# TEMA 2 – HTML.

## *ÍNDICE*

# 1. Esquema de funcionamiento de un servicio web

La Web funciona siguiendo el denominado **modelo cliente-servidor**, habitual en las aplicaciones que funcionan en una red: existe un servidor, que es quien presta el servicio, y un cliente, que es quien lo recibe.

### Cliente web

El cliente web es un programa con el que el usuario interacciona para solicitar a un servidor web el envío de páginas de información. Estas páginas se transfieren mediante el protocolo HTTP.

Las páginas que se reciben son documentos de texto codificados en lenguaje HTML. El cliente web debe interpretar estos documentos para mostrárselos al usuario en el formato adecuado.

Además, cuando lo que se recibe no es un documento de texto, sino un objeto multimedia (vídeo, sonido, etc.) no reconocido por el cliente web, éste debe activar una aplicación externa capaz de gestionarlo.

Entre los clientes web (también conocidos como visualizadores o navegadores) más usuales están: *Mozilla Firefox, Microsoft Internet Explorer, Opera, Safari y Google Chrome*. La mayoría de ellos soportan también otros protocolos, como por ejemplo el FTP (*File Transfer Protocol*), para la transferencia de ficheros.

### Servidor web

El servidor web es un programa que está permanentemente escuchando las peticiones de conexión de los clientes.

El servidor funciona de la siguiente manera: si encuentra en su sistema de ficheros el documento HTML solicitado por el cliente, lo envía y cierra la conexión; en caso contrario, envía un código de error que cierra la conexión. El servidor web también se ocupa de controlar los aspectos de seguridad, comprobando si el usuario tiene acceso a los documentos.

El proceso completo, desde que el usuario solicita una página hasta que el cliente web se la muestra con el formato adecuado, es el siguiente:

1. El usuario especifica en el cliente web la dirección (URL) de la página que desea consultar.
2. El cliente establece la conexión con el servidor web (por ejemplo, Apache).
3. El cliente solicita la página deseada.
4. El servidor busca la página que ha sido solicitada en su sistema de ficheros, Si la encuentra, la envía al cliente; en caso contrario, devuelve un código de error. Si la petición necesita la generación dinámica de una página HTML, por ejemplo, por necesitar hacer una consulta a una base de datos, debe haber un intérprete del lenguaje utilizado (p.e. PHP o ASP) en el servidor, que interactúa con un programa que maneja la base de datos y va generando paginas en HTML que se envían al cliente (lo veremos con detenimiento más adelante).
5. El cliente interpreta los códigos HTML y muestra la página al usuario.
6. Se cierra la conexión.

## 2. Introducción a HTML

**Definición:**

HTML (*HyperText Markup Language*) es una implementación del estándar SGML (*Standard Generalized Markup Language*). SGML es un estándar internacional para la definición de texto electrónico independiente de dispositivos, sistemas y aplicaciones.
Los autores utilizan un código de formato (en inglés *markup*) en sus documentos para representar información.

Cada lenguaje de formato de documentos definido con SGML se llama aplicación SGML. Una aplicación SGML se caracteriza generalmente por:

1) Una declaración SGML. La declaración SGML especifica qué caracteres y delimitadores pueden aparecer en la aplicación.
2) Una definición del tipo de documento (*document type definition*, DTD) que define la sintaxis de las estructuras de formato. El DTD puede incluir definiciones adicionales, tales como referencias a entidades de caracteres.
3) Una especificación que describe la semántica que se debe conferir al código de formato. Esta especificación también impone restricciones de sintaxis que no pueden expresarse dentro del DTD.
4) Documentos que contienen datos (contenido) y código (*markup*). Cada documento contiene una referencia al DTD que debe usarse para interpretarlo.

HTML es, por tanto, un caso particular de lenguaje de formato de documentos. Veamos un ejemplo de un documento HTML:

```
<!DOCTYPE html>
<HTML lang="es">
	<HEAD>
		<META charset="UTF-8">
		<TITLE>Mi primer documento HTML</TITLE>
	</HEAD>
	<BODY>
		<P>¡Hola mundo!</P>
	</BODY>
</HTML>
```

HTML, por lo tanto, es un lenguaje basado en marcas; una marca afecta a un trozo de texto dándole un formato, estructura o significado determinado. Un documento HTML contiene tanto la información que se desea presentar como instrucciones (marcas) para describir su presentación. También podemos decir que es un lenguaje de hipertexto, ya que dentro del documento existen áreas sensibles (hipervínculos) que al activarlas permiten acceder a otros elementos.

Un documento HTML se divide en una **sección de cabecera** (aquí, entre <HEAD> y </HEAD>) y un **cuerpo** (aquí, entre <BODY> y </BODY>). El título del documento aparece en la cabecera (junto con otras informaciones sobre el documento), y el contenido del documento aparece en el cuerpo. El cuerpo de este ejemplo contiene únicamente un párrafo, codificado o marcado como <P>.

**Versiones:**

- HTML 3.2, W3C Recommendation, 14 January 1997, Dave Raggett, Author
- HTML 4.01, W3C Recommendation, 24 December 1999, Dave Raggett, Arnaud Le Hors, Ian Jacobs, Editors
- HTML5, W3C Recommendation, 28 October 2014.

Para más información:
- Especificaciones html: http://www.w3.org/community/webed/wiki/HTML/Specifications
- HTML5:
    - Recomendación HTML5 publicado en Octubre del 2014.
    - http://www.w3.org/html/wiki/FAQs#When_can_I_use_HTML5.3F
    - http://es.wikipedia.org/wiki/HTML5

**Editores:**

¿Dónde hay que editar el código fuente? Cualquier editor de texto sin formato vale, eso sí, los ficheros deben tener extensión html (aunque se pueden encontrar documentos html muy antiguos con extensión htm). En Windows, por ejemplo, bastaría con el Bloc de notas.
Existen editores de html más avanzados que, por ejemplo, codifican las etiquetas con algún sistema de colores. Dos de los más utilizados son el HTML-KIT, que se descarga desde la página www.chami.com y el NOTEPAD++PORTABLE.

**Navegadores:**

El navegador es el software que interpreta el lenguaje HTML. Existen muchos navegadores. Una página web que usa HTML estándar es accesible desde cualquier navegador. Pero existen diferencias entre unos navegadores y otros a la hora de interpretar determinado código no estándar HTML, por eso lo mejor es ajustarse al estándar y no usar etiquetas que hacen cosas muy chulas pero sólo valen para un determinado navegador. Nuestro objetivo debe ser que cualquier usuario pueda ver bien (tal y como nosotros queremos) nuestra página web. Es absurdo ver mensajes que digan “Página optimizada para...” y ahí un navegador o resolución de pantalla (¿vas a tener que cambiar la resolución o el navegador sólo para ver una página?). Tus webs tienen que ser *optimizadas* para verse con cualquier navegador.

**URL (Uniform Resource Locator):**

La URL define objetos en una red Internet. En ella se contienen datos sobre:
- El tipo de objeto (objetos asociados con alguno de los protocolos o servicios disponibles en Internet: ftp, http, mailto, telnet, etc.).
- El nodo de la red en que se encuentra dicho objeto.
- El fichero físico que contiene el objeto.

Estructura: ***Servicio://[host]:[puerto]/[path del fichero]***

*Ej:* http://microsoft.com/download/aspdoc.zip

Si se omite el puerto se tomaría el válido por defecto para el protocolo o servicio utilizado (ej: puerto 80 para servicios web).

Tipos: absolutas y relativas:

- Absolutas: especifican un path completo.
  http://www.sobreasp.com/download/aspdoc.zip
- Relativas: especifican un path relativo a la url del documento.
  *imagenes/dibujo1.gif*

**Sintaxis:**

**Generalidades:**

- Son válidos todos los caracteres incluidos en Unicode / ISO 10646.
- El formato es libre. El formato introducido en el fichero fuente (saltos de línea, líneas en blanco, etc.) es irrelevante para el formato final del documento.
- Determinados caracteres tienen un significado especial:
  < Marca el comienzo de una etiqueta.
  > Marca el final de una etiqueta.
  & Marca el comienzo de una referencia a entidad.

  Estos caracteres, en caso de que sea necesario utilizarlos, se sustituyen por el nombre de la entidad que los representa en el repertorio ISO Latín 1 (ISO 8859-1):
  & se escribe &
  < se escribe <
  > se escribe >
  espacio en blanco se escribe  

Los nombres de las entidades están compuestos por el signo &, luego el nombre de la entidad y al final ;. Si se trata de un número de entidad en lugar de un nombre, añadiremos # después del ampersand (&) y a continuación el número de la entidad – si se expresa en hexadecimal irá precedido de una x - y al final ;.

Por ejemplo, para mostrar el signo "<" tenemos tres posibilidades:

Por su nombre de entidad **<**

Indicando su correspondencia decimal en la tabla ASCII **<**

Indicando su correspondencia hexadecimal en la tabla ASCII **<**

El uso más común de los caracteres especiales es el espacio en blanco.

Si en un texto figuran cinco espacios en blanco seguidos, HTML automáticamente borra cuatro.
Para incluir espacios en blanco usamos " " y HTML los dejará en su lugar.

Otro uso muy frecuente es el de insertar acentos en el código html por medio de los números de las entidades: al principio, en algunos navegadores las letras acentuadas y algunos caracteres especiales como la “ñ” no se visualizaban correctamente, por lo cual debían ser sustituidos por la referencia a su entidad.

La siguiente tabla muestra una lista de estos caracteres especiales:

| | | |
|---|---|---|
| • signo < | correspondencia en HTML | `<` |
| • signo > | correspondencia en HTML | `>` |
| • signo & | correspondencia en HTML | `&` |
| • signo " | correspondencia en HTML | `&quot;` |
| • espacio en blanco | correspondencia en HTML | ` ` |
| • Espacio ASCII | correspondencia en HTML | `&#x0020;` |
| • Tabulador ASCII | correspondencia en HTML | `&#x0009;` |
| • Avance de página ASCII | correspondencia en HTML | `&#x000C;` |
| • Espacio de anchura cero | correspondencia en HTML | `&#x200B;` |
| • Retorno de carro | correspondencia en HTML | `&#x000D;` |
| • Á, á | correspondencia en HTML | `&Aacute; &aacute;` |
| • É, é | correspondencia en HTML | `&Eacute; &eacute;` |
| • Í, í | correspondencia en HTML | `&Iacute; &iacute;` |
| • Ó, ó | correspondencia en HTML | `&Oacute; &oacute;` |
| • Ú, ú | correspondencia en HTML | `&Uacute; &uacute;` |
| • Ñ, ñ | correspondencia en HTML | `&Ntilde; &ntilde` |

Para más información: *http://ascii.cl/es/codigos-html.htm*

**Etiquetas (*tags*):**

- **Son los textos que delimitan los distintos elementos que componen un documento**. Los elementos de HTML suelen estar formados por una marca de inicio < >, un contenido y una marca de final </ >, aunque hay algunos elementos especiales que sólo tienen marca de inicio y se llaman **elementos vacíos.**
- No son sensibles a mayúsculas y minúsculas (*aunque* ***es aconsejable escribir todo en minúsculas porque el próximo estándar así lo exigirá***).
- Hay dos tipos de etiquetas: etiquetas de comienzo y etiquetas de final de elemento. La mayoría de los identificadores necesitan ambas etiquetas aunque unos pocos no necesitan la de final.

