTP POST.
- `$_COOKIE` : Variables proporcionadas al script por medio de una solicitud de cookies.
- `$_FILES` : Variables que proporciona al script información sobre los archivos subidos vía HTTP.
- `$_ENV` : Contiene información sobre el entorno
- `$_REQUEST` : Variable que contiene información de los datos enviados a la aplicación a través de los métodos GET o POST.
- `$_SESSION` : Variables donde se registran todos los datos de la sesión del script.

Constantes

Las constantes en PHP tienen que ser definidas por la función `define()` y no pueden ser redefinidas con otro valor.

PHP dispone de una serie de variables predefinidas, ellas son:

- `_FILE` : Nombre del archivo que se está procesando.
- `_LINE` : Línea del archivo que se está procesando
- `_PHP_VERSION`: Versión de PHP.
- `PHP_OS`: Sistema operativo del servidor.
- `TRUE`: Verdadero.
- `FALSE`: Falso.
- `E_ERROR`: Error sin recuperación.
- `E_WARNING`: Error recuperable.
- `E_PARSE`: Error no recuperable (sintaxis).
- `E_NOTICE`: Puede tratarse de un error o no. Normalmente permite continuar la ejecución.

Todas las constantes que empiezan por "E_" se utilizan normalmente con la función `error_reporting()`.

Un ejemplo sencillo para definir una constante sería:

```
<?php
define("CONSTANTE", "Hola mundo.");
echo CONSTANTE;           // imprimirá "Hola mundo."
echo Constante;           // imprimirá "Constante" y mostrara una advertencia.
?>
```

En el ejemplo anterior se aprecia la forma de declarar una constante, además se ilustra la sensibilidad de PHP ante mayúsculas y minúsculas.

Tipos de datos

PHP soporta ocho tipos de datos primitivos

Cuatro tipos escalares:

- `boolean`
- `integer`
- `float` (número de punto-flotante, también conocido como 'double')
- `string`

Dos tipos compuestos:

- `array`

- object

Y finalmente dos tipos especiales:

- resource
- NULL

Manipulación de tipos de datos

PHP (como ya se ha mencionado) no requiere la definición explícita de tipos en la declaración de variables; el tipo de una variable es determinado por el contexto en el que la variable es usada. Lo que quiere decir que si asigna un valor de cadena a la variable \$var, \$var se convierte en una cadena. Si luego asigna un valor entero a \$var, entonces se convierte en un entero.

No obstante, PHP tiene funciones que le permiten forzar una variable para que sea evaluada como un cierto tipo; esto se conoce como **Moldeado de Tipos** (casting en inglés).

El moldeado o casting en PHP funciona de forma muy similar a como ocurre en C/C++; el nombre del tipo deseado es escrito entre paréntesis antes de la variable que debe ser moldeada. Por ejemplo, suponga que tiene una variable \$num con la cadena de texto "100" y quiere sumar este valor con el entero 10. Basta con moldear la variable \$num a entero y realizar la suma.

```
<?php
$num = "100";      // $num es un string
$var = (int) $num; // $bar es un entero

echo $var+10;      // Mostrará 110 en pantalla
?>
```

Nota: PHP realiza moldeado automático para ciertos tipos de variables. En ejemplo anterior era suficiente con realizar la suma entre la cadena de caracteres y el entero; PHP automáticamente hará la conversión y luego la operación matemática.

Los moldeados permitidos son:

- (int), (integer) - moldeamiento a entero
- (bool), (boolean) - moldeamiento a booleano
- (float), (double), (real) - moldeamiento a flotante
- (string) - moldeamiento a cadena
- (binary) - moldeamiento a cadena binaria (PHP 6)
- (array) - moldeamiento a matriz
- (object) - moldeamiento a objeto

Capítulo 14. Echo

En capítulos anteriores se usó la función *echo()* sin saber de qué se trataba. Esta función se usa más que cualquier otra función de PHP, así que vamos a darle una explicación sólida y completa.

Las funciones **echo()** y **print()** se emplean para imprimir una salida de texto en el navegador. Aunque las dos producen el mismo resultado la más común es *echo()*; por lo tanto enfocaremos la explicación en ella.

La función echo puede usarse con variables o directamente con texto encerrado entre comillas. Un ejemplo de cómo usar la función sería:

Código PHP:

```
<html>
<head>
<title>Mi primer script en PHP</title>
</head>
<body>

<?php
    $myString = "Hola Mundo!";
    echo "<h1>Primer script en PHP</h1>";
    echo $myString;
    echo "<h5>PHP es muy fácil de usar</h5>";
?>

</body>
</html>
```

Salida HTML:

```
<html>
<head>
<title>Mi primer script en PHP</title>
</head>
<body>

<h1>Primer script en PHP</h1>
Hola Mundo!
<h5>PHP es muy fácil de usar</h5>

</body>
</html>
```

En el ejemplo anterior se cometió un error intencionalmente. La cadena "Hola Mundo!" se imprimió sin ninguna información HTML.

El texto que se está imprimiendo es enviado al usuario en forma de una página web, es decir como código HTML, por lo que es importante usar la sintaxis HTML apropiada.

En la primera y tercera instrucción de *echo()* escribimos etiquetas válidas de titulares de sección HTML. Esto se hace simplemente colocando las etiquetas de inicio y de cierre al inicio y al final de la cadena respectivamente. El hecho de estar usando PHP para crear páginas web no quiere decir que debamos olvidarnos de las reglas y la sintaxis HTML.

Nota: El texto que se imprime con la función `echo()` es enviado al navegador web en forma de código HTML por lo que debe respetar la sintaxis y la codificación HTML.

Echo con comillas

Poder generar salidas HTML usando PHP es maravilloso. Sin embargo hay que tener cuidado cuando se usa código HTML o cualquier otra cadena de texto que contenga comillas. La función `echo()` usa comillas para definir el inicio y el fin de la cadena a imprimir, por lo tanto es necesario emplear una de las siguientes medidas para imprimir texto que contenga comillas:

- Escapar las comillas presentes en la cadena de texto usando una barra invertida (backslash). Sólo basta con colocar la barra invertida justo antes de la comilla, por ejemplo: `\`
- Reemplazar las comillas dentro del texto por apóstrofes.

En el siguiente ejemplo se muestran las formas correctas e incorrectas de usar la función `echo` con comillas.

```
<?php
/* Incorrecto: Porque las comillas dobles le hacen pensar al interprete de PHP que la
cadena a imprimir es "<h5 class=" */
echo "<h5 class="specialH5">I love using PHP!</h5>";

/* Correcto: Las comillas fueron escapadas */
echo "<h5 class=\"specialH5\">I love using PHP!</h5>";

/* Correcto: Se reemplazaron las comillas dobles por apóstrofes dentro del texto*/
echo "<h5 class='specialH5'>I love using PHP!</h5>";
?>
```

Echo con variables

Usar `echo()` con variables es muy sencillo. No hacen falta comillas, ni siquiera si la variable contiene un tipo de datos que no sea string. El ejemplo siguiente muestra la forma de imprimir variables con `echo`.

Código PHP:

```
<?php
$my_string = "Hola Pedro. Mi nombre es: ";
$my_number = 4;
$my_letter = a;
echo $my_string;
echo $my_number;
echo $my_letter;
?>
```

Salida:

```
Hola Pedro. Mi nombre es: 4a
```

Echo con variables y cadenas de caracteres

También se pueden combinar las cadenas de caracteres y las variables al imprimir con `echo()`. Con esta práctica se pueden ahorrar un gran número de sentencias `echo()`. Las variables y las cadenas de caracteres se concatenan usando un punto (`.`).

El ejemplo siguiente muestra como hacer tal combinación.

Código PHP:

```
<?php
$my_string = "Hola Pedro. Mi nombre es: ";
$newline = "<br/>";

echo $my_string."María".$newline;
echo "Hola, yo soy Pedro. Quién eres tú? ".$newline;
echo "María.".$my_string."María";
?>
```

Salida:

```
Hola Pedro. Mi nombre es: María
Hola, yo soy Pedro. Quién eres tú?
María. Mi nombre es: María
```

Esta combinación se puede hacer múltiples veces, tal y como muestra el ejemplo. El método de concatenación será discutido más adelante.

Capítulo 15. Cadenas de caracteres

En este capítulo trataremos las cadenas de caracteres en PHP y las funciones más comunes para manipularlas.

Una cadena de texto puede crearse usando comillas dobles o comillas simples. En ambos casos, si se desea incluir un caracter especial entonces debe escaparse usando una barra invertida (backslash). El código que se muestra a continuación ilustra esta situación.

Código PHP:

```
<?php
    $var1="Hola Mundo con comillas \"dobles\"";
    $var2='Hola Mundo con comillas \'simples\'';
    echo $var1;
    echo $var2;
?>
```

Salida:

```
Hola Mundo con comillas "dobles"
Hola Mundo con comillas 'simples'
```

Nota: Se recomienda usar comillas dobles para crear cadenas de caracteres, pues permiten usar muchos caracteres especiales (sin necesidad de escaparlos) que no pueden ser usados con comillas simples.

Como se ha visto hasta ahora, las variables de tipo string almacenan un conjunto de caracteres que luego de ser creados, pueden ser manipulados. Ahora estudiaremos tres de funciones más comunes para manejar cadenas de texto.

strlen()

La función strlen() se usa para calcular la longitud (en caracteres) de la cadena de texto.

```
int strlen ( string $cadena )
```

Devuelve un entero con la longitud de *\$cadena*.

Ejemplo de strlen.

```
<?php
$cadena = 'abcdef';
echo strlen($cadena); // 6

$cadena = ' ab cd ';
echo strlen($cadena); // 7
?>
```

strpos()

Localiza la primera aparición de un caracter o de un fragmento de texto dentro de la cadena.

```
int strpos ( string $cadena, string $character )
```

Devuelve la posición numérica de la primera aparición de *\$character* en la cadena *\$cadena*. Esta función puede tomar una cadena completa como caracter y se utilizará en su totalidad.

