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 (Microsoft Edge), 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 (Uniform Resource Locator Localizador Uniforme de Recursos –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. Algunos son entre otros: HTML-Kit y NOTEPAD++ (para Windows) y, Geany (para Linux y MacOS).

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 más sofisticadas 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.

URI = URL + URN

<Identificador de Recursos Uniforme = Localizador de Recursos Uniforme + Nombre de Recursos Uniforme>

URNs were originally conceived to be part of a three-part information architecture for the Internet, along with Uniform Resource Locators (URLs) and Uniform Resource Characteristics (URCs), a metadata framework. As described in the 1994 RFC 1737,[1], and later in the 1997 RFC 2141 [2], URNs were distinguished from URLs, which identify resources by specifying their locations in the context of a particular access protocol, such as HTTP or FTP. In contrast, URNs were conceived as persistent, location-independent identifiers assigned within defined namespaces, typically by an authority responsible for the namespace, so that they are globally unique and persistent over long periods of time, even after the resource which they identify ceases to exist or becomes unavailable.[3]

URCs never progressed past the conceptual stage,[4] and other technologies such as the Resource Description Framework later took their place. Since RFC 3986[5] in 2005, use of the terms "Uniform Resource Name" and "Uniform Resource Locator" has been deprecated in technical standards in favor of the term Uniform Resource Identifier (URI), which encompasses both, a view proposed in 2001 by a joint working group between the World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF).

Examples [edit]

URN	corresponds to
urn:isbn:0451450523	The 1968 book *The Last Unicorn*, identified by its book number.
urn:isan:0000-0000-9E59-0000-O-0000-0000-2	The 2002 film *Spider-Man*, identified by its audiovisual number.
urn:ISSN:0167-6423	The scientific journal *Science of Computer Programming*, identified by its serial number.
urn:ietf:rfc:2648	The IETF's RFC 2648.
urn:mpeg:mpeg7:schema:2001	The default namespace rules for MPEG-7 video metadata.
urn:oid:2.16.840	The OID for the United States.

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]***

Un HTTP URL combina en una dirección simple los cuatro elementos básicos de información necesarios para recuperar un recurso desde cualquier parte en Internet:

- El protocolo que se usa para comunicar o enviar datos.
- El anfitrión (servidor o *host*) con el que se comunica.
- El puerto de red en el servidor para conectarse.
- La ruta al recurso en el servidor (por ejemplo, su nombre de archivo).

Un URL típico puede ser del tipo:

`http://es.wikipedia.org:80/wiki/Special:Search?search=tren&go=Go`

Donde:

- `http` es el protocolo.
- `es.wikipedia.org` es el anfitrión.
- `80` es el número de puerto de red en el servidor (siendo 80 el valor por defecto para el protocolo HTTP, esta porción puede ser omitida por completo).
- `/wiki/Special:Search` es la ruta de recurso.
- `?search=tren&go=Go` es la cadena de búsqueda (parte opcional).

Tipos de Rutas:

- **Absolutas**: Especifican una ruta completa
  http://www.sobreasp.com/download/aspdoc.zip

- **Relativas**: La URL no está completa. Especifican una ruta relativa a la URL del documento. Las partes que faltan de la dirección las genera el navegador a partir de la ruta de la página en la que está el enlace. Esto es, asume la ruta hasta la carpeta/directorio en el que se encuentra la página.
  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.

Una entidad es una estructura que, mediante el uso de una codificación, nos permite representar un símbolo

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:

Entidad	Carácter	Entidad	Carácter
Ñ / ñ	Ñ / ñ	<	<
Á / á	Á / á	>	>
É / é	É / é	&	&
Í / í	Í / í		espacio blanco
Ó / ó	Ó / ó		Espacio ASCII
Ú / ú	Ú / ú		Tabulador ASCII
€	€		Avance de página ASCII
&quote;	"		Espacio de ancho cero
©	©
Retorno de carro

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 en el futuro así se 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> *Ejemplo:* <head>

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

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

Comentarios:

- Texto introducido en un documento que no aparece en el formato final.

  <!-- comentario -->

  Ejemplo: <!-- 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. Con HTML 5 la declaración del tipo de documento es más sencilla:

<!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. Se suele visualizar en la barra de título en los navegadores.