  1) **Etiquetas de comienzo de elemento**. Delimitadas por los caracteres "<" y ">".
  *<identificador>*
  *Ej: <head>*

  2) **Etiquetas de final de elemento**. Delimitadas por los caracteres "</" y ">".
  *</identificador>*
  *Ej: </head>*

- Muchas etiquetas tienen atributos, los cuales modifican el funcionamiento de la misma. Sintaxis:
  *<identificador [atributos]>*

  Estructura general de un atributo: *Literal=valor*

  Normalmente, el valor de un atributo es una cadena de caracteres entre dobles comillas. Dentro de ella no se pueden poner los siguientes caracteres " , >, &. Si es necesario ponerlos se sustituyen por " > y &

*Ejemplo:*

*<font face=”arial” size=”12”>*
*....*
*</font>*

**Comentarios:**

- Texto introducido en un documento que no aparece en el formato final.
  *<!-- comentario -->*
  *Ej: <!-- Esto no aparecerá en la página web aunque ocupe varias líneas -->*

## 3. Estructura de un documento HTML

Un documento HTML consta de las siguientes partes:

1. Identificación SGML
2. Una etiqueta de comienzo de documento <html>
3. Cabecera (iniciada por la etiqueta <head> y cerrada por </head>)
4. Cuerpo del documento (iniciada por la etiqueta <body> y cerrada por </body>)
5. Una etiqueta de fin de documento </html>

```
<!DOCTYPE HTML>
<html lang="es">
   <head>
      ...
   </head>
   <body>
      ...
   </body>
</html>
```

A continuación veremos cada una de estas secciones.

### 3.1. Identificación SGML

Permite identificar la DTD (la estructura de etiquetas del documento, los atributos que pueden tener las etiquetas, los eventos, etc.) adecuada para procesar el documento.

Hasta HTML 4 se especificaban tres DTDs, (ver anexo HTML4). Con HTML 5 el doctype en es más sencillo:

<!DOCTYPE html>

Podemos validar un documento HTML *online* desde la página oficial de W3C http://validator.w3.org/. Aquí encontraremos 3 opciones: validar una web con URL, validar un fichero o validar código HTML copiado directamente a la web.

Hoy en día, algunos navegadores disponen de complementos que ayudan a validar código y a desarrollar. *Por ejemplo Firefox* y Chrome. En Firefox tenemos el complemento ***HTML validator****:* se descarga de Internet y se instala a través del menú Herramientas/Complementos/Extensiones. Una vez instalado aparece un botón en la esquina

inferior derecha del navegador que nos permitirá validar el código HTML asociado a la página. Otro complemento interesante de Firefox es Firebug.

## 3.2. Etiqueta <html>

En HTML5 es obligatorio definir qué lenguaje o idioma estamos utilizando para los contenidos. En la etiqueta <html> se define el idioma por defecto del documento mediante el atributo **lang.**

```
<html lang="es">
```

Además, no se pueden indicar varios idiomas en una misma página (el atributo lang sólo acepta un código de idioma o una cadena de texto vacía). Para poder escribir alguna parte del documento en otro idioma, debes añadir el atributo lang a los elementos que estén escritos en otro idioma:

<**p lang**="en"> ... párrafo en inglés ... </p>

Utiliza las etiquetas de idioma del *Registro de subetiqueta de idioma de la IANA*.

## 3.3. Cabecera <head>

Contiene información general acerca del propio documento y su visualización.

Se identifica con la etiqueta **<head>** y finaliza por tanto con **</head>.**

En su ámbito se pueden emplear diferentes elementos referenciados por sus etiquetas, los más relevantes son:

- **<title>** [cadena de caracteres] **</title>**
  Da título al documento, en la mayoría de los navegadores se visualiza en la barra de título.

- **<base href=”URL”>**
  Indica la localización de los ficheros, gráficos, sonidos, etc. a los que se hace referencia en nuestra página web. Si no se incluye esta directiva, el navegador entiende que los elementos se encuentran en el mismo lugar que nuestra página, por lo que esta etiqueta suele usarse cuando los ficheros están en otro servidor.
  *<base href="http://www.miweb.com/archivos/">*

- **<meta http-equip=”VALOR1” | name=”VALOR2” | charset=”VALOR3” content=VALOR4>**
  Las etiquetas <meta> o “*meta tags*” proporcionan información sobre el documento HTML. Son identificadores ocultos, es decir instrucciones especiales del lenguaje HTML que no son mostradas directamente en el navegador, pero que pueden ser utilizadas por navegadores, motores de búsqueda u otros servicios web.

  Existen **tres tipos** de *meta tags*, con distintas funciones:

  a) LAS META NAMES: se utilizan para optimizar el resultado en los motores de búsqueda. Contienen información referida al documento HTML (autor, fecha de modificación, descripción, etc). El formato es:

```
<meta name="valor_name" content="valor_content">
```

El atributo “name” puede tomar los siguientes valores entre otros: *author* (autor), *description* (descripción), *keywords* (palabras clave), *lang* (idioma), etc.

Ejemplos:

**<meta name=**”author” **content**=”Profesor”>
**<meta name=**”lang” **content**=”es”>
**<meta name=**”description” **content**=”Página sobre Palencia”>
**<meta name=**”keywords” **content**=”Palencia, Castilla, monumentos, ciudad, castellana, carrión”>

b) LAS HTTP-EQUIV: este atributo se utiliza para especificar información de las cabeceras http que regulan el diálogo entre el servidor y el navegador. Los valores establecidos por este metadato pueden ser utilizados por el servidor al entregar la página al navegador del usuario, y son utilizadas para refinar la información y dar instrucciones al navegador que las está leyendo. El formato es:

```
<meta http-equiv="valor_http-equiv" content="valor_content">
```

El atributo http-equiv puede tomar los siguientes valores entre otros: default-style (el estilo preferido), refresh (intervalo de refresco del documento).

Ejemplos:

<**meta http-equiv**= "default-style" **content**="hoja-por-defecto" />
<**meta http-equiv**= "default-style" **content**="estilo2" />
*<!—Indica la hoja de estilos por defecto o el estilo por defecto a utilizar -->*

<**meta http-equiv**="refresh" **content**="3">
*<!-- La página se recarga cada tres segundos-->*

c) CHARSET: especifica la codificación de caracteres utilizada. En HTML5 la codificación por defecto si no se especifica nada es UTF-8. No obstante, es el navegador el que finalmente decide la codificación a utilizar. Para evitar un visionado incorrecto en algunos navegadores o navegadores mal configurados, **es recomendable incluir siempre la etiqueta meta charset**.

<**meta charset**=”UTF-8”>

## 3.4. Cuerpo del documento <body>

Es el contenedor de la información propia del documento.
Se identifica con la etiqueta BODY que puede tener ciertos atributos, como por ejemplo BACKGROUND para poner una imagen como fondo de la página o BGCOLOR para poner un color de fondo a la página.

<**body** [background=”url de imagen”] [bgcolor=”color”]>

Veamos algún ejemplo:

1) Cuerpo de documento con imagen de fondo

```
<body background="imagenes/patito.gif" >
        .....
</body>
```

2) Cuerpo de documento con color de fondo

```
<body  bgcolor="#FF0000”>
        ......
</body>
```

El color de fondo suele especificarse en modo RGB (*Red Green Blue*). Los valores RGB se indican en hexadecimal. Para conseguir un color, mezclaremos valores de esta manera: RRGGBB donde cada valor puede crecer desde 00 hasta FF. Por ejemplo *FF0000* es el color rojo. Para indicar que está en hexadecimal anteponemos el carácter #.

| | |
|---|---|
| Naranja | #FF8000 |
| Verde turquesa | #339966 |
| Azul oscuro | #000080 |

Además de en modo RGB, los 16 colores de la paleta VGA Standard pueden ponerse en formato texto (nombre del color en inglés), aunque esto último no es aconsejable porque puede haber problemas con los distintos navegadores:

| | |
|---|---|
| Black = "#000000" | Green = "#008000" |
| Silver = "#C0C0C0" | Lime = "#00FF00" |
| Gray = "#808080" | Olive = "#808000" |
| White = "#FFFFFF" | Yellow = "#FFFF00" |
| Maroon = "#800000" | Navy = "#000080" |
| Red = "#FF0000" | Blue = "#0000FF" |
| Purple = "#800080" | Teal = "#008080" |
| Fuchsia = "#FF00FF" | Aqua = "#00FFFF" |

*Ej: <body bgcolor="red”>*

Veamos ahora el aspecto básico de la mínima página web:

```
<html lang="es">
<head>
    <meta name=”author” content=”Pepito Pérez”>
    <meta name=”lang” content=”es”>
    <meta name=”description” content=”Página sobre Palencia”>
    <meta name=”keywords” content=”Palencia, Castilla, monumentos”>
    <meta http-equiv="Content-Type" content="text/html”;
        charset=”iso-8859-1"> <!-- codificación española -->
    <title>Título de la página web que sale en el navegador</title>
</head>
<body>
      <!-- Aquí es donde va el contenido del documento, es decir,
      lo que se verá en el navegador. Ahora vamos a aprender a
      darle formato -->
</body>
</html>
```

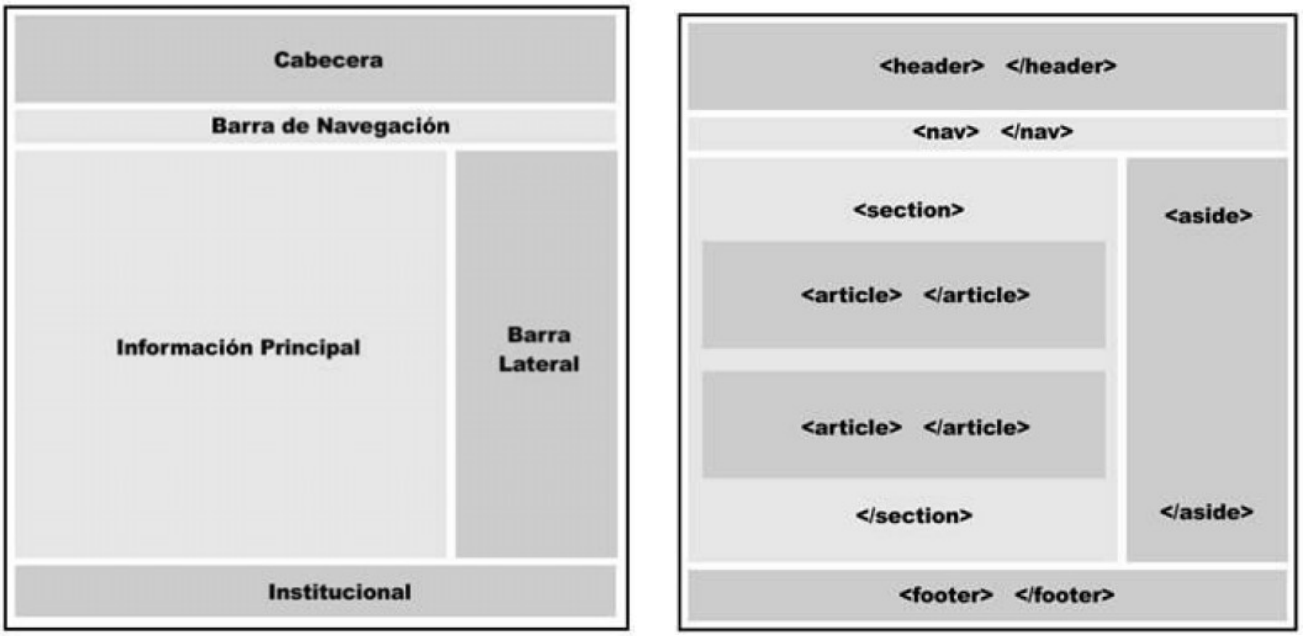
En el ámbito del cuerpo del documento se pueden emplear distintos elementos para dar formato al contenido del mismo (negrita, subrayado, tablas, formularios, …). Esto es lo que vamos a explicar en el siguiente apartado.

**Elementos del cuerpo del documento:**

### 3.4.1 Estructura del cuerpo. Etiquetas semánticas.

La idea de HTML5 es que los elementos HTML, más que para formato, sirvan para dar valor semántico al contenido. Es decir, para indicar qué tipo de contenido es. Con las siguientes etiquetas se marca la semántica y con las hojas de estilo CSS que veremos en la siguiente unidad didáctica le damos formato a esa semántica. Con todo ello se consigue que la estructura de la web sea más coherente y fácil de entender.