Si no se encuentra *\$character* , devuelve **FALSE**.

Ejemplo de strpos.

```
<?php
echo strpos("Hello world!","world");    // Devuelve 6
echo strpos("Hello world!","abc");      // Devuelve FALSE
?>
```

La primera llamada a la función devuelve 6 y no 7 porque la primera posición de la cadena de texto es 0 y no 1.

Nota: Las cadenas de texto comienzan a contarse desde la posición cero

strcmp()

Compara dos cadenas de texto en modo binario . Esta comparación es sensible a mayúsculas y minúsculas.

```
int strcmp ( string $cadena1, string $cadena2 )
```

Devuelve:

- Un entero < 0 si *\$cadena1* es menor que *\$cadena2*
- Un entero > 0 si *\$cadena1* es mayor que *\$cadena2*
- Cero (0) si son iguales.

Ejemplo de strpos.

```
<?php
echo strcmp("xyz","abc");                // Devuelve 1
echo strcmp(" hola","abc");              // Devuelve -1
echo strcmp("abc","abc");                // Devuelve 0
?>
```

substr()

Se emplea para obtener parte de una cadena.

```
string substr ( string $cadena, int $comienzo [, int $longitud] )
```

Devuelve la porción de *\$cadena* desde *\$comienzo* hasta el final de la cadena.

Si *\$comienzo* es positivo ó 0, la cadena devuelta comenzará en esa posición de *\$cadena*. Si *\$comienzo* es negativo, la cadena devuelta comenzará en dicha posición pero contando desde el final de *\$cadena* hacia el principio. Si *\$comienzo* es mayor que *\$cadena* se devuelve una cadena vacía.

Si se especifica *\$longitud* y es positiva, la cadena devuelta tendrá como máximo *\$longitud* de caracteres a partir de *\$comienzo*. Si *\$longitud* es negativa, se omitirán *\$longitud* caracteres desde el final de la cadena.

Ejemplo de substr.

```
<?php
echo substr("abcdef", 1);           // Devuelve "bcdef"
echo substr("abcdef", 1, 3);        // Devuelve "bcd"
echo substr("abcdef", -1);          // Devuelve "f"
echo substr("abcdef", -3, 1);        // Devuelve "d"
echo substr("abcdef", 2, -1);        // Devuelve "cde"
echo substr("abcdef", -3, -1);       // Devuelve "de"
?>
```

str_replace()

Sustituye todas las apariciones de una cadena en otra.

```
string str_replace ( string $cad_buscada, string $cad_sustituta, string $cad_original
[,int $veces] )
```

Esta función sustituye en *\$cad_original* todas las apariciones de *\$cad_buscada* por *\$cad_sustituta* y devuelve la cadena resultante.

Ejemplo de str_replace.

```
<?php
echo str_replace("o","", "Hola Mundo de PHP"); // Devuelve "Hla Mund de PHP"
echo str_replace("Mundo", "Maria", "Hola Mundo"); // Devuelve "Hola Maria"
?>
```

Ejercicio 15-1. Palabras

A partir de una frase almacenada en una variable, escriba el código en PHP que permita imprimir cada palabra en una línea.

Pistas:

- Las palabras están separadas por espacios en blanco

Ejercicio 15-2. New slang

Escriba un script que tome una cadena de texto y reemplace todas las apariciones de los siguientes caracteres:

- La letra A por el número 4
- La letra E por el número 3
- La letra S por el número 5

Ejercicio 15-3. Detectar palabras

Realice un script que permita detectar la presencia de la palabra “color” en una frase y pruebe con varias cadenas (que contengan o no la palabra “color”) para verificar que funciona correctamente.

Capítulo 16. Operadores en PHP

Un operador es algo a lo que se le entrega uno o más valores y produce otro valor. En PHP existen tres tipos de operadores. En primer lugar se encuentra el operador unario, el cual opera sobre un único valor, por ejemplo ! (el operador de negación Not) o ++ (el operador de incremento). El segundo grupo se conoce como operadores binarios; este grupo contiene la mayoría de operadores que soporta PHP y el tercer grupo está formado por el operador ternario.

Debido al alcance de este curso solo estudiaremos los operadores unarios y binarios. El operador ternario se deja para un módulo más avanzado o como tarea para el lector.

Operadores aritméticos

| Operador | Descripción | Ejemplo | Resultado |
|----------|----------------|----------------|-----------|
| + | Adición | 2+2 | 4 |
| - | Sustracción | 5-3 | 2 |
| * | Multiplicación | 5*5 | 25 |
| / | División | 5/2 | 2.5 |
| % | Módulo (resto) | 10%8 | 2 |
| ++ | Incremento | \$x=5 \$x++ | 6 |
| -- | Decremento | \$x=5 \$x-- | 4 |

Operadores de asignación

| Operador | Descripción | Ejemplo | Es igual a |
|----------|--------------------------|------------|-----------------|
| = | Asignación | \$x = \$y | \$x = \$y |
| += | Suma abreviada | \$x += \$y | \$x = \$x + \$y |
| -= | Resta abreviada | \$x -= \$y | \$x = \$x - \$y |
| *= | Multiplicación abreviada | \$x *= \$y | \$x = \$x * \$y |
| /= | División abreviada | \$x /= \$y | \$x = \$x / \$y |
| .= | Concatenación abreviada | \$x .= \$y | \$x = \$x . \$y |
| %= | Módulo (resto) abreviado | \$x %= \$y | \$x = \$x % \$y |

Operadores de comparación

| Operador | Descripción | Ejemplo | Devuelve |
|----------|-------------------|---------|----------|
| == | Igual | 5==8 | false |
| != | Diferente | 5!=8 | true |
| > | Mayor que | 5 > 8 | false |
| < | Menor que | 5 < 8 | true |
| >= | Mayor o igual que | 5 >= 8 | false |
| <= | Menor o igual que | 5 <= 8 | true |

Operadores lógicos

| Operador | Descripción | Ejemplo | Devuelve |
|----------|-------------|--|----------|
| && | And | \$x=6 \$y=3 (\$x > 10 && \$y > 1) | false |
| | Or | \$x=6 \$y=3 (\$x == 5 \$y == 3) | true |
| ! | Not | \$x=6 \$y=3 !(\$x == \$y) | true |

Operador de cadenas de caracteres

| Operador | Descripción | Ejemplo | Devuelve |
|----------|---|--|--------------|
| . | Concatena una o más cadenas de caracteres | \$x = "Hola" \$y = "mundo" \$x . " " . \$y | "Hola mundo" |

Capítulo 17. If...else...elseif

Piense en las decisiones que tomamos antes de dormir. **Si** tenemos algo que hacer al día siguiente (ir a la universidad, al trabajo o a una reunión) **entonces** ponemos la alarma del despertador, **sino** idormimos tanto como podemos!

Cuando programamos, a menudo tenemos la necesidad de ejecutar diferentes acciones basadas en diferentes condiciones. Es común querer ejecutar una sentencia si se cumple cierta condición y otra diferente en caso de que no se cumpla.

Para esto se usa un grupo de sentencias condicionales; y las sentencia *if*, *else* y *elseif* forman parte de ese grupo.

Sentencia if

Imagine que todos los viernes usted desea mostrar en su página web un mensaje que diga “Feliz fin de semana”. La sentencia **if** es la indicada para esa labor.

Al igual que en otros lenguajes de programación, en PHP, la sentencia *if* evalúa una condición; si el resultado es TRUE se ejecuta una porción de código.

Sintaxis:

```
if (condicion)
    //instrucciones que se ejecutarán si condicion es TRUE;
```

Nota: si en un bloque condicional se va a ejecutar más de una instrucción, éstas deben ir encerradas entre llaves “{ }” para denotar el inicio y el fin del bloque (esto aplica para todos los bloques de código en PHP).

Una sentencia if verdadera

Código PHP:

```
<?php
$dia="Viernes";

echo "Gracias por visitar mi página.<br/>";

if ($dia == "Viernes") {
    echo "Feliz fin de semana";
}

?>
```

Salida:

```
Gracias por visitar mi página.
Feliz fin de semana
```


Una sentencia if falsa

Código PHP:

```
<?php
$dia="Lunes";

echo "Gracias por visitar mi página<br/>";

if ($dia == "Viernes") {
    echo "Feliz fin de semana";
}

?>
```

Salida:

Gracias por visitar mi página

Sentencia if...else

Ya estudiamos como evaluar una condición usando la sentencia *if*. Pero ¿qué pasa si la condición no se cumple y aún así queremos realizar una acción? Por ejemplo, suponga que en su página web quiere mostrar el mensaje de despedida los días viernes pero además quiere mostrar otro mensaje para los demás días de la semana. Esto se hace con la ayuda de **else**

La combinación if...else permite evaluar una condición; si ésta se cumple se ejecuta un conjunto de instrucciones, sino se ejecuta otro.

Sintaxis:

```
if (condicion)
    //instrucciones que se ejecutarán si la condicion es TRUE;
else
    //instrucciones que se ejecutarán si la condicion es FALSE;
```

Una sentencia if...else verdadera

Código PHP:

```
<?php
$dia="Viernes";

echo "Gracias por visitar mi página.<br/>";

if ($dia == "Viernes") {
    echo "Feliz fin de semana";
} else{
    echo "Que tenga feliz día";
}
```

```
}  
?>
```

Salida:

```
Gracias por visitar mi página.  
Feliz fin de semana
```

Una sentencia if...else falsa

Código PHP:

```
<?php  
$dia="Lunes";  
  
echo "Gracias por visitar mi página.<br/>";  
  
if ($dia == "Viernes") {  
    echo "Feliz fin de semana";  
} else{  
    echo "Que tenga feliz día";  
}  
  
?>
```

Salida:

```
Gracias por visitar mi página.  
Que tenga feliz día
```

Sentencia elseif

La combinación if...else es excelente para evaluar una condición pero, ¿que pasa si queremos evaluar más de una condición a la vez? Imagine que ahora desea mostrar en su página web un mensaje de bienvenida para cada día de la semana. Para eso existe **elseif**.

elseif permite evaluar un grupo de condiciones y ejecutar instrucciones específicas para cada caso.