- **<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/">
```

- 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.

```
<meta charset="conjunto_caracteres">
```

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** antes del título.

```
<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>.

En HTML5 se han suprimido los atributos propios de la etiqueta <body>. Por lo tanto, para modificar la apariencia de la página hay que utilizar hojas de estilo. Por ejemplo, usando estilos en línea, para poner una imagen como fondo de la página se puede usar el atributo style y la propiedad background-image mientras que para poner un color de fondo a la página la propiedad que hay que usar es background-color.

Veamos unos ejemplo:

1) Cuerpo de documento con imagen de fondo

```
<body style=”background-image:url(imagenes/patito.gif)" >
        .....
</body>
```

2) Cuerpo de documento con color de fondo

```
<body style=”background-color:#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

Los 16 colores de la paleta VGA Standard también se pueden expresar en formato texto (nombre del color en inglés), aunque esto último no es aconsejable porque puede haber problemas con los algunos navegadores:

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

Ejemplo: <body style=”background-color:red”>

Veamos ahora el aspecto básico de la mínima página web:

```
<!DOCTYPE html>
<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 va el contenido del documento  -->
</body>
</html>
```

3.5 Elementos del cuerpo del documento

En HTML5 se han suprimido en muchas etiquetas los atributos que se utilizaban en versiones previas de HTML para especificar el aspecto de los contenidos en la página. A partir de HTML5 hay que utilizar hojas de estilo para indicar la apariencia de las páginas. En estos apuntes, como introducción a las hojas de estilo. se utiliza el atributo style junto con algunas propiedades propias de hojas de estilo para modificar el aspecto de las páginas.

3.5.1 Cabeceras <h1> … <h6>

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>
```

3.5.2 Bloques de texto

- ** **
  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
. La etiqueta de finalización no es obligatoria, aunque conviene ponerla.

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

- **
**

  Salto de línea. Fuerza que se parta una línea de texto independientemente del formato en que se esté trabajando.

```
<br/>
```

- **<pre>**

  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>
```

- ** **

  Permite agrupar varios bloques de texto en uno solo, heredando todos ellos las características de presentación dadas. Se trata de una etiqueta muy potente que demuestra su potencial cuando se usa con las hojas de estilo.

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

- ****

  Agrupación de varios elementos para establecer una presentación común para todos. No añade salto de línea.

```
<span>texto</span>
```

3.5.3 Etiquetas para citas

- **<q>**

  El contenido es una cita textual de una frase

```
<p>El consorcio W3C es: <q> una comunidad internacional que trabaja
para desarrollar estándares Web</q></p>
```

- **<blockquote>**

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

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

- **<abbr>**

Indica una forma abreviada (p.ej., WWW, HTTP, URI, Mass., etc.).

```
<abbr title="World Health Organization">WHO</abbr>
```

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

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

- **<cite>**
  Contiene una cita o una referencia a otras fuentes.

```
<p><cite>El grito </cite> de Edvard Munch. Pintado en 1893.</p>
```

- **<dfn>**
  Indica que aquí es donde se define el término encerrado

```
<p>El <dfn>HTML</dfn> es un lenguaje de marcado para hipertextos.</p>
```

3.5.4 Etiquetas para formato de textos

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

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

<i>	Letra inclinada (itálica o cursiva)
****	Indica énfasis (emphasis). Usualmente los navegadores usan la cursiva.
<b>	Letra en negrita. Nota: actualmente el uso de `<b>` se ha sustituido por `<strong>`
****	Indica un énfasis más fuerte.
<mark>	Representa texto resaltado con propósitos de *referencia*, es decir por su relevancia en otro contexto.
<small>	Representa un *comentario* que no es esencial para la comprensión del documento Disminuye el tamaño de la fuente. Es acumulativa
<time>	Define una fecha y/o hora.
<del>	El texto afectado ha sido eliminado en esta versión del documento

<ins>	El texto afectado ha sido añadido en esta versión del documento
<sup>	Superíndice
<sub>	Subíndice
<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.

**** y **** se usan para indicar énfasis añadiendo importancia al texto contenido. Frente a <i> y <b> que sólo indican que el texto debe estar en cursiva y negrita, respectivamente.

Ejemplos:

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

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

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>
```

La siguiente frase muestra varios tipos de texto:

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

3.5.5 Línea horizontal

- **<hr>**
  Línea horizontal.
  A partir de HTML5 sus atributos noshade, width, align y size han sido eliminados

```
<hr/>
```

3.5.6 Listas

HTML ofrece varios mecanismos para especificar listas de información. Existen 4 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.