En la imagen de la izquierda tenemos un diseño común de cualquier sitio web actual, con las diferentes secciones que lo componen. En la imagen de la derecha se muestran los diferentes elementos de HTML5 para cada una de las secciones.




- **<header></header>**
  Cabecera de la página o de una sección. Por lo general siempre se define un header principal donde se incluye el logo o el nombre del sitio, pero además se pueden (y deben) definir otros elementos header dentro de los elementos section o article.
- **<nav></nav>**
  Ofrece ayuda para la navegación, usualmente menús principales o grandes bloques de enlaces. Debe ser utilizado sólo para la navegación principal del sitio y no para enlaces externos.
  Es un elemento muy versátil, normalmente aparece insertado dentro de un elemento header o footer pero puede aparecer en cualquier otra parte del cuerpo siempre que mantenga su finalidad.
- **<section></section>**
  Contiene la información más relevante del documento y puede ser encontrada en diferentes formas (por ejemplo, dividida en varios bloques o columnas).
- **<aside></aside>**
  Puede contener datos relacionados con la información principal pero que no son relevantes o igual de importantes, como aclaraciones del contenido o referencias. Y también puede contener información no directamente relacionada con el contenido que lo rodea, como elementos publicitarios.
- **<footer></footer>**

Contiene información general sobre el autor, el copyright, la fecha de última modificación o la organización.

- **<article></article>**
  Define contenido autónomo o independiente, con la intención de ser reutilizado de modo aislado, es decir, información que puede ser entendida como un *todo* de forma íntegra. Tiene sentido dentro del elemento <section> e incluso puede aparecer de forma independiente.

Es muy importante tener en cuenta que estas etiquetas no indican su posición en la página Web, sino su valor semántico.

### 3.4.2 Cabeceras

Existen 6 niveles de cabeceras:

- Cabecera de nivel 1: <h1>Texto de la cabecera</h1>
- Cabecera de nivel 2: <h2>Texto de la cabecera</h2>
- Cabecera de nivel 3: <h3>Texto de la cabecera</h3>
- Cabecera de nivel 4: <h4>Texto de la cabecera</h4>
- Cabecera de nivel 5: <h5>Texto de la cabecera</h5>
- Cabecera de nivel 6: <h6>Texto de la cabecera</h6>

El formato en que se visualizan las cabeceras depende de su nivel, variando:

- Tamaño de la letra
- Tipo de resaltado
- Líneas a saltar antes y después del texto.

*Ejemplos:*

```
<h1>Cabecera de nivel 1</h1>
<h3>Cabecera de nivel 3</h3>
```

### 3.4.3 Bloques de texto

Definen la estructura de un bloque:

-  Párrafo
  Etiqueta de párrafo normal. Hace que el texto que está contenido en esta etiqueta sea considerado un párrafo. A ese texto se le colocará un salto de párrafo al final. Los saltos de línea y los espacios blancos seguidos introducidos no se representan (se comprimen en un único espacio). La longitud de las líneas viene definida por el tamaño de la ventana del navegador. Para provocar un salto de línea hay que usar la etiqueta <br/>. La etiqueta de finalización  no es obligatoria, aunque conviene ponerla.
  Admite el parámetro align = (left|center|right|justify) para definir la alineación del texto dentro del bloque.

```
<p>texto</p>
```

- <pre> Texto con formato previo
  Conjunto de texto que se muestra como se introdujo en el formato original, incluyendo los saltos de línea y los espacios en blanco.
  Se recomienda no utilizarlo en la medida de lo posible.

```
<pre>texto</pre>
```

- <address> Dirección
  Información sobre el autor del documento, dirección, etc.

```
<address>texto</address>
```

- <blockquote> Anotación
  Sirve para escribir una cita, el texto se presenta indentado y en un formato distinto al del párrafo normal.

```
<blockquote>texto</blockquote>
```

-  Permite agrupar varios bloques de texto en uno solo, heredando todos ellos la alineación especificada mediante el parámetro align=(left|center|right|justify). Se trata de una etiqueta muy potente que demuestra su potencial cuando se usa con las hojas de estilo.

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

### 3.4.4 Elementos generales

- **<br/>** Salto de línea. Fuerza que se parta una línea de texto independientemente del formato en que se esté trabajando. Se puede usar sin la barra final, pero en el próximo estándar será obligatorio, así es que es mejor ponerla.

```
<br>     mejor: <br/>
```

  Extensiones de la etiqueta <br/>: su uso está indicado para romper la secuencia de texto cuando se sitúa este alrededor de una imagen

```
<br clear=left> Busca el primer margen libre a la izquierda.
<br clear=right> Busca el primer margen libre a la derecha.
<br clear=all> Busca el primer margen libre a ambos lados.
```

- **<hr>** Línea horizontal.
  Admite varios atributos:
  noshade  Hace que la línea no aparezca en relieve
  width = ”tamaño” Anchura de la línea. Puede ser relativa (en porcentaje, p.e. 50%) o absoluta (en píxeles).
  align=”alineación” Indica cómo se alineará la línea. (Left, Center, Right o Justify –esta última admitida sólo por algunos navegadores-).
  size=”n” Grosor de la línea

```
<hr  width=”300” noshade align=”left” size=5 color=”blue”>
```

### 3.4.5 Etiquetas de resaltado de textos

Se utilizan para enfatizar o resaltar una zona del texto. Todas las etiquetas tienen el mismo formato:

```
<etiqueta>texto</etiqueta>
```

Dos tipos:

1. Estilos lógicos: asociados a distintos tipos de información.
   - sup: Superíndice
   - sub: Subíndice
   - em: Indica énfasis (*emphasis*). Usualmente los navegadores usan la cursiva.
   - strong: Indica un énfasis más fuerte.
   - cite: Contiene una cita o una referencia a otras fuentes.
   - dfn: Indica que aquí es donde se define el término encerrado.
   - code: Designa un fragmento de código de computadora.
   - samp: Designa una muestra de la salida de un programa, script, etc.
   - kbd: Indica texto que debe ser introducido por el usuario.
   - var: Indica que el texto es una variable o un argumento de un programa.
   - abbr: Indica una forma abreviada (p.ej., WWW, HTTP, URI, Mass., etc.).
   - q: el contenido es una cita textual de una frase
   - del: el texto afectado ha sido eliminado en esta versión del documento
   - ins: el texto afectado ha sido añadido en esta versión del documento
   - time: define una fecha y/o hora.
   - small: Representa un *comentario aparte*, es decir, textos como un descargo de responsabilidad o una nota de derechos de autoría, que no son esenciales para la comprensión del documento.
   - mark: Representa texto resaltado con propósitos de *referencia*, es decir por su relevancia en otro contexto.

   em y strong se usan para indicar énfasis. Los demás elementos de frase tienen significados particulares en documentos técnicos.

   Ejemplos:

   ```
   H<sub>2</sub>O
   E = mc<sup>2</sup>

   Como dijo <cite>Harry S. Truman</cite>,
   <q lang="en-us">The buck stops here.</q>

   Se puede encontrar más información en <cite>[ISO-0000]</cite>.

   Por favor, utilice el siguiente número de referencia en nuestra correspondencia
   futura: <strong>1-234-55</strong>
   ```

2. Estilos físicos: asociados a distintos tipos de letra.
   - **<b>** : Letra en negrita.
     *Nota: actualmente el uso de <b> se ha sustituido por *
   - **<i>** : Letra inclinada (itálica o cursiva)
   - **<u>** : Letra subrayada.
     No se recomienda su uso porque confunde al usuario (normalmente los subrayados se interpretan como si fueran enlaces)

La siguiente frase muestra varios tipos de texto:

```
<p><b>negrita</b>, <i>itálica</i>, <b><i>negrita e itálica</i></b>,
texto <u>subrayado</u> y <small>pequeño <small>más
pequeño</small></small>.</p>
```

### 3.4.6 Etiquetas de control de las fuentes de texto

- **<font size=... color=... face=... >**

  Indica el tipo de letra.

  size: Tamaño (de 1 a 7, siendo 3 el tamaño por defecto). Puede ser absoluto (size=4) o relativo (size=+2 significa subir dos tamaños a la letra).

  color: Color de la fuente en formato RGB.

  face: Nombre de la fuente a utilizar. Se pueden indicar varias fuentes (en orden de preferencia) separadas por comas. Esto hace que si la primera no está disponible en el ordenador del usuario, se use la segunda y si no la tercera, … y así sucesivamente.

```
<font  size=+1 color="#008000">Texto</font>
```

### 3.4.7 Listas

HTML ofrece a los autores varios mecanismos para especificar listas de información. Existen 5 tipos diferentes de listas.

El comienzo de una lista desplaza el margen izquierdo a la derecha. El final de una lista lo devuelve a su posición anterior.

En todos los casos la etiqueta <li> indica que el texto que sigue es una nueva entrada en la lista, esta etiqueta puede no llevar </li> pero es aconsejable ponerlo.

Las listas pueden anidarse.

- Lista sin ordenar **<ul>**
  Lista de textos, cada entrada de la lista comienza por un carácter de señalización (*bullet*).

  Atributos: type=”tipo”
  Indica que tipo de *bullet* se usará (Circle | Disk | Square).

```
<ul>
      <li> Texto </li>
      <li> Texto </li>
      .....
</ul>
```

- Lista numerada **<ol>**

Cada elemento de la lista comienza por un número, igual a su posición en la lista.

```
<ol>
      <li> Texto </li>
      <li> Texto </li>
      .....
</ol>
```

La única lista que todavía guarda sus atributos en la versión 5 de HTML es la lista ordenada delimitada por la directiva <ol>. Las demás listas se pueden formatear utilizando CSS, pero aunque la directiva <ol> guarda parte de sus atributos, estos tienen equivalencia en CSS y se recomienda su uso. Los tres atributos de la lista ordenada controla:

- el tipo: **type**. Permite cambiar el tipo de numeración entre numérica (cifras árabes o romanas) o a través de las letras (minúsculas o mayúsculas). Solamente tenemos que añadir como valor al atributo el signo (1, A, a, i, I).
- el orden (ascendente o descendente): reversed. Controla el orden de la numeración que por defecto viene ascendente.

```
<ol reversed>
   <li>Elemento</li>
   <li>Elemento</li>
   <li>Elemento</li>
</ol>

3. Elemento
2. Elemento
1. Elemento
```

- el inicio de la enumeración: start. Especificar que inicie la lista por el número indicado.

```
<ol type="A" start="3">
   <li>Elemento</li>
   <li>Elemento</li>
   <li>Elemento</li>
</ol>

C. Elemento
D. Elemento
E. Elemento
```

- Lista sencilla **<menu>**
  Utilizado para menús contextuales, barras de herramientas y para listar formularios de control y comandos. *(Algunos navegadores lo muestran como si fuera una lista sin ordenar, es decir, con carácter de señalización, por lo que están en desuso)*