Nota: La sentencia **elseif** solo puede usarse si previamente se ha empleado la sentencia **if**

Sintaxis:

```
if (condicion_1)
    //instrucciones que se ejecutan si condicion_1 es TRUE;
elseif (condición_2)
    //instrucciones que se ejecutan si condicion_2 es TRUE;
...
elseif (condición_n)
    //instrucciones que se ejecutan si condicion_n es TRUE;
else
    //instrucciones que se ejecutan si toda las condiciones anteriores son FALSE;
```

Nota: La sentencia else se ejecuta solamente cuando la expresión if y todas las expresiones elseif (que pudieran haber) se evalúan como FALSE.

Una sentencia elseif

Código PHP:

```
<?php
$dia="Viernes";

echo "Gracias por visitar mi página.<br/>";

if ($dia == "Lunes") {
    echo "Hoy es lunes";
}elseif ($dia == "Martes") {
    echo "Hoy es martes";
}elseif ($dia == "Miércoles") {
    echo "Hoy es miércoles";
}elseif ($dia == "Jueves") {
    echo "Hoy es jueves";
}elseif ($dia == "Viernes") {
    echo "Hoy es viernes";
} else{
    echo "Ya es iFin de semana!";
}

?>
```

Salida:

```
Gracias por visitar mi página.
Hoy es viernes
```

Ejercicio 17-1. Jugando con strings y condicionales

Desarrolle una página web que imprima un mensaje de bienvenida diferente para cada día de la semana y que además le de al usuario los buenos días, las buenas tardes o las buenas noches.

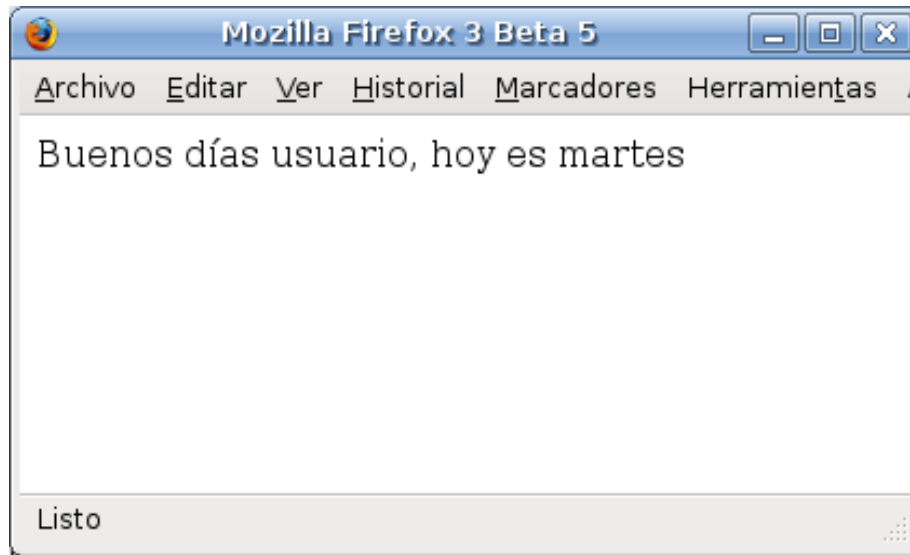


Figura 26. Ejercicio de strings y condicionales

Pistas:

1. La función `date("w")` devuelve una cadena de caracteres con el día actual. ("0" para el domingo y "6" para el sábado)
2. La función `date("H")` devuelve una cadena de caracteres con la hora actual en formato 24H (sin tomar en cuenta los minutos). Por ejemplo: retorna "13" para el período comprendido entre la 1:00pm y la 1:59pm
3. Para una referencia completa de la función `date` visite <http://php.net/date>

Ejercicio 17-2. Edades

Sabiendo la edad de una persona, escriba un script en PHP que indique si la persona es menor de edad, mayor de edad o jubilado (suponiendo que la mayoría de edad se cumple a los 18 años y la jubilación es a los 55 años). Pruebe con diferentes valores para la edad.

Ejercicio 17-3. ¿Mayor o menor?

Escriba un script en PHP que a partir de dos números indique cuál de ellos es el mayor y cuál el menor. Pruebe con diferentes números.

Ejercicio 17-4. Calificaciones

Ahora debe escribir el código para convertir las calificaciones del 1 al 20 en letras de la A a la E.

Capítulo 18. Switch...case

En el capítulo anterior estudiamos cómo realizar diferentes acciones tomando en cuenta diferentes condiciones. Sin embargo, algunas veces la sentencia *if* no es la forma más eficiente de realizar esta labor.

Por ejemplo, imagine que disponemos de 30 posibles localidades para viajar y tenemos una variable que almacena a cuál de esos sitios se viajará. Tomar la decisión usando bloques de *if...elseif...else* no sería lo más eficiente ni lo más simple; el programa tendría que evaluar una por una todas las condiciones hasta encontrar la que se cumple. ¿Y si las localidades cambian a 100? Sería una locura implementarlo de esa manera, tanto por el rendimiento de la aplicación como por la complejidad del código.

Usando la instrucción **switch** podemos evaluar todas esas condiciones de una sola vez; es, de hecho, la forma más eficiente de llevar a cabo esta tarea.

La sentencia *switch* toma una variable como entrada y luego la evalúa contra todos los diferentes casos que se hayan definido. Así, en lugar de tener que evaluar cada condición una por una, se evalúa la condición una sola vez.

Sintaxis:

```
switch ($expresion)
{
case val1:
    // instrucciones que se ejecutan si $expresion == val1;
    break;
case val2:
    // instrucciones que se ejecutan si $expresion == val2;
    break;
case valn:
    // instrucciones que se ejecutan si $expresion == valn;
    break;
...
default:
    // instrucciones que se ejecutan si $expresion es
    // distinto de todos los valores listados arriba
}
```

Donde *val1*, *val2*, ... , *valn* representan los posibles valores que puede tomar la variable *\$expresion*. Si el valor de *\$expresion* no satisface ninguno de los casos listados entonces se ejecuta el código de la sección *default*.

Nota: La sentencia *switch* se usa para evitar largos e ineficientes bloques de instrucciones *if..elseif..else*.

Código PHP:

```
<?php
$x=0;

switch ($x) {
case 1:
    echo "Número 1";
```

```

        break;
    case 2:
        echo "Número 2";
        break;
    case 3:
        echo "Número 3";
        break;
    default:
        echo "Número que no está entre 1 y 3";
}
?>

```

Salida:

```
Número que no está entre 1 y 3
```

Es importante entender cómo se ejecuta la sentencia *switch* para evitar errores lógicos en la aplicación. Cuando PHP encuentra una sentencia *case* con un valor que coincide con el valor de la expresión comienza a ejecutar las instrucciones de ese bloque hasta el final del bloque *switch* o hasta que consiga una sentencia *break*.

La sentencia **break** interrumpe la ejecución de *switch* y es necesario colocarla al final de cada bloque *case* (excepto en el último), porque de lo contrario las instrucciones del siguiente bloque también serán ejecutadas.

El ejemplo siguiente demuestra el comportamiento de las sentencias *switch* y *break*.

```

<?php
$i=0;

switch ($i) {
    case 0:
        echo "i es igual a 0";
    case 1:
        echo "i es igual a 1";
    case 2:
        echo "i es igual a 2";
}
?>

```

Salida:

```
i es igual a 0
i es igual a 1
i es igual a 2
```

No es el resultado que esperábamos ver ¿o sí? Es lógico pensar que la ejecución del *switch* se detendría luego de la primera instrucción, pero como se mencionó anteriormente, PHP seguirá ejecutando el código hasta el final del *switch* a no ser que se encuentre con un *break*.

La forma correcta de escribir el ejemplo anterior sería:

```
<?php
$i=0;
switch ($i) {
    case 0:
        echo "i es igual a 0";
        break;
    case 1:
        echo "i es igual a 1";
        break;
    case 2:
        echo "i es igual a 2";
    }
?>
```

Salida:

```
i es igual a 0
```

Algunas veces, podemos aprovechar éste comportamiento del *switch* para escribir nuestro código de forma más simple y elegante. El siguiente ejemplo ilustra esa situación.

```
<?php
$i=1;
switch ($i) {
    case 0:
    case 1:
    case 2:
        echo "i es menor que 3";
        break;
    case 3:
        echo "i es 3";
        break;
    default:
        echo "i es mayor que 3 o negativo";
    }
?>
```

Salida:

```
i es menor que 3
```

Ejercicio 18-1. Switch en lugar de if...elseif

Reescriba el código del ejercicio 17-1 pero esta vez usando la sentencia switch.

Ejercicio 18-2. Calificaciones II

En un instituto las calificaciones son dadas con letras; A es la más alta y F la más baja. Escriba un script que permita mostrar un mensaje al estudiante según su calificación

Ejercicio 18-3. Rangos de temperatura.