Las listas pueden anidarse.

- Lista sin ordenar **<ul>**

  Lista de textos, cada entrada de la lista comienza por un carácter o marca de señalización (*bullet*).

  Los atributos de este tipo de listas han sido eliminados a partir de HTML5, así que las presentación se debe controlar mediante hojas de estilo. Para indicar el tipo de marcador que se desea utilizar para cada uno de los elementos de las lista, en lugar del atributo type, se utilizará la propiedades de list-style-type que podrá tomar uno de los siguientes valores: circle | disc | square | none

```
<ul style="list-style-type:valor">
      <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 type="tipo" start="numero" reversed>
      <li> Texto </li>
      <li> Texto </li>
      .....
</ol>
```

  La única lista que todavía guarda sus atributos en HTML5 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 son:

  - Atributo **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 indicar como valor: 1 (para número, opción por defecto), A (letras en mayúsculas), a (letras en minúsculas), i (números romanos en minúsculas), I (números romanos en mayúsculas).

  - Atributo **reversed**: Controla el orden ascendente o descendente de la numeración que por defecto es ascendente.

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

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

- Atributo **start**: Permite indicar que se 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>**

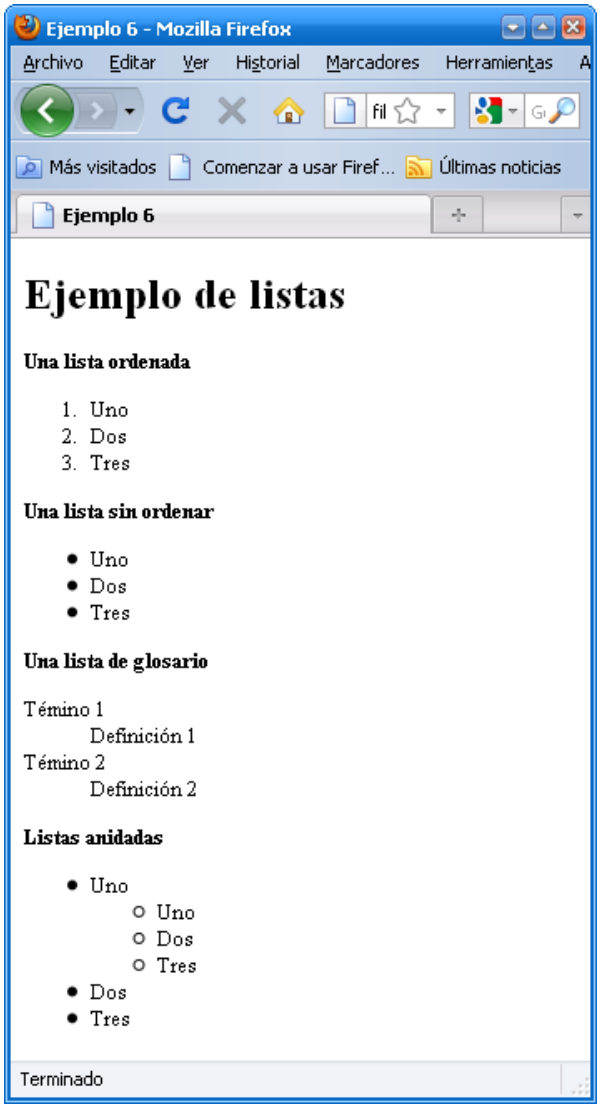
  Cada entrada en la lista tiene dos partes: el término que se define (encabezado por la etiqueta **<dt>**) y la definición (encabezada por la etiqueta **<dd>**).

```
<dl>
      <dt>Termino1</dt>

      <dd>Definicion1</dd>
      <dt>Termino2</dt>

      <dd>Definicion2</dd>
 .....
</dl>
```

Ejemplo: Salida del navegador y código fuente asociado.