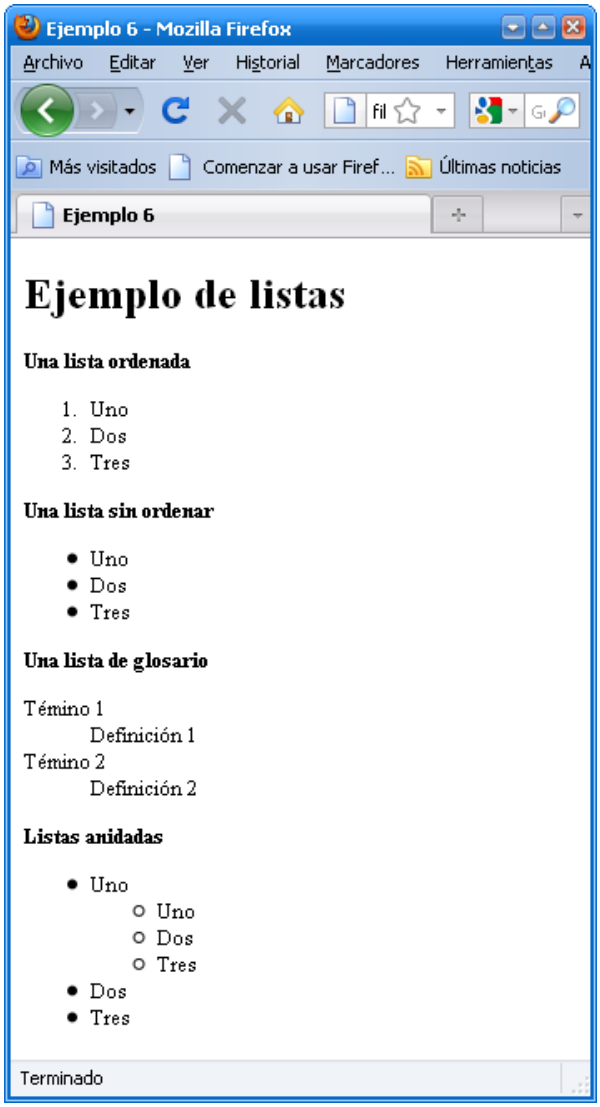
```
<menu>
      <li> Texto </li>
      <li> Texto </li>
      .....
</menu>
```

- Lista de definiciones: **<dl>**

Cada entrada en la lista tiene dos partes: el término que se define (encabezado por la etiqueta **<dt>**) y la definición (encabezada por la etiqueta **<dd>**).

```
<dl>
      <dt>Termino1</dt>
            <dd>Definición1</dd>
      <dt>Termino2</dt>
            <dd>Definición2</dd>
      .....
</dl>
```

Ejemplo: Salida del navegador y código fuente asociado.




```
<html lang="es">
<head>
   <title>Ejemplo 6</title>
</head>

<body>
<font face=”trebuchet ms”>
<h1>Ejemplo de listas</h1>

<strong>Una lista ordenada </strong>
<ol>
<li>Uno</li>
<li>Dos</li>
<li>Tres</li>
</ol>

<strong>Una lista sin ordenar </strong>
<ul>
<li>Uno</li>
<li>Dos</li>
<li>Tres</li>
</ul>

<strong>Una lista de glosario </strong>
<dl>
<dt>T&eacute;mino 1</dt>
<dd>Definici&oacute;n 1</dd>
<dt>T&eacute;mino 2</dt>
<dd>Definici&oacute;n 2</dd>
</dl>

<strong>Listas anidadas </strong>
<ul>
<li>Uno
   <ul>
      <li>Uno</li>
      <li>Dos</li>
      <li>Tres</li>
   </ul>
</li>
<li>Dos</li>
<li>Tres</li>
</ul>
</font>

</body>
</html>
```

### 3.4.8 Hiperenlaces o hipervínculos

La etiqueta <a> permite crear enlaces a otras páginas u objetos.
Atributos:

- href="URL"

Identifica el destino del enlace. La URL puede ser absoluta (http://www.ya.com), relativa (pagina3.html), de salto hacia un marcador interno (#marca1) o una mezcla entre ambas (pagina3.html#marca1)
Ejemplo: *<a href="http://www.microsoft.com">La Web de Microsoft</a>*

- name="nombre"

Coloca un marcador con ese identificador en la posición en la que se encuentre la etiqueta <a>, permitiendo hacer un enlace a otro lugar en nuestra página.

<a NAME=nombre del enlace></a>

Ejemplo:

<!-- Establecer un marcador dentro de una página -->
<a name="marca1"></a>

<!-- En otro lugar del documento podemos poner un enlace a esa marca así: -->
<a href="#marca1">Documentación pública</a>

<!-- En otra página web podemos poner un enlace a esa marca así: (suponemos que el marcador se encuentra en un fichero llamado página3.html) -->
<a href="pagina3.html#marca1">Documentación pública</a>

- target="destino"

Indica en qué parte de la ventana se muestra el enlace. Es indispensable para utilizar con los marcos. Lo veremos más adelante.

Veamos un ejemplo de hipervínculos:

```
<html lang="es">
<head>
   <title>Ejemplo hipervínculos</title>
</head>

<body>
   <a name="arriba"><h1>Página de enlaces</h1></a>
   <a href="#abajo">Ir abajo</a><br/>

   <a href="ej4.html">Ir a ejemplo 4</a><br/><br/>
   <a href="http://www.google.com/">Ir a google</a><br/>

   <br/>.<br/>.<br/>.<br/>.<br/>.<br/>.<br/>.<br/>.<br/>
   <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
   <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
   <br/><br/><br/><br/><br/>.<br/>.<br/>.<br/>.<br/>.<br/>

   <a name="abajo"><br/></a>
   <a href="#arriba">Ir arriba</a>
</body>
</html>
```

### 3.4.9 Imágenes

Las imágenes se incluyen en una página web con la etiqueta <img>.
Atributos:

**<img**
**alt=**texto_alternativo
**src=** URL_de_la_imagen
**width=**ancho en puntos o porcentaje(%)
**height=**alto en puntos o porcentaje(%)
**usemap=**elemento mapa asociado a la imagen.
**ismap=**si aparece indica que la imagen es un mapa interactivo.
**>**

***A partir de HTML5, la alineación de la imagen, su borde y todo lo referente a su aspecto en la página web se realiza con CSS***

**src**: URL del fichero que contiene la imagen. Como siempre, puede ser relativa o absoluta. Es un parámetro obligatorio. Si no se indica lo contrario, se busca la imagen en el mismo directorio en el que se encuentra el fichero HTML del servidor.

**alt**: Texto alternativo a presentar si no se encuentra la imagen. Sirve para que los navegadores lo muestren en un cartel amarillo, para mostrar el texto mientras se carga la imagen y para crear registros en los buscadores de imágenes. También se utiliza para que en navegadores como LINX que no soportan gráficos la imagen no genere un error en la carga de la imagen y muestre en su lugar el texto alternativo.

**usemap:** Define en donde está el archivo que engloba las coordenadas, el tipo de forma del mapa de imagen y el enlace establecido (por ejemplo #nombre). La sintaxis es igual a una referencia interna dentro un documento HTML

**ismap**: Es un atributo booleano. Su presencia indica que la imagen es un mapa interactivo.

```
<img usemap=″#mapa″  ismap/>
```

¿Puede una imagen ser utilizada dentro de un hiperenlace en lugar del texto normal?. La respuesta es “Sí”, ejemplo:

```
<a  href=″http://www.google.es″> <img  src=″IMAGEN1.JPG″ border=1></a>
```

Es aconsejable poner ancho y alto en pixels, porque así el navegador, cuando está cargando la página, dibuja el recuadro y continúa cargando la página, en lugar de parar la carga de la página hasta que se carga la imagen. Además así nos aseguramos que la imagen ocupa el espacio que queremos.

### 3.4.10 Mapas sensibles

La estructura de mapas sensibles nos permite asignar diferentes áreas de una imagen a diferentes hipervínculos. Consta de dos elementos:

**1)** Una estructura de mapa:

**<map name=..... >**
**<area** shape=..... coords=..... href=..... **>**
**.....**
**</map>**

- <**map** name=..... >
  Etiqueta de apertura del Mapa, define el nombre de este.

- <**area** shape=..... coords=..... (href=..... | nohref)>
  Definición de la zona activa del mapa. Las zonas activas se definen con respecto al recuadro que ocupa la imagen.

  Atributos:

  | | |
  |---|---|
  | href | Hipervínculo de la zona activa. |
  | nohref | La zona activa no tiene ningún hipervínculo. |
  | shape | Define la forma de la zona activa (rectángulo, círculo, polígono). |
  | coords | Coordenadas de la zona activa dentro de la imagen. Se especifican tomando como origen de los ejes x-y la esquina superior izquierda de la imagen. Su formato depende de la forma de la zona activa. |

  shape=**rect** coords="*left_x, top_y, right_x, bottom_y*"
  shape=**circle** coords="*center_x, center_y, radius*"
  shape=**poly** coords="$x1_1$,$y1_1$, $x2_2$,$y2_2$, $x3_3$,$y3_3$, ..."

  Por ejemplo, si la imagen insertada tiene width=400 (ancho) y height=300 (alto), las coordenadas de los cuatro vértices del recuadro que ocupa la imagen serían:

  - Superior izquierdo: x=0, y=0 - Inferior izquierdo: x=0, y=300
  - Superior derecho: x=400, y=0 - Inferior derecho: x=400, y=300

  Conociendo estos valores, podemos deducir o estimar las coordenadas de cualquier punto dentro de la imagen y así definir zonas sensibles circulares, rectangulares o poligonales.

**2)** Atributo para la etiqueta **img:** en el que haremos referencia a la estructura de mapa a utilizar.

<img src=..... **usemap**="#Nombre del mapa">

**Ejemplo:** mapa de 300 x 250 con dos zonas activas, un círculo centrado en la imagen, y un rectángulo ocupando el cuadrante superior izquierdo.

```
<map  name="Mapa1">
<area  shape="circle" coords="150, 125, 50" href="http://www.inicio.com">
<area  shape="rect" coords="0, 0, 150, 125" href="ejemploenlaces.html">
</map>
<IMG SRC="imagenes/hidra.gif" height=250 width=300 USEMAP="#Mapa1">
```

### 3.4.11 Tablas

Las tablas son elementos que están compuestos por filas y columnas. Se emplean para conseguir maquetaciones complejas, es decir, permiten componer y diseñar documentos posicionando sus elementos alineados y en distintas columnas. Las tablas se pueden anidar.

Una tabla se compone de los siguientes elementos:

- Definición de la tabla: **<table>**
- Título: **<caption>**
- Definición de fila: **<tr>**
- Definición de cabecera de columna: **<th>**
- Definición de casilla: **<td>**

*Nota: Normalmente los navegadores consideran las celdas de cabecera <th> como títulos de tabla y las visualizan en negrita (formato).*

Uso:

<**table**
align=alineación horizontal
width=ancho en puntos o porcentaje
border=valor
cellspacing=valor
cellpadding**=**valor>

<**caption** align=alineación horizontal>
*Título*
</**caption**>

<**tr**
align=alineación horizontal
valign=alineación vertical>

*Nota: cuando ponemos th | td el carácter | significa que los atributos valen tanto para th como para td.*

<**th | td**
colspan=número
rowspan=número
align=alineación horizontal
valign=alineación vertical>

<**th | td**>
...
</**tr**>

<**tr**>
...
</**tr**>

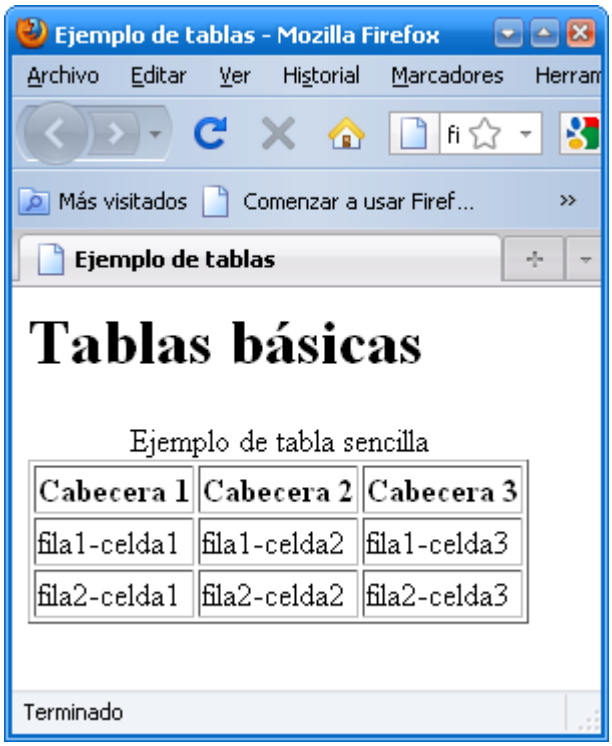
...
</**table**>

Veamos un ejemplo:

```
<html lang=”es”>
<head>
     <title>Ejemplo de tablas</title>
</head>

<body>
<h1>Tablas básicas</h1>
<table  border=”1”>
<caption>Ejemplo de tabla sencilla</caption>
<tr>
         <th>Cabecera 1</th>
         <th>Cabecera 2</th>
         <th>Cabecera 3</th>
</tr>
<tr>
         <td>fila1-celda1</td>
         <td>fila1-celda2</td>
         <td>fila1-celda3</td>
</tr>
<tr>
         <td>fila2-celda1</td>
         <td>fila2-celda2</td>
         <td>fila2-celda3</td>
</tr>
</table>
</body>
</html>
```

La salida sería:



Ahora vamos a ver qué atributos pueden tener las tablas.