Conociendo el valor de una temperatura (entre 0 y 10 °C), desarrolle una página web en php que permita mostrar un mensaje al usuario para cada rango de temperatura. Los rangos están separados entre sí 2 °C

Capítulo 19. Bucles

Durante la programación de una página web también es necesario ejecutar una porción de código muchas veces para completar una tarea; por ejemplo, enviar 20 mensajes de correo, borrar 10 archivos, imprimir la información de 50 usuarios, etc. Sería realmente tedioso tener que escribir 50 veces la misma línea de código sólo para realizar una tarea trivial como imprimir información. Los bucles son nuestros aliados en estos casos.

Un bucle es una estructura de control que nos permite ejecutar código PHP repetidamente durante un número finito de iteraciones o hasta que cierta condición se cumple. En este capítulo estudiaremos tres tipos básicos de bucles: *while*, *do...while* y *for*.

While

La sentencia **while** le dice a PHP que ejecute un conjunto de instrucciones repetidamente **mientras** la expresión indicada se evalúe como TRUE. El valor de la expresión es comprobado al inicio de cada iteración.

Sintaxis:

```
while (condicion)
    // instrucciones a ejecutar
```

Al igual que en la sentencia *if*, si en el *while* se va a ejecutar más de una instrucción entonces éstas deben ir encerradas entre llaves para indicar el principio y el fin del bloque.

Nota: Es necesario comprobar que el valor de la expresión cambia durante la ejecución del bucle; de lo contrario éste se repetirá infinitas veces (bucle infinito) haciendo que la aplicación se estanque en esa sección del código para siempre.

Ejemplo:

```
<?php
    $i=1;

    while($i<=5) {
        echo "El número es ".$i."<br/>";
        $i++;    // Aquí se modifica el valor de la expresión
    }
?>
```

Salida:

```
El número es 1
El número es 2
El número es 3
El número es 4
El número es 5
```

Do...while

Los bucles **do...while** son similares a los bucles *while*, excepto que las condiciones se comprueban al final de cada iteración y no al principio.

Sintaxis:

```
do {  
    // instrucciones a ejecutar  
}  
while (condicion);
```

La principal diferencia del *do...while* es que garantiza la ejecución del código al menos una vez (la condición se comprueba al final de la iteración luego de ejecutar el código), mientras que con el *while* puede que el código nunca se ejecute porque la condición se comprueba al principio de cada iteración (justo antes de ejecutar el código).

Nota: Los bucles *do...while* garantizan la ejecución del código al menos una vez, mientras que con los *while* el código puede que nunca se ejecute.

Ejemplo con while()

```
<?php  
  
$i=1;  
  
while($i < 0) {  
    echo "Este texto nunca se mostrará<br/>";  
    $i++;  
}  
  
echo "Salida del bucle";  
?>
```

Salida:

```
Salida del bucle
```

Ejemplo con do...while()

```
<?php  
  
$i=1;  
  
do{  
    echo "Este texto si que se mostrará<br/>";  
    $i++;  
} while($i < 0)  
  
echo "Salida del bucle";  
?>
```

Salida:

```
Este texto si que se mostrará  
Salida del bucle
```

For

La sentencia **for** también ejecuta un código repetidas veces, pero se utiliza principalmente cuando se conoce el número exacto de iteraciones que se desean realizar.

Esta sentencia utiliza tres parámetros. El primer parámetro inicializa la variable a evaluar, el segundo almacena la condición que se debe cumplir para que se ejecute el bucle (TRUE o FALSE) y el tercero representa el incremento que se aplicará a la variable en cada iteración del bucle.

Sintaxis:

```
for (inicialización; condición; incremento){  
    // instrucciones a ejecutar  
}
```

Ejemplo:

```
<?php  
    for($i=0; $i <= 5; $i++) {  
        echo "El número es ".$i."<br/>";  
    }  
?>
```

Salida:

```
El número es 0  
El número es 1  
El número es 2  
El número es 3  
El número es 4  
El número es 5
```

Nota: Con la sentencia *for* no es necesario que se modifique el valor de la variable de control dentro del bucle porque la instrucción lo hace automáticamente al final de cada iteración.

Ejercicio 19-1. Pirámide

Escriba un script en PHP que muestre en la ventana del navegador lo siguiente:

```
*
**
***
****
*****
*****
```

Ejercicio 19-2. Primos

Realice un script en PHP que imprima los números primos que hay entre 2 y 50

Ejercicio 19-3. Temperaturas.

Escriba una página web en PHP que muestre una tabla de dos columnas para la conversión de temperaturas Fahrenheit a Celsius. El rango debe ir de 0 °F a 300 °F en incrementos de 20°F.

Pistas:

- La ecuación para transformar de Fahrenheit a Celsius es: $^{\circ}\text{C} = \frac{5 * (^{\circ}\text{F} - 32)}{9}$

Ejercicio 19-4. Los 100 primeros

Escriba el código PHP para mostrar en la ventana del navegador la suma de los 100 primeros números enteros.

Ejercicio 19-5. Pirámide II

Escriba un script en PHP que a partir de un número n permita construir una pirámide como la que se muestra a continuación.

```
1
12
123
1234
...
1234 ... n
```

Capítulo 20. Funciones

Creando una función

Una función es sólo el nombre que le damos a un bloque de código que podemos ejecutar cuando y donde necesitemos. A simple vista no parece importante, pero el uso de funciones nos da la capacidad de agrupar varias instrucciones bajo un solo nombre y poder llamarlas varias veces desde diferentes puntos del programa, ahorrándonos la necesidad de escribirlas de nuevo.

Para crear una función en PHP es necesario cumplir estas simples reglas:

- Todas las funciones deben comenzar con la palabra "function"
- El nombre de la función solo puede comenzar con una letra (a-z A-Z) o un underscore
- El inicio del bloque de instrucciones debe estar marcado por una llave "{"
- El fin del bloque de instrucciones debe estar marcado por una llave "}"

Sintaxis:

```
function nombre_de_la_funcion(){  
    // instrucciones a ejecutar  
}
```

Ejemplo:

```
<?php  
  
function miPrimeraFuncion() {  
    echo "Hemos definido nuestra primera función en PHP.<br/>";  
    echo "Dentro de la función podemos escribir código PHP tal<br/>";  
    echo "y como lo veníamos haciendo en los capítulos anteriores<br/>";  
}  
  
?>
```

Usando una función

Para usar una función en PHP basta con escribir su nombre en la parte del programa donde queremos ejecutarla y listo. Recuerde colocar los paréntesis luego del nombre de la función, de otra manera PHP no sabrá que está intentando llamarla.

Ejemplo:

```
<?php  
  
function miPrimeraFuncion() {  
    echo "Hemos definido nuestra primera función en PHP.<br/>";  
    echo "Dentro de la función podemos escribir código PHP tal<br/>";  
    echo "y como lo veníamos haciendo en los capítulos anteriores.<br/>";  
}  
  
echo "Hola Mundo<br/>";
```

```
miPrimeraFuncion();  
echo "Podemos concluir que la función ha sido todo un iéxito!";  
  
?>
```

Salida:

```
Hola Mundo  
Hemos definido nuestra primera función en PHP.  
Dentro de la función podemos escribir código PHP tal  
y como lo veníamos haciendo en los capítulos anteriores.  
Podemos concluir que la función ha sido todo un iéxito!
```

En la salida anterior se resalta la porción de texto que es generado por la función. Ese texto es interpretado por el navegador como HTML.

Funciones y parámetros de entrada

La función que acabamos de crear es realmente simple, solo imprime cadenas de texto invariables en pantalla. Sin embargo podemos hacerla mucho más versátil pasándole parámetros. Los parámetros se le pasan a las funciones especificándolos dentro de los paréntesis y se trataran como variable locales dentro del bloque de código. Son muy útiles cuando la función debe conocer el valor de una variable o algún dato específico para realizar alguna acción.

Sintaxis:

```
function nombre_de_la_funcion($param_1, $param_2, ..., $param_n)  
    // instrucciones a ejecutar  
}
```

Vamos a crear una función que reciba dos parámetros: el nombre del usuario y la edad, y a continuación imprima un saludo.

```
<?php  
  
function saludo($nombre, $edad){  
    echo $nombre. "Tengo ". $edad . " años<br/>";  
}  
  
echo "Mi nombre es ";  
saludo("Pedro", "23");  
  
echo "Mi nombre es ";  
saludo("Maria", "14");  
  
echo "Mi nombre es ";  
saludo("Julia", "25");  
?>
```

Salida:

```
Mi nombre es Pedro. Tengo 23 años  
Mi nombre es Maria. Tengo 14 años
```

```
Mi nombre es Julia. Tengo 25 años
```

Note que con la función creada en el ejemplo anterior no es necesario escribir el código para el saludo una y otra vez para cada persona. Solo basta con llamar a la función *saludo()* con el nombre y la edad de cada persona y ella se encarga del resto.

Nota: Usando parámetros creamos una función mucho más poderosa, pero también mucho más delicada. Si la función no recibe los parámetros que se indican entonces generará un error y el script no continuará con su ejecución.

Funciones y valores de retorno

Además de pasarle información a una función (parámetros), también podemos obtener información de ella. Esto es útil cuando la función, por ejemplo, realiza alguna tarea y por alguna razón necesitamos conocer su resultado. Hay que aclarar que la función solo puede retornar una cosa; bien sea un entero, un arreglo, un string o lo que queramos, pero solo uno.

Para retornar un valor usamos la instrucción **return** al final del bloque de código de la función y para capturar este valor en el programa debemos asignarle una variable a la función para que, al terminar, deposite el valor retornado en dicha variable. Luego de ejecutar la sentencia return el programa saldrá de la función y continuará con su ejecución.

Sintaxis:

```
function nombre_de_la_funcion($param_1, $param_2, ..., $param_n)
    // instrucciones a ejecutar
    return valor_deseado;
}
```

Ahora crearemos una función que realice la suma de dos números y nos devuelva el resultado de dicha suma.

```
<?php

function suma($numX, $numY){
    $total = $numX + $numY;
    return $total;
}

$numero = 0;
echo "Antes de la función, numero = ". $numero . "<br/>";

$numero = suma(3, 4);    // Almacena el valor de retorno de suma en $numero

echo "Después de la función, numero = " . $numero . "<br/>";
?>
```

Salida:

```
Antes de la función, numero = 0
Después de la función, numero = 7
```

Ejercicio 20-1. Geometría básica

Desarrolle un script en PHP con funciones que permitan calcular el perímetro y el área de un círculo con un radio conocido.

Pistas:

- El perímetro de un círculo viene dado por $2 * PI * radio$ y el área por $PI * radio^2$
- Una función para cada cálculo
- La función de PHP `pi()`, devuelve el valor del número PI

Ejercicio 20-2. Factorial

Realice una función en PHP que permita calcular el factorial de un número cualquiera y compruebe con varios números.

Pistas:

- La ecuación para el factorial de un número n es: $n * (n - 1) * (n - 2) * ... * 2 * 1$

Capítulo 21. Manejo de formularios

Hasta ahora hemos estudiado las características y funciones básicas de PHP, pero ya es hora de aplicar esos conocimientos en el mundo real. Una de las aplicaciones más comunes es tener un formulario HTML que obtiene información de los usuarios que visitan la página y la envía a un script PHP que se encarga de procesarla y generar un resultado. Eso precisamente es lo que vamos a estudiar ahora.

Primero comencemos estudiando las variables de PHP que nos permiten obtener los datos de un formulario. Esas variables son `$_GET` y `$_POST`.

La variable `$_GET`

Esta variable se usa para recolectar los valores de un formulario que ha enviado los datos usando el método GET. Consiste de un arreglo de nombres y valores; cada valor del arreglo puede accederse usando como índice el valor del atributo *name* del elemento HTML que queremos obtener.

La información enviada usando GET estará visible para todos los usuarios (se muestra en la barra de direcciones del navegador) y no puede exceder los 100 caracteres de longitud.

Supongamos que tenemos un formulario como el siguiente:

Código HTML

```
<form action="prueba.php" method="get">

Nombre: <input type="text" name="nombre" />
Edad: <input type="text" name="edad" />
<input type="submit" />

</form>
```

Para obtener los valores de los campos *nombre* y *edad* del formulario, basta con colocar el siguiente código en nuestro script y estaremos almacenando dichos valores en las variables *\$nombre* y *\$edad*.

Código PHP

```
<?php
$nombre = $_GET['nombre'];
$edad = $_GET['edad'];
?>
```

Nota: Al usar GET todos los nombres y valores de las variables se muestran en la URL de la página, por lo que no es conveniente usar éste método cuando se trabaja con contraseñas o información sensible

La variable `$_POST`

Al igual que `$_GET`, es una variable propia de PHP y se usa para recolectar los valores de un formulario que ha enviado los datos usando el método POST. Cada valor del arreglo se accede de la misma forma que con la variable `$_GET`.

Las principales ventajas de usar POST es que las variables no se muestran en el URL de la página y tampoco hay límite para la longitud de la información enviada.

Retomando el ejemplo anterior pero con el método POST, tendríamos:

Código HTML

```
<form action="prueba.php" method="post">

Nombre: <input type="text" name="nombre" />
Edad: <input type="text" name="edad" />
<input type="submit" />

</form>
```

Código PHP

```
<?php
$nombre = $_POST['nombre'];
$edad = $_POST['edad'];
?>
```

Nota: Usando POST las variables son “invisibles” para los usuarios y no existe límite en la cantidad de información a enviar

PHP, formularios y acción!

Ahora que sabemos como obtener los valores de un formulario usando las variables internas de PHP vamos a divertirnos un poco. Imaginemos que se desea conocer la opinión de los visitantes de nuestro sitio web. Lo primero que debemos hacer es diseñar el formulario para obtener la información.

Código HTML (formulario.html):

```
<html>
<head>
<title>Formularios y PHP</title>
</head>
<body>

<form action="comment.php" method="post">

Nombre:<br/>
<input type="text" name="nombre" value="su nombre" size="20"><br/>
Correo electrónico:<br/>
<input type="text" name="correo" value="su correo" size="20"><br/>
Comentario:<br/>
<input type="text" name="comentario" value="su comentario" size="40"><br/>
<br/>
<input type="submit" value="Enviar">
<input type="reset" value="Borrar">

</form>

</body>
</html>
```

En el código anterior se han resaltado los elementos clave del formulario; los atributos *action* y *method* del form y el atributo *name* de cada elemento. Veamos ahora como procesamos este formulario usando PHP. El formulario se verá en la ventana del navegador tal y como lo muestra la Figura 27.

Para manejar los datos y generar una respuesta al usuario podemos implementar un script en el archivo *comment.php* como el siguiente :

```
<?php

$nombre = $_POST['nombre'];
$correo = $_POST['correo'];
$comentario = $_POST['comentario'];

echo "<html>
<head>
<title>Formularios y PHP</title>
</head>
<body>
<p><strong>Fecha:</strong> ".date('r')."<br/>
<strong>Usuario: </strong>".$nombre."<br/>
<strong>Comentario:</strong> ".$comentario."</p>
</body>
</html>";

?>
```

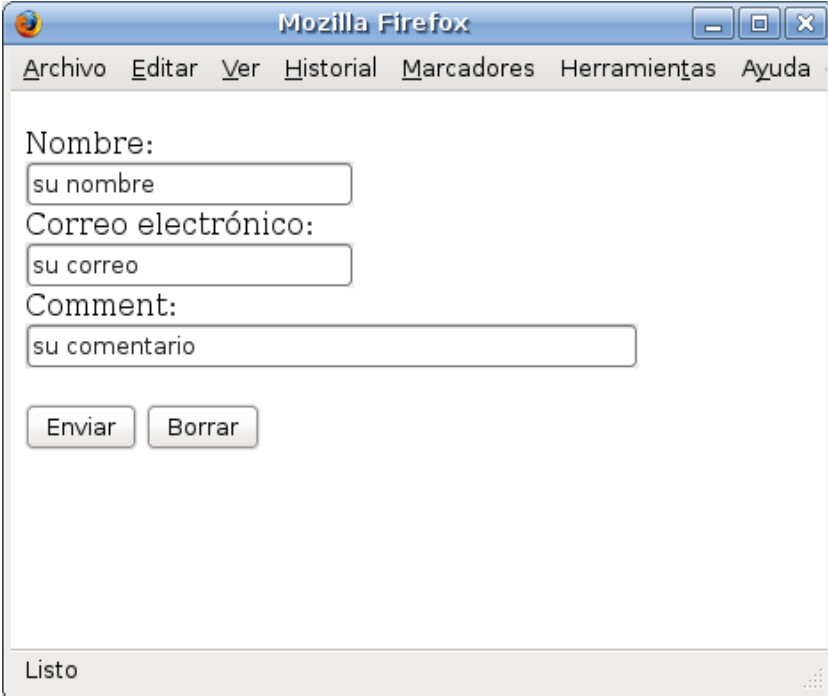
A screenshot of a Mozilla Firefox browser window. The title bar says "Mozilla Firefox". The menu bar includes "Archivo", "Editar", "Ver", "Historial", "Marcadores", "Herramientas", and "Ayuda". The main content area displays a form with three labels: "Nombre:", "Correo electrónico:", and "Comment:". Each label is followed by a text input field. The input fields contain the placeholder text "su nombre", "su correo", and "su comentario" respectively. Below the input fields are two buttons: "Enviar" and "Borrar". The status bar at the bottom of the window displays the word "Listo".

Figura 27. Formulario para captar datos en PHP

El resultado será una página como la que se observa en la Figura 28.

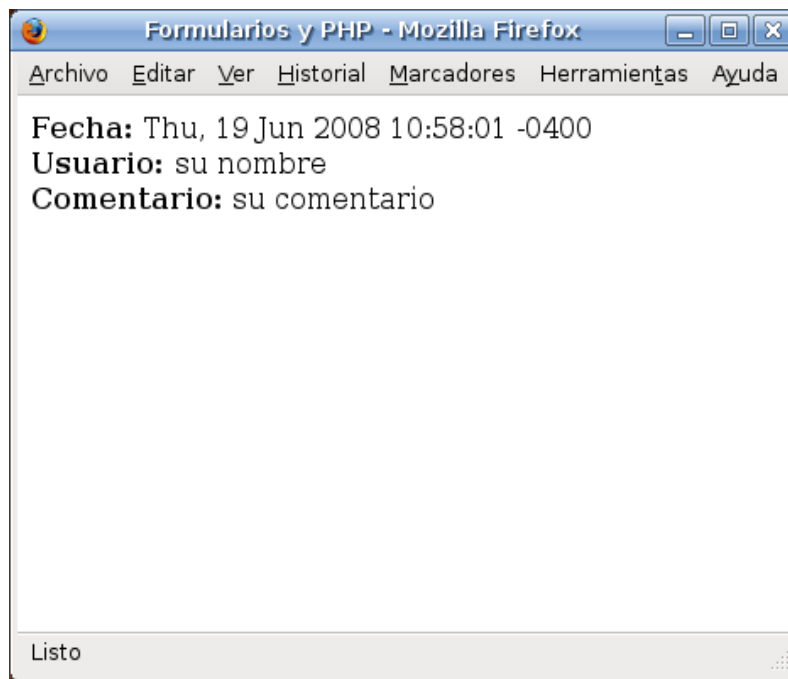


Figura 28. Salida HTML del script PHP anterior

Nota: Una de las cosas más importantes que se deben tener en cuenta cuando se trabajan con formularios HTML y PHP es que todos los elementos HTML del formulario estarán disponibles automáticamente para nuestro script PHP luego de realizar el envío (submit).