Atributos de <**table**>:

align=”alineación”
Alineación horizontal de la tabla (left, center o right).

width=”ancho”
Anchura de la tabla (en píxeles o en porcentaje).

height=”altura”
Altura de la tabla (en píxeles o en porcentaje). No es un atributo incluido en el estándar.

cellpadding=”píxeles”
Indica el espacio interior en las tablas (desde el borde de las celdas al texto).

cellspacing=”píxeles”
Indica el espacio entre las celdas de la tabla.

border=”píxeles”
Indica el número de píxeles que mide el borde. Un valor 0 indica que no hay borde.

frame=”*valor*”
Este atributo especifica qué lados del marco que rodea a una tabla serán visibles. Valores posibles: void | above | below | hsides | lhs | rhs | vsides | box | border
- void: Ningún lado. Este es el valor por defecto.
- above: Sólo el borde superior.
- below: Sólo el borde inferior.
- hsides: Sólo los bordes superior e inferior.
- vsides: Sólo los lados derecho e izquierdo.
- lhs: Sólo el lado izquierdo.
- rhs: Sólo el lado derecho.
- box: Los cuatro lados.
- border: Los cuatro lados.

rules=”*valor*”
Este atributo especifica qué líneas de división aparecerán entre las celdas de una tabla. La representación de las líneas de división depende del navegador. Valores posibles:
- none: Sin líneas. Este es el valor predeterminado.
- rows: Las líneas se mostrarán entre filas solamente.
- cols: Las líneas se mostrarán entre columnas solamente.
- all: Las líneas se mostrarán entre todas las filas y columnas.

bgcolor=”código color”
Especifica un color de fondo.

Algunos navegadores soportan también los atributos bordercolor y background:
bordercolor=”código color”
Especifica un color de borde.

background=”URL”
Especifica una imagen de fondo, la cual se repetirá hasta llenar toda la tabla.

Atributos de filas **<tr>**:

height=”altura”
Altura de la fila (en píxeles o en porcentaje).

align=”alineación”
Alineación horizontal del contenido de las celdas de la fila (left, center o right).

valign=”alineación”
Indica la alineación vertical del contenido de las celdas de la fila. Puede ser top (alineación superior), middle (alineación media) o bottom (alineación inferior).

bgcolor=”código color”
Especifica un color de fondo.

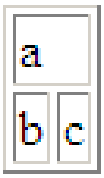
background=”URL”
Especifica una imagen de fondo, la cual se repetirá hasta llenar toda la fila.

bordercolor=”código color”
Especifica un color de borde.

Atributos de celdas **<td>**:

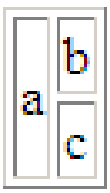
width=”ancho”
Anchura de la celda (en píxeles o en porcentaje). Normalmente al indicar la anchura de la celda, se indica también la anchura de la columna.

colspan=”n”
Hace que la celda se combine a través del número de columnas indicadas por n.
Así <td colspan=4>. Hace que la celda se una con las tres columnas siguientes, formando una celda que ocupará 4 columnas.




```
<table border="1">
    <tr>
        <td colspan="2"> a </td>
    </tr>
    <tr>
        <td> b </td>
        <td> c </td>
    </tr>
</table>
```

rowspan=”n”
Hace que la celda se combine a través del número de filas indicadas por n.
Así <TD rowspan=4>. Hace que la celda se una con las tres filas siguientes, formando una celda que ocupará 4 filas.




```
<table border="1">
    <tr>
        <td rowspan="2"> a </td>
        <td> b </td>
    </tr>
    <tr>
        <td> c </td>
    </tr>
```

</table>

align=”alineación”
Indica la alineación horizontal del contenido de la celda (left, center o right)

valign=”alineación”
Indica la alineación vertical del contenido de la celda. Puede ser top (alineación superior), middle (alineación media) o bottom (alineación inferior).

bgcolor=”código color”
Especifica un color de fondo.

background=”URL”
Especifica una imagen de fondo, la cual se repetirá hasta llenar toda la celda.

bordercolor=”código color”
Especifica un color de borde.

Ejemplo:

```
<html lang="es">
<head>
   <title>Ejemplo de tabla un poco más avanzada</title>
</head>

<body>

<h1>Tablas avanzadas</h1>

<table width="50%" border="1" cellspacing="3" cellpadding="2">
<tr>
   <td colspan="2" align="right">Dato 1</td>  <!--Dato1 ocupa dos columnas-->
   <td>Un texto cualquiera</td>
   <td rowspan="2">Dato 3</td> <!-- Dato3 ocupa dos filas -->
</tr>
<tr>
   <td>Dato 4</td>
   <td>Dato 4</td>
   <td align="center">Dato 5</td>
</tr>
</table>

</body>
</html>
```

### 3.4.12 Formularios

Los formularios nos permiten, dentro de una página Web, solicitar información al visitante para que después sea procesada por algún agente (p.ej., un servidor web, un servidor de correo, etc.).

En el formulario podremos solicitar diferentes datos, cada uno de los cuales quedará asociado a una variable. Los usuarios interaccionan con los formularios a través de los llamados campos o controles. El "nombre de control" de un control viene dado por su atributo *name*. El "campo de acción" o alcance del atributo *name* de un control contenido en un elemento FORM es el elemento FORM.

Una vez se hayan introducido los valores en los campos, el contenido de estos será enviado a la dirección (URL) donde tengamos el programa que pueda procesar las variables.

La declaración del formulario se pone entre las directivas **<form> y </form>**. En el interior de la declaración se indican los elementos (variables) de entrada.
Parámetros: *action, method y enctype.*

<**form action**=url
**method**=método de envío
**enctype**=método de codificación de la información
**name=**nombre del formulario
**autocomplete=**muestra entradas previas
**novalidate=**capacidad propia de validación
…..
</**form**>

**action =** "programa"
Indica la dirección URL del programa al que hay que enviar los datos, es decir, del programa que va a "tratar" las variables que se envíen con el formulario.

**method =** post / get
Indica el método según el que se transferirán las variables.

**enctype =**
Especifica el tipo de contenido usado para enviar el formulario al servidor (cuando el valor del atributo method sea "post"). Por defecto su valor es *"application/x-www-form-urlencoded"*.

**name=**
Este atributo da nombre al formulario de modo que se pueda hacer referencia a él desde hojas de estilo o scripts. Nota: este atributo ha sido incluido por motivos de compatibilidad con versiones anteriores. **Las aplicaciones deberían usar el atributo *id* para identificar elementos.**

**autocomplete= on|off**
El valor por defecto es **on**. Cuando es configurado como **off**, los valores <**input**> pertenecientes a este formulario tendrán la función de autocompletar desactivada, sin mostrar entradas previas como posibles valores. Puede ser implementado en el elemento <**form**> o en cualquier elemento <**input**> independiente.

**novalidate**

Una de las características de los formularios en HTML5 es la capacidad propia de validación. Los formularios son automáticamente validados. Para evitar este comportamiento podemos usar el atributo **novalidate.** Para lograr lo mismo para elementos **<input> e**specíficos, existe otro atributo llamado **formnovalidate.** Ambos atributos son booleanos, ningún valor tiene que ser especificado (su presencia es suficiente para activar su función)

El siguiente ejemplo muestra un formulario que va a ser procesado por el programa "usuarionuevo" cuando sea enviado. El formulario será enviado al programa usando el método HTTP "post".

```
<form action="http://algunsitio.com/prog/usuarionuevo" method="post">
...contenidos del formulario...
</form>
```

Para probar que los datos del formulario se están recogiendo bien, podemos utilizar el siguiente programa:

```
<form method="get" action="http://salonso.etsisi.upm.es/curso/procesa.php">
```

Un formulario puede contener cualquier otro elemento html (párrafos, listas, tablas, etc.) además de controles de formulario. De hecho, si el formulario es muy complejo, se suelen usar tablas para organizar los campos o controles.

## Método de envío del formulario

El atributo method del elemento FORM especifica el método HTTP usado para enviar el formulario al agente procesador. Este atributo puede tener dos valores:

- **get:** Con el método HTTP "get", el conjunto de datos del formulario se agrega al URL especificado por el atributo action (con un signo de interrogación "?" como separador) y este nuevo URL se envía al agente procesador.

- **post:** Con el método HTTP "post", el conjunto de datos del formulario se incluye en el cuerpo del formulario y se envía al agente procesador.

El método "get" debería usarse cuando el formulario es idempotente (es decir, cuando no tiene efectos secundarios). Muchas búsquedas en bases de datos no tienen efectos secundarios visibles y constituyen aplicaciones ideales del método "get".
Si el servicio asociado con el procesamiento de un formulario causa efectos secundarios (por ejemplo, si el formulario modifica una base de datos o la suscripción a un servicio), debería usarse el método "post".
Además hay que tener en cuenta que **con el método "get" se realiza una única conexión con el servidor y los datos que se envíen se pueden visualizar en el navegador, mientras que con el método "post" se realizan dos conexiones distintas: una para enviar el formulario con la petición y otra para enviar los datos**, de forma que estos últimos no se muestran al usuario.

Pensemos por ejemplo en un formulario con usuario y contraseña; aunque protejamos la visualización en pantalla de la contraseña, con el método “get” ésta se mostraría en la url que aparece en el navegador, mientras que con el método “post” no se podría ver ningún dato.

Nota: El método "get" restringe los valores del conjunto de datos del formulario a caracteres ASCII. Sólo el método "post" (con enctype="multipart/form-data") cubre el conjunto de caracteres [ISO10646] completo.

**Procesamiento de los datos del formulario**

Cuando el usuario envía un formulario (p.ej., activando un botón de envío), el agente de usuario (el navegador) lo procesa del siguiente modo.

**Paso uno:** Identificar los controles con éxito, es decir, los elementos que tienen valores y son válidos para su envío.

**Paso dos:** Construir el conjunto de datos del formulario.
Un conjunto de datos del formulario es una secuencia de parejas nombre de control/valor actual construida a partir de los elementos con éxito.

**Paso tres:** Codificar el conjunto de datos del formulario.
El conjunto de datos del formulario se codifica a continuación de acuerdo con el tipo de contenido especificado por el atributo *enctype* del elemento FORM.

**Paso cuatro:** Enviar el conjunto de datos del formulario codificado.
Finalmente, los datos codificados se envían al agente procesador designado por el atributo *action* usando el protocolo especificado por el atributo *method*.

**Campos de Entrada de datos**

Para la introducción de las variables se utiliza la directiva **<input>** (que no necesita cierre). Esta directiva tiene el parámetro **type** que indica el tipo de variable a introducir y **name** que indica el nombre que se le dará al campo. Cada tipo de variable tiene sus propios parámetros.

En HTML5 los nuevos tipos de variables no solo están especificando qué clase de entrada es esperada sino también diciéndole al navegador qué debe hacer con la información recibida. El navegador procesará los datos introducidos de acuerdo al valor del atributo type y validará la entrada o no.