Hemos creado nuestra primera interacción entre HTML y PHP pero vamos a hacerlo un poco más divertido. Agreguemos unos cuantos campos más a la consulta; el sexo del usuario, la calificación que el usuario le daría a nuestro sitio y un pequeño check button para que el visitante nos diga si volvería a visitar nuestra página.

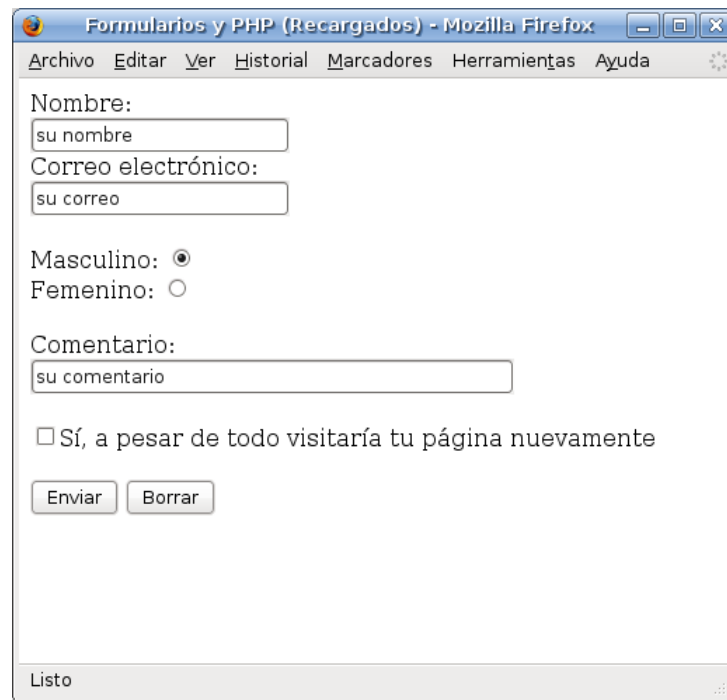
El código HTML se vería así:

```
<html>
<head>
<title>Formularios y PHP (Recargados)</title>
</head>
<body>

<form action="shit.php" method="post">

Nombre:<br/>
<input type="text" name="nombre" value="su nombre" size="20"><br/>
Correo electrónico:<br/>
<input type="text" name="correo" value="su correo" size="20"><br/><br/>
Masculino: <input type="radio" name="sexo" value="masculino" checked="checked"><br/>
Femenino: <input type="radio" name="sexo" value="femenino"><br/><br/>
Comentario:<br/>
<input type="text" name="comentario" value="su comentario" size="40"><br/><br/>
<input type="checkbox" name="volver" value="si">Sí, a pesar de todo visitaría tu página
nuevamente<br/>
<br/>
<input type="submit" value="Enviar">
<input type="reset" value="Borrar">
```

```
</form>
</body>
</html>
```



Formularios y PHP (Recargados) - Mozilla Firefox

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

Nombre:
su nombre

Correo electrónico:
su correo

Masculino: ☒
Femenino: ☐

Comentario:
su comentario

☐ Sí, a pesar de todo visitaría tu página nuevamente

Enviar Borrar

Listo

Figura 29. Resultado del formulario en la ventana del navegador.

El código en PHP para manejar el formulario sería algo como:

```
<?php
    $nombre = $_POST['nombre'];
    $correo = $_POST['correo'];
    $comentario = $_POST['comentario'];
    $sexo = $_POST['sexo'];
    $volver = $_POST['volver'];

    echo "<html>
    <head>
    <title>Formularios y PHP</title>
    </head>

    <body>
    <p><strong>Fecha:</strong> ".date('r')."<br/>
    <strong>Usuario: </strong>".$nombre."<br/>
    <strong>Sexo: </strong>".$sexo."<br/>
    <strong>Comentario:</strong> ".$comentario."<br/><br/>";

    if (strcmp($volver, 'si') == 0)
        echo "¡Al fin! Un usuario que volvería a visitar nuestra web";
    else
        echo "Este visitante ni de casualidad vuelve a pasar por aquí";
```

```
echo "</p>  
</body>  
</html>";  
?>
```

Y luego de procesar el script PHP obtenemos un resultado como éste:

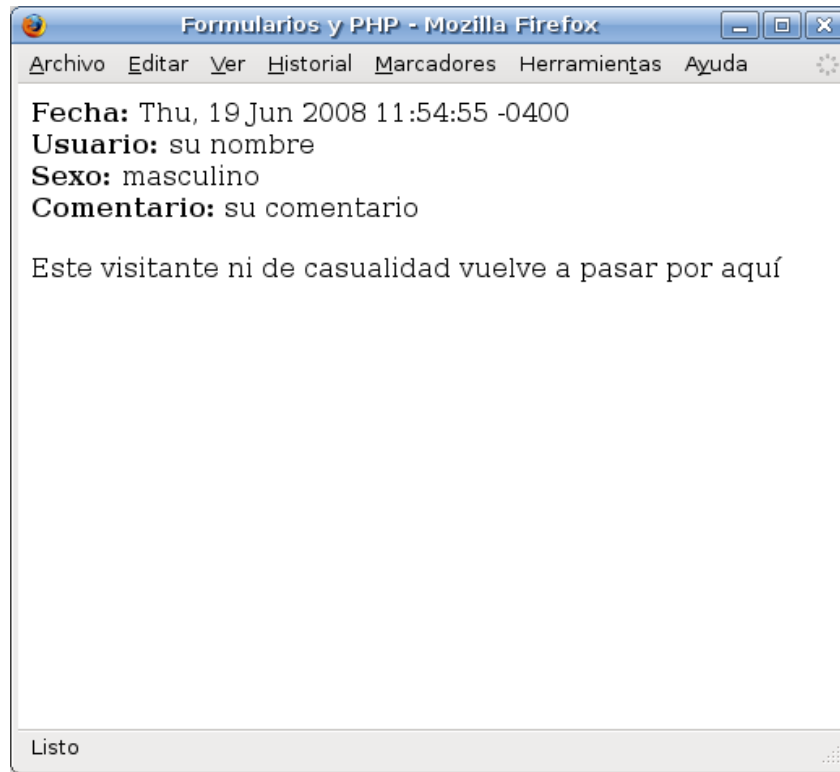


Figura 30. HTML generado por el script PHP

Ejercicio 21-1. Formulario de registro II

Diseñe un script en PHP que permita manejar el formulario del Ejercicio 10-1 y devuelva como respuesta una tabla HTML con los datos del usuario.



The screenshot shows a web browser window titled "Ejercicio de Formularios - Mozilla Firefox 3 Beta 5". The browser's menu bar includes "Archivo", "Editar", "Ver", "Historial", "Marcadores", "Herramientas", and "Ayuda". The main content area displays a registration form with the heading "Bienvenido al sistema de registro". The form contains the following fields and controls:

- Nombre y apellido:** A text input field.
- Usuario:** A text input field.
- Contraseña:** A text input field.
- Sexo:** A dropdown menu with the text "- selecciona -".
- Tipo de conexión:** Three radio buttons labeled "Cable/ADSL", "Dial-Up" (which is selected), and "No disponible".
- Avatar:** A text input field followed by a button labeled "Examinar...".
- Subscription options:** Two checkboxes labeled "Suscribirme a la lista de correos" and "Suscribirme al boletín de novedades".
- Buttons:** Two buttons at the bottom labeled "Enviar" and "Borrar datos del formulario".

At the bottom of the browser window, the status bar shows the word "Listo".

Figura 31. Formulario de registro del ejercicio 10-1.

Capítulo 22. Caracteres Especiales

Durante todo este manual hemos hecho una pequeña trampa a la hora de explicar las etiquetas y poner ejemplos, pues se han omitido ciertas exigencias del HTML respecto al uso de **caracteres especiales** que habitualmente se usan en los textos pero que no se pueden incluir directamente en un documento HTML. Esto se hizo con la idea de facilitar la comprensión de las ideas fundamentales sobre HTML, sin desviar la atención del lector en otros detalles.

Estos caracteres especiales son los que usa el HTML para su sintaxis (<, >, ", &, etc.) y los que **no** son propios del idioma inglés. Lamentablemente para nosotros los habla-hispanos la ñ y las letras acentuadas no están en ese grupo (tampoco están ç, ÷, i, entre otros). Existen varias soluciones a este inconveniente, pero tomaremos la más sencilla de todas. Vamos a sustituir todos los caracteres problemáticos por sus correspondientes **entidades HTML**.

Una entidad HTML es un conjunto de caracteres codificados que representan a un caracter especial. En la siguiente tabla se listan los caracteres especiales más comunes y sus respectivas entidades HTML:

| Caracter | Entidad HTML |
|---------------------|--------------|
| < | < |
| > | > |
| & | & |
| " | " |
| ' | ' |
| (espacio en blanco) | |
| á | á |
| é | é |
| í | í |
| ó | ó |
| ú | ú |
| Á | Á |
| É | É |
| Í | Í |
| Ó | Ó |
| Ú | Ú |
| ü | ü |
| Ü | Ü |
| ñ | ñ |
| Ñ | Ñ |
| ÷ | ¿ |
| ¡ | ¡ |

Volviendo al reconocimiento de culpa que se hizo por ocultar este “detalle”, es necesario aclarar que muchos de los ejemplos antes expuestos no eran del todo correctos. Por ejemplo, cuando se escribió:


```
<p>Éste es un párrafo y éste es un <a href="www.google.com">enlace</a></p>
```

En realidad debería haberse escrito:

```
<p>&Eacute;ste es un p&aacute;rrafo y &eacute;ste es un <a href="www.google.com">enlace</a></p>
```

Es por esto que se recomienda al lector a revisar y escribir con la codificación de caracteres correcta todos los ejemplos listados en la guía.

Apéndice A. Tabla de etiquetas

A continuación se presenta una tabla con un resumen de todas las etiquetas HTML estudiadas en la guía, a manera de referencia y con la idea de facilitar su búsqueda.

| Etiqueta | Descripción |
|--------------|---|
| <html> | Empieza y termina el documento HTML |
| <head> | Empieza y termina la zona de encabezado |
| <title> | Empieza y termina la zona de título |
| <body> | Empieza y termina el cuerpo del documento |
| <p> | Permite definir los párrafos que conforman un texto. |
| <h1>...<h6> | Define los títulos de mayor importancia de la página, enumerados en forma decreciente del 1 al 6. |
| | Permite aplicar un énfasis cursivo al texto marcado. |
| | Permite aplicar un efecto de negritas al texto marcado. |
| <ins> | Se emplea para marcar una inserción de texto en el contenido original de la página web. |
| | Se emplea para marcar un borrado de texto en el contenido original de la página web. |
| | Inserta una nueva línea en la página web. |
| <blockquote> | Se emplea para indicar que el contenido marcado es una cita textual de otro contenido externo. |
| <abbr> | Se emplea para marcar las abreviaturas del texto y proporcionar su significado. |
| <acronym> | Se emplea para marcar acrónimos o siglas y proporcionar su significado. |
| <dfn> | Se emplea para marcar las definiciones de ciertos términos y proporcionar su significado. |
| <cite> | Se emplea para marcar una cita o referencia a otras fuentes. |
| <pre> | Muestra el texto tal cual como está escrito (respetando los espacios en blanco). |
| <code> | Se emplea para delimitar un fragmento de texto considerado código fuente. |
| <a> | Se emplea para crear enlaces a recursos. |
| <script> | Se emplea para enlazar o definir un bloque de código (generalmente JavaScript). |
| <link> | Se emplea para enlazar y establecer relaciones entre el documento y otros recursos. |
| | Se emplea para definir el inicio y el fin de una lista no ordenada. |
| | Se emplea para definir el inicio y el fin de una lista ordenada. |

| | |
|------------|--|
| | Se emplea para definir cada elemento de las lista ordenadas y no ordenadas. |
| <dl> | Se emplea para definir el inicio y el fin de una lista de definiciones. |
| <dt> | Se emplea para definir los términos de una lista de definición. |
| <dd> | Se emplea para indicar las definiciones de los términos de la lista. |
| | Se emplea para incluir imágenes en los documentos HTML. |
| <table> | Se emplea para definir una tabla de datos. |
| <tr> | Se emplea para definir cada fila de la tabla de datos. |
| <td> | Se emplea para definir cada una de las celdas de datos de la tabla, es decir las columnas. |
| <th> | Se emplea para definir las celdas que son cabeceras de una fila o una columna en la tabla. |
| <caption> | Define el título o leyenda de una tabla. |
| <div> | Define zonas o divisiones en una página HTML. |
| <form> | Define el comienzo y el fin de un formulario. |
| <input> | Se emplea para insertar un control en un formulario. |
| <textarea> | Se emplea para insertar áreas de texto en un formulario |
| <select> | Se emplea para insertar listas desplegables en un formulario. |
| <option> | Se emplea para definir cada elemento de una lista desplegable. |

Apéndice B. Respuestas

Ejercicio 1-1. Marcado de texto

```
<html>
<head>
<title>Ejercicio de Efectos de texto</title>
</head>
<body>

<h2>La sonda Phoenix descubre posibles rastros de hielo en Marte</h2>

<p><strong>Washington. (EFE).-</strong> La sonda Phoenix ha encontrado posibles rastros
de hielo en torno a la zona próxima al polo norte de Marte donde descendió, informó hoy
el
<acronym title="Laboratorio de Propulsión a Chorro">JPL</acronym>.</p>

<p><cite>Ray Arvidson</cite>, científico de la Universidad de Washington y encargado de
las operaciones del
brazo robótico, indicó: <blockquote><em>"Podríamos estar viendo rocas, o podríamos estar
viendo hielo
en el lugar del descenso"</em></blockquote></p>

<p>Ese hielo pudo haber quedado al descubierto como resultado de la emisión de gases
candentes de los cohetes
que redujeron la velocidad del descenso de la nave, que llegó a Marte el pasado
domingo.</p>

<p><em>"Hemos completado todas las tareas básicas de ingeniería y se han hecho todos los
despliegues cruciales"</em>, indicó <cite>Barry Goldstein</cite>, director del
proyecto.</p>
</body>
</html>
```

Ejercicio 4-1. Prueba de enlaces

```
<html>
<head>
<title>Ejercicio de Enlaces</title>
</head>
<body>
<h1>Enlaces</h1>
<p>HTML permite realizar varios tipos de enlaces. </p>
<p>- Enlaces a recursos externos como por ejemplo, <a
href="http://www.google.co.ve">Google</a></p>
<p>- Enlaces a recursos locales como nuestro ejercicio de <a href="text-
```

```
effects.html">Efectos de texto</a></p>
<p>- Enlaces a un <a href="archivo.zip">archivo</a> en un servidor remoto</p>
<p>- Enlaces a <a href="mailto:example@gmail.com">correos electrónicos</a></p>
</body>
</html>
```

Ejercicio 5-1. Listas anidadas

```
<html>
<head><title>Ejercicio de listas</title></head>
<body>

<h1>Indice</h1>

<ol>
  <li>Introducción</li>
  <li>Características Básica</li>
    <ul>
      <li>Elementos HTML</li>
      <li>Estructura Básica</li>
      <li>Sintáxis HTML</li>
    </ul>
  <li>Listas</li>
  <dl>
    <dt><strong>No Numeradas</strong>
      <dd>No llevan ningún orden particular
    <dt><strong>Numeradas</strong>
      <dd>Son enumeradas en forma creciente
    <dt><strong>De definición</strong>
      <dd>Actúan como glosarios con parejas <em>término/definición</em>
  </dl>
</ol>

</body>
</html>
```

Ejercicio 7-1. Tablas

```
<html>
<head><title>Ejercicio de tablas</title></head>
<body>

<h1>Pedido</h1>

<table border="1">
<tr>
  <td scope="col"><strong>Unid.</strong></td>
  <td scope="col"><strong>Descripción</strong></td>
  <td scope="col"><strong>Precio unitario</strong></td>
```

```

    <td scope="col"><strong>Subtotal</strong></td>
</tr>

<tr>
    <td>1</td>
    <td>Pen Drive USB 4GB</td>
    <td>140,00</td>
    <td>140,00</td>
</tr>

<tr>
    <td>4</td>
    <td>CD Vírgenes</td>
    <td>1,50</td>
    <td>6,00</td>
</tr>

<tr>
    <td>3</td>
    <td>Cartuchos de impresora</td>
    <td>105,00</td>
    <td>315,00</td>
</tr>

<tr>
    <td colspan="3" scope="row"><strong>Total</strong></td>
    <td><strong>461,00</strong></td>
</tr>
</table>

</body>
</html>

```

Ejercicio 10-1. Formulario de registro

```

<html>
<head>
<title>Ejercicio de Formularios</title>
</head>
<body>
<h3>Bienvenido al sistema de registro</h3>
<form>
Nombre y apellido: <br/>
<input type="text" name="nombre" value="" size="50"/><br/>
Usuario: <br/>
<input type="text" name="user" value="" /><br/>
Contraseña: <br/>
<input type="password" name="pass" value="" /><br/><br/>

Sexo:
<select id="sexo" name="sexo">
    <option value="" selected="selected">- selecciona -</option>
    <option value="m">Masculino</option>
    <option value="f">Femenino</option>
</select> <br/><br/>

```

```

Tipo de conexión: <br/>
<input type="radio" name="conex" value="dsl" checked="checked" />Cable/ADSL <br/>
<input type="radio" name="conex" value="dial" />Dial-Up <br/>
<input type="radio" name="conex" value="nd" />No disponible <br/><br/>
Avatar: <br/>
<input type="file" name="adjunto" /> <br/><br/>
<input name="suscl" type="checkbox" value="suscl"/>Suscribirse a la lista de correos<br/>
<input name="suscl2" type="checkbox" value="suscl2"/>Suscribirse al boletín de
novedades<br/> <br/>

<input type="submit" name="enviar" value="Enviar" />
<input type="reset" name="limpiar" value="Borrar datos del formulario" />

</form>

</body>
</html>

```

Ejercicio 15-1. Palabras

```

<?php
    $palabra="Esta es una frase de prueba";

    echo str_replace(" ", "<br/>", $palabra);

?>

```

Ejercicio 15-2. New slang

```

<?php
    $frase="Quiero que esta frase se imprima con el new slang";

    $nueva_frase=str_replace("a", "4", $frase);
    $nueva_frase=str_replace("A", "4", $nueva_frase);
    $nueva_frase=str_replace("e", "3", $nueva_frase);
    $nueva_frase=str_replace("E", "3", $nueva_frase);
    $nueva_frase=str_replace("s", "5", $nueva_frase);
    $nueva_frase=str_replace("S", "5", $nueva_frase);

    echo "La frase normal se lee: ".$frase."<br/>";
    echo "La frase new slang se lee: ".$nueva_frase;

?>

```

Ejercicio 15-3. Detectar palabras

```
<?php
    $frase="Esta frase por alguna parte tiene la palabra color";

    echo strpos($frase, "color");

?>
```