- Entrada de texto en una sola línea (text input)
  **type =** text **name =** campo
  Indica que el campo a introducir será un texto. Sus parámetros son:

  **maxlength =** número
  Número máximo de caracteres a introducir en el campo.

  **size =** numero
  Tamaño en caracteres que se mostrará en pantalla.

  **value =** "texto"
  Valor inicial del campo. Normalmente será " ", o sea, vacío.

*<input type="text" name="nombre" placeholder="inserta tu nombre">*


- Entrada de una contraseña
  **type =** password **name =** campo
  Indica que el campo será una palabra de paso. Mostrará asteriscos en lugar de las letras escritas. Sus parámetros opcionales son los mismos que para text.

- Entrada de una dirección de email
  **type =** email **name =** campo
  Indica que el campo será una un email. El texto insertado será controlado por el navegador y validado como un email. Si la validación falla, el formulario no será enviado.

  *Introduce tu email: <input type="email" name="email">*


- Entrada de una búsqueda
  **type =** search **name =** campo
  El tipo search (búsqueda) no controla la entrada, es solo una indicación para los navegadores. Al detectar este tipo de campo algunos navegadores cambiarán el diseño del elemento para ofrecer al usuario un indicio de su propósito.

  *<input type="search" name="busqueda">*

- Entrada de una URL
  **type =** url **name =** campo
  Este tipo de campo trabaja exactamente igual que el tipo email pero es específico para direcciones web. Está destinado a recibir solo URLs absolutas y retornará un error si el valor es inválido.

- Entrada de un teléfono
  **type =** tel **name =** campo
  Este tipo de campo es para números telefónicos. A diferencia de los tipos email y url , el tipo tel no requiere ninguna sintaxis en particular. Es solo una indicación para el navegador en caso de que necesite hacer ajustes de acuerdo al dispositivo en el que la aplicación es ejecutada.

- Entrada de un número
  **type =** number **name =** campo
  Como su nombre lo indica, el tipo number es sólo válido cuando recibe una entrada numérica. Incorpora dos botones para incrementar o decrementar el valor del campo. El valor es almacenado en el atributo **value**. Existen algunos atributos nuevos que pueden ser útiles para este campo:
  - **min** El valor de este atributo determina el mínimo valor aceptado para el campo.
  - **max** El valor de este atributo determina el máximo valor aceptado para el campo.

- **step** El valor de este atributo determina el tamaño en el que el valor será incrementado o disminuido en cada paso. Por ejemplo, si declara un valor de 5 para step en un campo que tiene un valor mínimo de 0 y máximo de 10, el navegador no le permitirá especificar valores entre 0 y 5 o entre 5 y 10.

No es necesario especificar ambos atributos (min y max), y el valor por defecto para step es 1.


- Entrada a partir de una selección o rango
  **type =** range **name =** campo

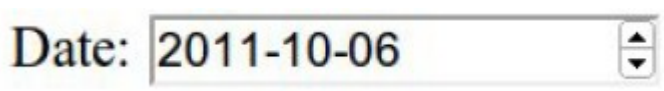
  Este tipo de campo hace que el navegador construya una nueva clase de control que no existía previamente. Este nuevo control le permite al usuario seleccionar un valor a partir de una serie de valores o rango. Normalmente es mostrado en pantalla como un puntero deslizable o un campo con flechas para seleccionar un valor entre los predeterminados, pero no existe un diseño estándar hasta el momento. Su valor es almacenado en el atributo **value.**

  El tipo range usa los atributos **min** y **max** estudiados previamente para configurar los límites del rango. También puede utilizar el atributo **step** para establecer el tamaño en el cual el valor del campo será incrementado o disminuido en cada paso.

- Entrada de una fecha
  **type =** date **name =** campo

  Este es otro tipo de campo que genera una nueva clase de control. En este caso fue incluido para ofrecer una mejor forma de ingresar una fecha. Algunos navegadores muestran en pantalla un calendario que aparece cada vez que el usuario hace clic sobre el campo. El calendario le permite al usuario seleccionar un día que será ingresado en el campo junto con el resto de la fecha. Un ejemplo de uso es cuando necesitamos proporcionar un método para seleccionar una fecha para un vuelo o la entrada a un espectáculo. Gracias al tipo date ahora es el navegador el que se encarga de construir un almanaque o las herramientas necesarias para facilitar el ingreso de este tipo de datos.

  La interface no fue declarada en la especificación. Cada navegador provee su propia interface y a veces adaptan el diseño al dispositivo en el cual la aplicación está siendo ejecutada. Normalmente el valor generado y esperado tiene la sintaxis año-mes-día




- Entrada de una semana
  **type =** week **name =** campo

  Este tipo de campo ofrece una interface similar a date, pero solo para seleccionar una semana completa. Normalmente el valor esperado tiene la sintaxis 2011-W50 donde 2011 es al año y 50 es el número de la semana.

- Entrada de un mes
  **type =** month **name =** campo

  Similar al tipo de campo previo, éste es específico para seleccionar meses. Normalmente el valor esperado tiene la sintaxis año-mes.


- Entrada de una hora
  **type =** time **name =** campo

  El tipo de campo time es similar a date , pero solo para la hora. Toma el formato de horas y minutos, pero su comportamiento depende de cada navegador en este momento. Normalmente el valor esperado tiene la sintaxis hora:minutos:segundos, pero también puede ser solo hora:minutos.


- Entrada de una fecha y hora con zona horaria
  **type =** datetime **name =** campo

  El tipo de campo datetime es para ingresar fecha y hora completa, incluyendo la zona horaria.


- Entrada de una fecha y hora
  **type =** datetime-local **name =** campo

  El tipo de campo datetime-local es como el tipo datetime sin la zona horaria.


- Entrada de un color
  **type =** color **name =** campo

  Además de los tipos de campo para fecha y hora existe otro tipo que provee una interface predefinida similar para seleccionar colores. Normalmente el valor esperado para este campo es un número hexadecimal, como #00FF00.

  Ninguna interface fue especificada como estándar en HTML5 para el tipo de campo color, pero es posible que una grilla con un conjunto básico de colores sea adoptada e incorporada en los navegadores

- Visualización de valores
  **Output**: se utiliza para visualizar valores, por ejemplo el valor de un campo **range.** Se suele utilizar junto con la propiedad **onformchange** para actualizar su valor:

  <output onformchange=”value = rango.value”>0</output>

- Entrada de texto en área de texto multilínea:
  Representa un campo de texto de múltiples líneas. Normalmente se utiliza para que se incluyan en él comentarios. La directiva usada es **<textarea> </textarea>.**

  Sus parámetros son:

**name =** campo
Nombre del campo.

**cols =** num
Número de columnas de texto visibles.

**rows =** num
Número de filas de texto visibles.

- Casillas de verificación (*CheckBoxes*)
  **type =** checkbox **name =** campo
  Permite definir una casilla de verificación. La casilla puede estar marcada o no. El campo se elegirá marcando una casilla. Se permite marcar varias casillas. Las casillas que no se marcan, no se envían. Los valores de las casillas serán indicados por:

  **value =** "valor"
  Este es el valor que se envía cuando se selecciona.

  **checked**
  Indica si la casilla aparecerá marcada por defecto.

- Radiobotones (*Radio Buttons*)
  **type =** radio **name =** campo
  Permite definir una casilla de elección. El campo se elegirá marcando una casilla. Sólo se permite marcar una de las casillas. Los valores de las casillas serán indicados por:
  **value =** "valor"
  **checked**

**IMPORTANTE:**
**Para que varios radio button pertenezcan al mismo grupo deben tener el mismo nombre.**

- Controles ocultos (*hidden controls*)
  **type =** hidden **name =** campo
  El usuario no puede modificar su valor, ya que el campo no es visible se manda siempre con el valor indicado por el parámetro. Sirve para enviar información al servidor sin que el usuario la vea.

  **value =** "valor"

  Los campos ocultos sirven para que podamos recibir información de la página, p.e. para saber si los datos vienen de una página web alojada en un servidor o de otra alojada en otro servidor diferente.

- **type = submit**
  Representa un botón de envío. Al pulsar este botón la información de todos los campos se envía al programa indicado en **<form>** (URL indicada en el apartado **action** de la etiqueta **form**). Tiene el parámetro **value =** "texto" que indica el texto que aparecerá en el botón.

<input type=“submit” value=”texto del botón”/>

- **type = reset**
  Representa un botón de reinicialización. Al pulsar este botón se borra el contenido de todos los campos y se recuperan sus valores por defecto. El parámetro **value =** "texto" indica el texto que aparecerá en el botón.

  <input type="reset" value="texto del botón"/>

- **type = button** **name** = campo **value** = título del botón
  Permite definir un botón genérico al que se puede asociar una acción utilizando algún lenguaje de script en el cliente. Lo veremos más adelante durante el curso.

  <input type="button" value="Pulsa aquí …" onclick="alert('Hola mundo')">

  *Nota: en este ejemplo vemos que en una línea HTML en la que se aniden las comillas, estas deben alternarse, de forma que usemos las dobles y las simples para que se distinga perfectamente donde empieza y termina la cadena de caracteres.*

- Selección de ficheros (*file select*)

  **type = file** **name** = campo
  Permite definir un control para enviar ficheros al servidor junto con los datos del formulario. El fichero se enviará al servidor para ser tratado allí. Los usuarios deben especificar la ruta local del archivo como contenido del control. Para ayudar con esto, los navegadores usualmente agregan un botón que, cuando es presionado, abre un navegador de archivos que permite a los usuarios elegir el archivo visual y fácilmente. Para que la subida de archivos sea satisfactoria, el atributo enctype debe ser "multipart/form-data" y method = "post".

  Fichero a enviar: <input type="file" name="losficheros" size="10" />

- **type = image** **src**= "fichero.imagen"
  Permite crear un botón *submit* con una imagen, aunque no se suele usar.

  <input type="image" src="imagen.gif">

## Campos de Selección (Menús desplegables)

Este tipo de campos despliegan una lista de opciones, entre las que debemos escoger una o varias. Se utiliza para ellos la directiva **<select> </select>**.
Sus parámetros son:

**name =** campo
Nombre del campo

**multiple**
Permite seleccionar más de un valor para el campo (pulsando la tecla CONTROL + distintas opciones).

**disabled**
La lista solo se visualiza. El usuario no puede seleccionar.

**size =** valor
Definimos el número de opciones visibles en cada momento.

Las diferentes opciones de la lista se indican con la directiva **<option>**. Sus parámetros son:

**value =** valor
Valor a enviar si se selecciona el elemento. Si no está se devuelve el texto de la opción.

**selected**
La opción aparece seleccionada.

**disabled**
La opción no se puede seleccionar

El formato de SELECT es:

```
<select name="mimenu" size=1 >
        <option value="valor"> texto </option>
        .....
</select>
```

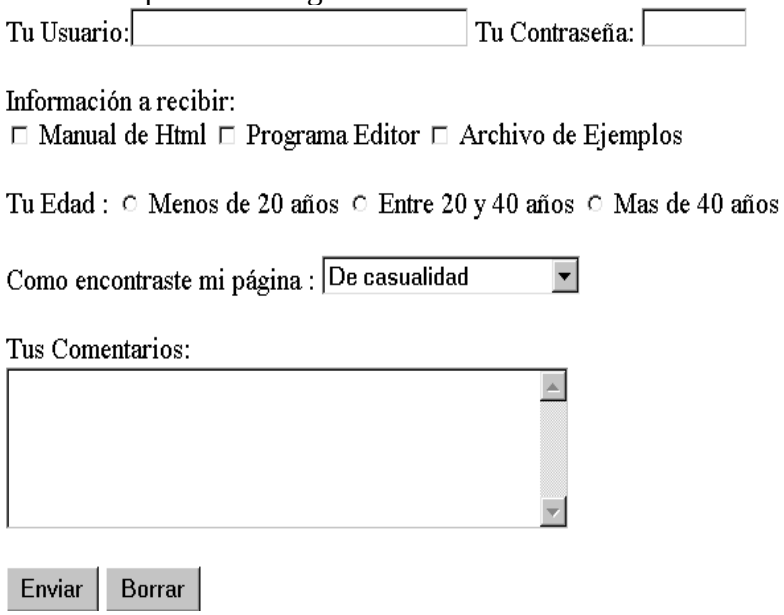
Es posible agrupar lógicamente las opciones de un elemento SELECT mediante el elemento **<optgroup>**. Esto es particularmente útil cuando el usuario debe elegir entre una larga lista de opciones; es más fácil apreciar y recordar grupos de opciones relacionadas que una larga lista de opciones sueltas. Entre la etiqueta inicial **<optgroup>** y la final **</optgroup>** se incluyen las opciones del grupo. El atributo ***label*** del elemento OPTGROUP específica el rótulo del grupo de opciones.