Ejercicio 17-1. Jugando con strings y condicionales

```
<?php
    $numdia = date("w");

    /*
    PHP hace el moldeado de los datos automaticamente
    asi que puede escribirse también sin el (int)
    */
    $hora = (int) date("H");

    $saludo="";
    $dia="";

    if ($numdia=="0")
        $dia="domingo";
    elseif ($numdia=="1")
        $dia="lunes";
    elseif ($numdia=="2")
        $dia="martes";
    elseif ($numdia=="3")
        $dia="miércoles";
    elseif ($numdia=="4")
        $dia="jueves";
    elseif ($numdia=="5")
        $dia="viernes";
    elseif ($numdia=="6")
        $dia="sábado";

    if (($hora >= 0) && ($hora < 12))
        $saludo="Buenos días, ";
    elseif (($hora >= 12) && ($hora < 18))
        $saludo="Buenos tardes, ";
    elseif (($hora >= 18) && ($hora <= 23))
        $saludo="Buenos noches, ";

    echo $saludo."hoy es ".$dia;

?>
```

Ejercicio 17-2. Edades

```
<?php
    $edad=57;
```



```
if ($edad < 18)
    echo "Esta persona es menor de edad";
elseif (($edad >= 18) && ($edad < 55))
    echo "Esta persona es mayor de edad";
elseif ($edad >= 55)
    echo "Esta persona ya está jubilada";
?>
```

Ejercicio 17-3. ¿Mayor o menor?

```
<?php
$num1=10;
$num2=20;

if ($num1 > $num2)
    echo "El número 1 es mayor que el número 2";
elseif ($num1 < $num2)
    echo "El número 2 es mayor que el número 1";
else
    echo "Ambos números son iguales";
?>
```

Ejercicio 17-4. Calificaciones

```
<?php
$calif = 8;

if (($calif >= 0) && ($calif < 5))
    $c="E";
elseif (($calif >= 5) && ($calif < 10))
    $c="D";
elseif (($calif >= 10) && ($calif < 14))
    $c="C";
elseif (($calif >= 14) && ($calif < 17))
    $c="B";
elseif (($calif >= 17) && ($calif <= 20))
    $c="A";

echo $c;
?>
```

Ejercicio 18-1. Switch en lugar de if...elseif

```

<?php
    $numdia = date("w");

    /*
    PHP hace el moldeado de los datos automaticamente
    asi que puede escribirse también sin el (int)
    */
    $hora = (int) date("H");

    $saludo="";
    $dia="";

    switch ($numdia){
        case "0":
            $dia="domingo";
            break;
        case "1":
            $dia="lunes";
            break;
        case "2":
            $dia="martes";
            break;
        case "3":
            $dia="miércoles";
            break;
        case "4":
            $dia="jueves";
            break;
        case "5":
            $dia="viernes";
            break;
        case "6":
            $dia="sábado";
            break;
    }

    if (($hora >= 0) && ($hora < 12))
        $saludo="Buenos días, ";
    elseif (($hora >= 12) && ($hora < 18))
        $saludo="Buenos tardes, ";
    elseif (($hora >= 18) && ($hora <= 23))
        $saludo="Buenos noches, ";

    echo $saludo."hoy es ".$dia;

?>

```

Ejercicio 18-2. Calificaciones II

```

<?php
    $calif = "H";

    $mens="";
    switch ($calif){
        case "A":

```

```

        $mens="Excelente trabajo";
        break;
    case "B":
        $mens="Muy bien. Sigue asi";
        break;
    case "C":
        $mens="Bien, pero aun puedes mejorar";
        break;
    case "D":
        $mens="Debe mejorar para aprobar";
        break;
    case "E":
        $mens="Bajo rendimiento";
        break;
    case "F":
        $mens="Reprobado";
        break;
    default:
        $mens="Esa calificación no es válida";
}

echo $mens;

```

?>

Ejercicio 18-3. Rangos de temperatura.

```

<?php
    $temp = 8;

    $mens="";
    switch ($temp){
        case 0:
            $mens="Temperatura en cero";
            break;
        case 1:
        case 2:
            $mens="Tempreatura entre 1 y 2 ºC";
            break;
        case 3:
        case 4:
            $mens="Tempreatura entre 3 y 4 ºC";
            break;
        case 5:
        case 6:
            $mens="Tempreatura entre 5 y 6 ºC";
            break;
        case 7:
        case 8:
            $mens="Tempreatura entre 7 y 8 ºC";
            break;
        case 9:
        case 10:
            $mens="Tempreatura entre 9 y 10 ºC";
            break;
        default:
    }

```

```
        $mens="Tempreatura fuera de rango";
    }
    echo $mens;

?>
```

Ejercicio 19-1. Pirámide

```
<?php
    $x=1;
    while ($x < 7){
        for ($i=0; $i<$x; $i++){
            echo "*";
        }
        $x++;
        echo "<br/>";
    }

?>
```

Ejercicio 19-2. Primos

```
<?php
    $x=2;

    while ($x < 50){
        $primo=false;
        for ($i=2; $i < $x; $i++){
            if (($x % $i) == 0) {
                $primo=true;
                break;
            }
        }
        if ($primo==false) echo $i." es un número primo<br/>";
        $x++;
    }

?>
```

Ejercicio 19-3. Temperaturas.

```
<?php

    echo "<html>
```

```

<head>
<title>Conversion de temperatura</title>
</head>
<body>
<table border='1'>
    <tr>
        <td>Farhenheit</td>
        <td>Celsius</td>
    </tr>";

    for ($i=0; $i <= 300; $i += 20){
        echo "<tr>
        <td>". $i. "</td>
        <td>". ((5*($i-32))/9). "</td>
        </tr>";
    }

    echo "</table>
    </body>
    </html>";
?>

```

Ejercicio 19-4. Los 100 primeros

```

<?php
    $x=0;
    $suma=0;

    while ($x <= 100){
        $suma += $x;
        $x++;
    }

    echo $suma;
?>

```

Ejercicio 19-5. Pirámide II

```

<?php
    $n=7;
    $x=1;

    while ($x <= $n){
        for ($i=1; $i <= $x; $i++)
            echo $i;
        $x++;
        echo "<br/>";
    }

?>

```

Ejercicio 20-1. Geometría básica

```
<?php
    $radio=4;

    function perimetro($r){
        return 2*pi()*$r;
    }

    function area($r){
        return pi()*$r*$r;
    }

    echo "El perimetro del circulo es: ". perimetro($radio)."<br/>";
    echo "El area del circulo es: ". area($radio)."<br/>";
?>
```

Ejercicio 20-2. Factorial

```
<?php
    $num=10;

    function factorial($n){
        $fact=1;

        for ($i=$n; $i > 1; $i--){
            $fact *= $i;
        }

        return $fact;
    }

    echo "El factorial de ".$n." es: ". factorial($num);
?>
```

Ejercicio 21-1. Formulario de registro II

```
<?php
    $nombre = $_POST['nombre'];
    $usuario = $_POST['user'];
    $password = $_POST['pass'];
    $sx = $_POST['sexo'];
    $cx = $_POST['conex'];
    $suscl = $_POST['suscl'];
    $suscl2 = $_POST['suscl2'];
```

```

if ($sx=="m")
    $sexo="Masculino";
elseif ($sx=="f")
    $sexo="Femenino";
else
    $sexo="No disponible";

if ($cx=="dsl")
    $conex="ADSL";
elseif ($cx=="dial")
    $conex="Dial-Up";
elseif ($cx=="nd")
    $conex="No disponible";

echo "<html>
<head>
<title>Formularios y PHP</title>
</head>
<body>
<table border='1'>";

echo "<tr>
    <td><strong>Nombre</strong></td>
    <td>".$nombre."</td>
</tr>";
echo "<tr>
    <td><strong>Usuario</strong></td>
    <td>".$usuario."</td>
</tr>";
echo "<tr>
    <td><strong>Contraseña</strong></td>
    <td>".$password."</td>
</tr>";
echo "<tr>
    <td><strong>Sexo</strong></td>
    <td>".$sexo."</td>
</tr>";
echo "<tr>
    <td><strong>Conexión</strong></td>
    <td>".$conex."</td>
</tr>";
if ($suscl=='enable')
    echo "<tr>
        <td><strong>Suscrito a lista de correo</strong></td>
        <td>Si</td>
    </tr>";
if ($susc2=='enable')
    echo "<tr>
        <td><strong>Suscrito al boletín de novedades</strong></td>
        <td>Si</td>
    </tr>";

echo "</table>
</body>
</html>";
?>

```

Licencia de Documentación Libre GNU

GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- * A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

- * B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

- * C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

- * D. Preserve all the copyright notices of the Document.

- * E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- * F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

- * G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

- * H. Include an unaltered copy of this License.

- * I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year,

authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- * J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- * K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

- * L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

- * M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

- * N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

- * O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Referencias Bibliográficas

- [01] HTML Básico. <http://entren.dgsca.unam.mx/Html/intr.htm>
- [02] HTML - Wikipedia. http://es.wikipedia.org/wiki/C%C3%B3digo_HTML
- [03] Comenzando con HTML + CSS. <http://www.w3.org/Style/Examples/011/firstcss.es.html>
- [04] Introducción a XHTML | LibrosWeb.es. <http://www.librosweb.es/xhtml/>
- [05] Web Browser - Wikipedia. http://en.wikipedia.org/wiki/Web_browser
- [06] Manual práctico de HTML. <http://www-app.etsit.upm.es/~alvaro/manual/manual.html>
- [07] Tutorial Básico PHP. <http://geneura.ugr.es/~maribel/php/>
- [08] PHP Tutorial - Introduction. <http://www.tizag.com/phpT/>
- [09] Nuestro primer PHP. Manual de PHP. Tutorial de PHP. WebEstilo. <http://www.webestilo.com/php/php01.phtml>
- [10] PHP Syntax. <http://www.w3schools.com/PHP/default.asp>
- [11] PHP: Manual de PHP. <http://ve2.php.net/manual/es/>
- [12] GNU Free Documentation License. <http://www.gnu.org/licenses/fdl.html>