Ejemplo:

```
<select name="pais" size="8">
<optgroup label="Europa"> <!-- Primer grupo de opciones: Europa -->
        <option value="es" selected>España</option>
        <option value="fr">Francia</option>
        <option value="dt">Alemania</option>
</optgroup> <!-- Cierre del grupo "Europa"-->
<optgroup label="Asia"><!-- Segundo grupo de opciones: Asia -->
        <option value="jp">Japón</option>
        <option value="ch">China</option>
        <option value="in">India</option>
</optgroup> <!-- Cierre del grupo "Asia"-->
</select>
```

**Veamos un ejemplo completo:**

El siguiente código muestra en pantalla el siguiente formulario:




```
<form action = "mailto: pepita@gmail.com" method = post enctype="text/plain">

Tu Usuario:<input type = text name = nombre size = 30 />

Tu Contrase&ntildea: <input type = password name = clave size = 8 />

<p>Informaci&oacute;n a recibir:<br />
<input type = checkbox name = archivo value = "Html" /> Manual de Html
<input type = checkbox name = archivo value = "Editor" /> Programa Editor
<input type = checkbox name = archivo value = "Ejemplo" /> Archivo de Ejemplos </p>

<p>Tu Edad :
<input type = radio name = edad value = "-20" /> Menos de 20 a&ntildeos
<input type = radio name = edad value = "20-40" /> Entre 20 y 40 a&ntildeos
<input type = radio name = edad value = "+40" /> M&acutes de 40 a&ntildeos  </p>

<p><input type = hidden name = lugar value = "p&aacutegina personal" /> </p>

Como encontraste mi p&aacutegina :
<select name = donde >
<option>De casualidad
<option>Por el buscador Google
<option>Por el buscador Yahoo
<option>Me la comentaron
</select>

<p>Tus Comentarios: </p>
<br />
<textarea name = comentario rows = 5 cols = 40></textarea>

<p>
<input type = submit value = "Enviar" />
<input type = reset value = "Borrar" />
</p>
</form>
```

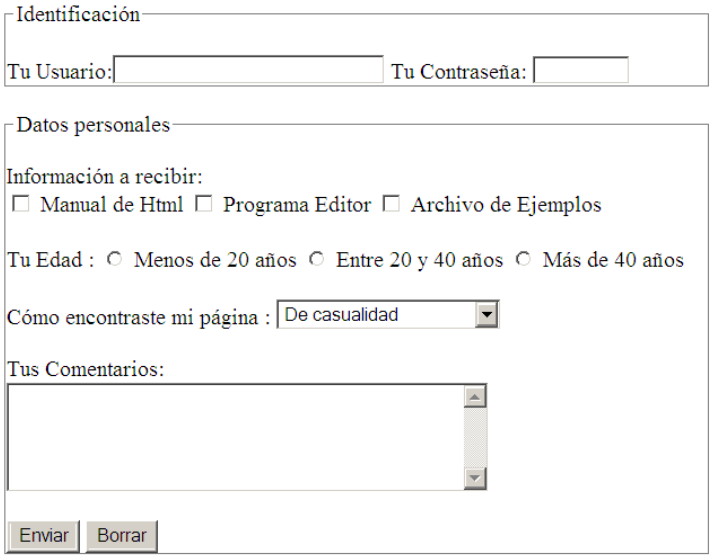
*Nota:* hasta HTML 4 lo más habitual era incluir código script (p.e. javascript) para validar los campos que el usuario ha introducido en nuestro formulario. Con HTML 5 se incorporó la capacidad propia de validación.

## Añadir estructura a los formularios: los elementos FIELDSET y LEGEND

El elemento FIELDSET (grupo de campos) permite a los autores agrupar temáticamente controles y rótulos relacionados.

Gracias al agrupamiento de controles es más fácil para los usuarios entender su propósito y al mismo tiempo se facilita la navegación con agentes de usuario visuales y la navegación por voz para agentes de usuario basados en voz.

El elemento LEGEND permite a los autores asignar un título a un FIELDSET. La leyenda mejora la accesibilidad cuando el FIELDSET no se representa visualmente.




```
<html lang="es">
<head>
<title>Ejemplo de formulario con mailto</title>
</head>

<body>
<form action = "mailto:pepita@gmail.com" method = post>
<fieldset>
  <legend>Identificaci&oacuten</legend>
  <p>Tu Usuario:<input type = text name = nombre size = 30 />
  Tu Contrase&ntildea:<input type = password name = clave size = 8 /> </p>
</fieldset>
<br />
<fieldset>
  <legend>Datos personales</legend>
  <p>Informaci&oacute;n a recibir:<br/>
  <input type = checkbox name = archivo value = "Html" /> Manual de Html
  <input type = checkbox name = archivo value = "Editor" /> Programa Editor
  <input type = checkbox name = archivo value = "Ejemplo" /> Archivo de Ejemplos </p>
```

```
<p>Tu Edad :
<input type = radio name = edad value = "-20" /> Menos de 20 a&ntildeos
<input type = radio name = edad value = "20-40" /> Entre 20 y 40 a&ntildeos
<input type = radio name = edad value = "+40" /> M&aacutes de 40 a&ntildeos  </p>

<p><input type = hidden name = lugar value = "p&aacutegina personal" /> </p>

C&oacutemo encontraste mi p&aacutegina :
<select name = donde >
        <option>De casualidad
        <option>Por el buscador Google
        <option>Por el buscador Yahoo
        <option>Me la comentaron
</select>

<p>Tus Comentarios:
<br /><textarea name = comentario rows = 5 cols = 40></textarea> </p>

<p><input type = submit value = "Enviar" />
<input type = reset value = "Borrar" /> </p>
</fieldset>
</form>
</body>
</html>
```

**Navegación con tabulador: TABINDEX**

Este atributo especifica la posición del elemento actual dentro del orden de tabulación del documento actual. Este valor debe ser un número entre 0 y 32767.

Los siguientes elementos soportan el atributo tabindex:

a, area, button, input, object, select y textarea.

Ejemplo:

```
<input type=text name=nombre size=30 tabindex=”1” />
```

**Atributos comunes para los elementos de formularios**

Algunos tipos de campo requieren de la ayuda de atributos, como los anteriormente estudiados: min, max y step. Otros tipos de campo requieren la asistencia de atributos para mejorar su rendimiento o determinar su importancia en el proceso de validación. Ya vimos algunos de ellos, como novalidate para evitar que el formulario completo sea validado o formnovalidate para hacer lo mismo con elementos individuales. El atributo autocomplete, también estudiado anteriormente, provee medidas de seguridad adicionales para el formulario completo o elementos individuales. Veamos algunos más.

- Atributo **placeholder**

  Especialmente en tipos de campo search, pero también en entradas de texto normales, el atributo placeholder representa una sugerencia corta, una palabra o frase provista para ayudar al usuario a ingresar la información correcta. El valor de este atributo es presentado en pantalla por los navegadores dentro del campo, como una marca de agua que desaparece cuando el elemento es enfocado.

  <input type="search" name="busqueda" id="busqueda" **placeholder**="escriba su búsqueda">

  Otro ejemplo:

```
<label for="referer">Nombre</label>
<input id="referer" name="referer" type="text"
                    placeholder="Escribe tu nombre completo" />
```

  Obteniendo como resultado:


- Atributo **required**

  Este atributo booleano no dejará que el formulario sea enviado si el campo se encuentra vacío. Por ejemplo, cuando usamos el tipo email para recibir una dirección de email, el navegador comprueba si la entrada es un email válido o no, pero validará la entrada si el campo está vacío. Cuando el atributo required es incluido, la entrada será válida sólo si se cumplen las dos condiciones: que el campo no esté vacío y que el valor ingresado esté de acuerdo con los requisitos del tipo de campo.

  <input type="email" name="miemail" id="miemail" **required**>

- Atributo **multiple**

  El atributo **multiple** es otro atributo booleano que puede ser usado en algunos tipos de campo (por ejemplo, email o file) para permitir introducir entradas múltiples en el mismo campo.
  Los valores insertados deben estar separados por coma para ser válidos.

  <input type="email" name="miemail" id="miemail" **multiple**> permite la inserción de múltiples valores separados por coma, y cada uno de ellos sera validado por el navegador como una dirección de email.

- Atributo **autofocus**

  Esta es una función que muchos desarrolladores aplicaban anteriormente utilizando el método focus() de Javascript. Este método era efectivo pero forzaba el foco sobre el elemento seleccionado, incluso cuando el usuario ya se encontraba posicionado en otro diferente. Este comportamiento era irritante pero difícil de evitar hasta ahora. El atributo **autofocus** enfocará la página web sobre el elemento seleccionado pero considerando la situación actual. No moverá el foco cuando ya haya sido establecido por el usuario sobre otro elemento.

  <input type="search" name="busqueda" id="busqueda" **autofocus**>

- Atributo **pattern**

  El atributo **pattern** es para propósitos de validación. Usa expresiones regulares para personalizar reglas de validación. Algunos de los tipos de campo ya estudiados validan cadenas de texto específicas, pero no permiten hacer validaciones personalizadas, como por ejemplo un código postal que consiste en 5 números. No existe ningún tipo de campo predeterminado para esta clase de entrada. El atributo **pattern** nos permite crear nuestro propio tipo de campo para controlar esta clase de valores no ordinarios. Puede incluso incluir un atributo title para personalizar mensajes de error.

  <input **pattern**="[0-9]{5}" name="codigopostal" id="codigopostal" title=”inserte los 5 números de su código postal”>

  IMPORTANTE: Expresiones regulares son un tema complejo. Para obtener información adicional al respecto ir a:
  http://es.wikipedia.org/wiki/Expresi%C3%B3n_regular

- Atributo **form**

  El atributo **form** es una adición útil que nos permite declarar elementos para un formulario fuera del ámbito de las etiquetas <form>. Hasta ahora, para construir un formulario teníamos que escribir las etiquetas <form> de apertura y cierre y luego declarar cada elemento del formulario entre ellas. A partir de HTML5 podemos insertar los elementos en cualquier parte del código y luego hacer referencia al formulario que pertenecen usando su nombre y el atributo **form**:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Formularios</title>
</head>
<body>
  <nav>
      <input type="search" name="busqueda" id="busqueda"
                          form="formulario">
  </nav>
  <section>
    <form name="formulario" id="formulario" method="get">
      <input type="text" name="nombre" id="nombre">
      <input type="submit" value="Enviar">
    </form>
  </section>
</body>
</html>
```

- Atributo **list**

  Usando el atributo **list** con un elemento <input> podemos especificar una lista de opciones. Esto permite al usuario seleccionar un valor de la lista o escribir uno que no esté en ella (este tipo de elemento se suele llamar Combo Boxes). Los elementos de la lista se deben de indicar utilizando, **datalist,** el cual simplemente nos permite crear una lista de valores.

El elemento <datalist> es un elemento específico de formularios usado para construir una lista de ítems que luego, con el atributo **list**, será usada como sugerencia en un campo del formulario.

```
<datalist id="informacion">
  <option value=”123123123” label=”Teléfono 1”>
  <option value=”456456456” label=”Teléfono 2”>
</datalist>
```

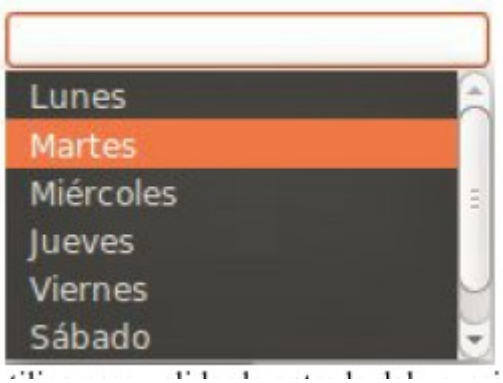
Este elemento utiliza el elemento <option> en su interior para crear la lista de datos a sugerir. Con la lista ya declarada, lo único que resta es referenciarla desde un elemento <input> usando el atributo list:

```
<input type="tel" name="telefono" id="telefono" list="informacion">
```

En este caso mostrará posibles valores para que el usuario elija.

Ejemplo:

```
<input type=”text” name=”diasdelasemana” id=”diasdelasemana” list=”dias”/>
<datalist id=”dias”>
    <option value=”Lunes” />
    <option value=”Martes” />
    <option value=”Miércoles” />
    <option value=”Jueves” />
    <option value=”Viernes” />
    <option value=”Sábado” />
    <option value=”Domingo” />
</datalist>
```



## 4. Páginas estáticas vs páginas dinámicas

Como hemos visto, los documentos HTML son de carácter estático, inmutables con el paso del tiempo. La página se carga, y ahí termina la historia. Tenemos ante nosotros la información que buscábamos, pero no podemos interactuar con ella. Solución a este problema: los lenguajes de script.

Los lenguajes de script permiten incluir “programación” en las páginas web.

**Un lenguaje de script es un pequeño lenguaje de programación cuyo código se inserta dentro del documento HTML.**

Podemos distinguir por tanto dos tipos de páginas web:

- **Estáticas**: sin movimiento y sin funcionalidades más allá de los enlaces. Construidas con el lenguaje html, que no permite grandes florituras para crear efectos ni funcionalidades más allá de los enlaces. Estas páginas son muy sencillas de crear, aunque ofrecen pocas ventajas tanto a los desarrolladores como a los visitantes, ya que sólo se pueden presentar textos planos acompañados de imágenes y a lo sumo contenidos multimedia.
- **Dinámicas**: tienen efectos especiales y podemos interactuar con ellas. Para crearlas es necesario utilizar otros lenguajes de programación, aparte del simple HTML.

En realidad HTML no es lenguaje de programación sino, más bien, un lenguaje descriptivo que tiene como objeto dar formato al texto y las imágenes que pretendemos visualizar en el navegador.

A partir de este lenguaje somos capaces de introducir enlaces, seleccionar el tamaño del texto o intercalar imágenes, todo esto de una manera prefijada. El lenguaje HTML no permite el realizar un simple cálculo matemático o crear una página de la nada a partir de una base de datos. A decir verdad, el HTML, aunque muy útil a pequeña escala, resulta bastante limitado a la hora de concebir grandes sitios (*web sites*) o portales.

Es esta deficiencia del HTML la que ha hecho necesario el empleo de otros lenguajes accesorios mucho más versátiles y de un aprendizaje relativamente más complicado.

Supongamos que hemos decidido realizar un portal de televisión donde una de las informaciones principales a proveer podría ser la programación semanal. Efectivamente, esta información suele ser dada por las televisiones con meses de antelación y podría ser muy fácilmente almacenada en una base de datos.

Si trabajásemos con páginas HTML, tendríamos que construir una página independiente para cada semana en la cual introduciríamos "a mano" cada uno de los programas de cada una de las cadenas. Asimismo, cada semana nos tendríamos que acordar de descolgar la página de la semana pasada y colgar la de la anterior.

Todo esto podría ser fácilmente resuelto mediante páginas dinámicas. En este caso, lo que haríamos sería crear un programa (sólo uno) que se encargaría de recoger de la base de datos de la programación aquellos programas que son retransmitidos en las fechas que nos interesan y de confeccionar una página donde aparecerían ordenados por cadena y por hora de retransmisión. De este modo, podemos automatizar un proceso y desentendernos de un aspecto de la página por unos meses.

## 5. Lenguajes de scripts

Como ya hemos visto, el navegador es una aplicación capaz de interpretar las órdenes recibidas en forma de código HTML fundamentalmente y convertirlas en las páginas web que son el resultado de dicha orden. Cuando nosotros pinchamos sobre un enlace hipertexto, en realidad lo que pasa es que establecemos una petición de un archivo HTML residente en el servidor (un ordenador que se encuentra continuamente conectado a la red) el cual es enviado e interpretado por nuestro navegador (el cliente).

Existen dos tipos de lenguajes de script:

- **lenguajes de lado cliente** que son aquellos que **pueden ser directamente "digeridos" por el navegador y no necesitan un pretratamiento.** Entre los lenguajes que se ejecutan en el lado del cliente se encuentran Java y JavaScript los cuales son simplemente incluidos en el código HTML.

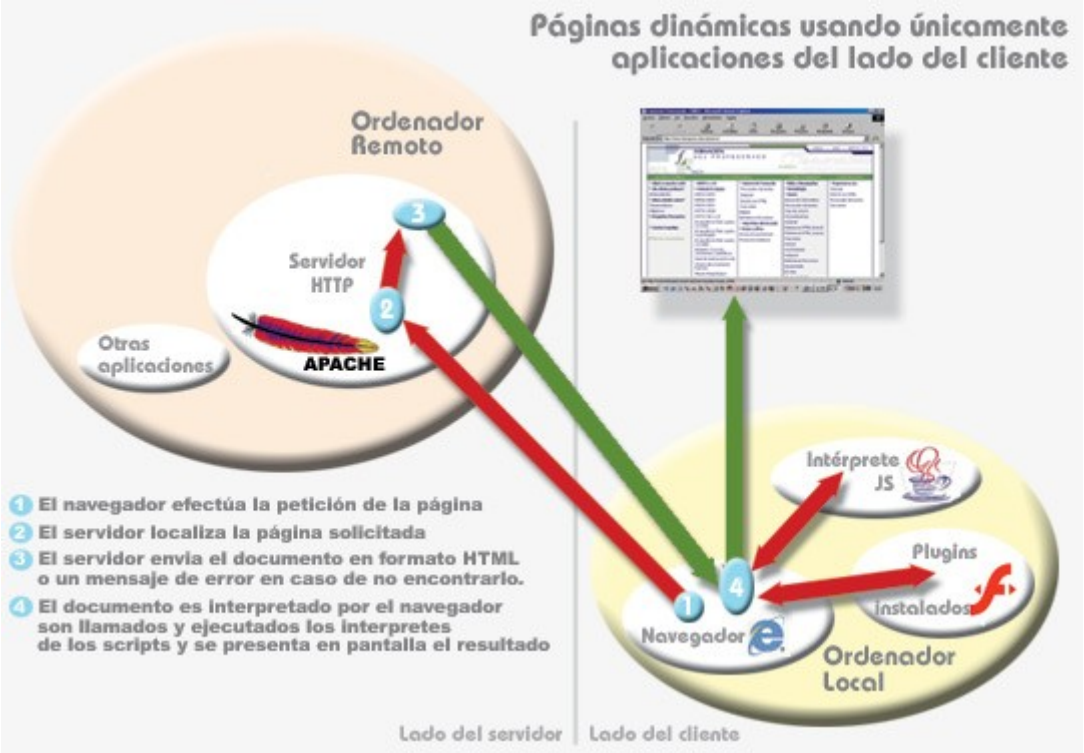
- **lenguajes de lado servidor** que son aquellos **lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él**. Los más importantes son php y asp.

Cada uno de estos tipos tiene sus ventajas y sus inconvenientes:

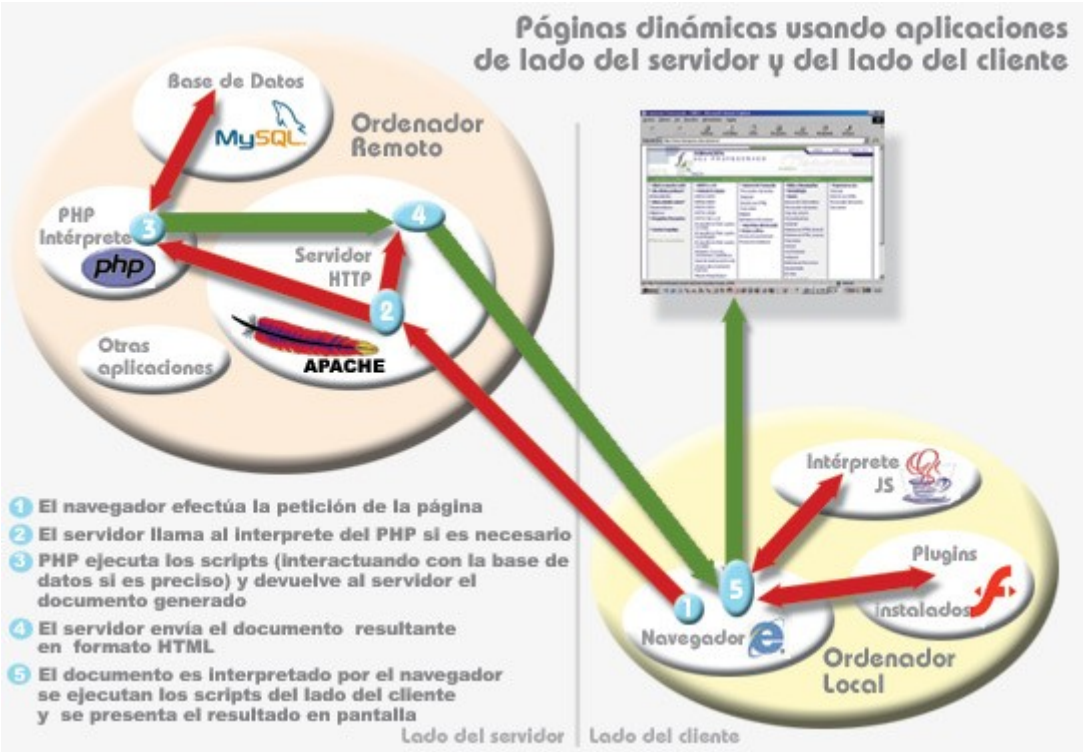
- **un lenguaje de lado cliente es totalmente independiente del servidor**, lo cual permite que la página pueda ser albergada en cualquier sitio sin necesidad de pagar más ya que, por regla general, los servidores que aceptan páginas con scripts de lado servidor son en su mayoría de pago o sus prestaciones son muy limitadas.

- **un lenguaje de lado servidor es independiente del cliente** por lo que es mucho menos rígido respecto al cambio de un navegador a otro o respecto a las versiones del mismo.

La siguiente imagen muestra el proceso utilizado cuando se trabaja con aplicaciones del lado del cliente:




La siguiente imagen muestra el proceso utilizado cuando se trabaja con aplicaciones del lado del cliente y también del servidor:



## 6. Bibliografía


- Recomendación HTML5: https://www.w3.org/TR/2014/REC-html5-20141028/
- Documento del W3C de referencia de HTML (no normativa): http://www.w3.org/TR/html-markup/
- *Recomendación XHTML 1.1 (Second Edition):* http://www.w3.org/TR/2002/REC-xhtml1-20020801/
- *Libros:*

  **El Gran Libro De HTML5, CSS3 Y Javascript. Juan Diego Gauchat**
  HTML5 y CSS3. Dpto. de ciencias de la computación e IA.
- *Otros sitios de interés:*

  http://www.w3.org/html/wiki/Specifications
  http://www.sidar.org/recur/desdi/traduc/es/index.php#especicss
  http://www.jorgesanchez.net/web/html.pdf
  http://www.webestilo.com/html/cap1a.phtml
  http://www.publispain.com/supertutoriales/diseno/html/cursos/7/
  http://es.html.net/tutorials/html
  http://www.adelat.org/media/docum/nuke_publico/paginas_estaticas_vs_dinamicas.html
  http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html