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

<body>

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

3.5.7 Hiperenlaces o hipervínculos

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

```
<a href="url" target="destino">texto</a>
```

Algunos atributos de esta etiqueta son:

- **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).

  También se pueden especificar otros protocolos como https://, ftp://, mailto://, file://, etc y scripts (href="javascript:alert('Hola');")

  Ejemplo: *La Web de Microsoft*

- **target**="destino"

  Indica en qué parte de la ventana se muestra el enlace. Posibles valores son: _blank (para mostrar el hiperenlace en una nueva copia del navegador) | _self (para mostrarlo en la misma instancia del navegador)

  Ejemplo:

  ```
  <a href="https://www.w3schools.com" target="_blank">Visita W3Schools</a>
  ```

- **id**="nombre"

  A partir de HTML5 el atributo name ha sido eliminado y su funcionalidad es asumida por el atributo id

  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 id="marcador" >texto</a>
  ```

  Ejemplo:

  ```
  <!-- Establecer un marcador dentro de una pagina -->
  <a id="marca1"></a>

  <!-- En otro lugar del documento se pone un enlace a esa marca -->
  <a href="#marca1">Documentación pública</a>

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

Veamos un ejemplo de hipervínculos:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Ejemplo hipervínculos</title>
  <meta charset=”iso-8859-1">
</head>

<body>
  <a id="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/>.<br/>

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

3.5.8 Imágenes

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

```
<img  src=”url_imagen”    alt=”texto_alternativo”
      width=”pixels_ancho”    height=” pixels_alto”
      usemap=”mapa_asociado”    ismap   >
```

A partir de HTML5, la alineación de la imagen (atributo align), su borde (atributo border) y todo lo referente a su aspecto (atributos hspace y vspace) en la página web se indica con CSS

Algunos atributos de esta etiqueta son:

- **src**=”url”

  Es un parámetro obligatorio. Indica la URL donde se encuentra el fichero que contiene la imagen. Puede ser relativa o absoluta. 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”

  Texto alternativo que se presentará en la página 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.

  También es utilizado por las herramientas de accesibilidad para invidentes.

- **usemap**=”mapa”
  Indica que la imagen tiene asociado un mapa sensible que ha sido definido en el propio documento HTML. La sintaxis es igual a una referencia interna dentro del propio documento (por ejemplo #nombre) (ver apartado 3.5.9 “Mapas sensibles”)

- **ismap**
  Su presencia indica que la imagen es un mapa interactivo.

```
<img usemap=”#mapa” ismap />
```

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.

El color del borde de la imagen dependerá del color del texto en la página, párrafo en el que esté incluida la imagen.

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

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

3.5.9 Mapas sensibles

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

1) Definición de la estructura del mapa

```
<map name=”nombre_mapa”>
      <area shape=”forma” coords=”coordenadas” ( href=”url” | nohref ) >
      ....
      <area shape=”forma” coords=”coordenadas” ( href=”url” | nohref ) >
</map>
```

- **<map name**=”nombre_mapa”**>**
  Con la etiqueta <map> se define el mapa sensible. El atributo name es obligatorio y permite asignar un nombre al mapa sensible para su identificación.

- **<area shape**=”forma” **coords=”**coordenadas” **(href**=”url” **| nohref) >**
  La etiqueta <area> se utiliza para definir cada una de las zonas activas del mapa. Las zonas activas se definen con respecto a la forma que ocupan en la imagen.

Para acotar una zona activa se utilizan los siguientes atributos:

- **href=**”url”
  Hipervínculo asociado a la zona activa.
- **nohref**
  Indica que la zona activa no tiene ningún hipervínculo.
- **shape**
  Define la forma de la zona activa: rect (rectángulo), circle (círculo), poly (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.

  Para definir una zona rectangular hay que indicar la esquina superior izquierda del área y la esquina inferior derecha

  shape=”**rect**” coords="**left_x, top_y, right_x, bottom_y**"

  Para crear una zona cicular hay que indicar el centro de la circunferencia y su radio

  shape=”**circle**” coords="**center_x, center_y, radius**"

  Para definir un polígono hay que dar los diferentes puntos del polígono de forma ordenada (siguiendo el camino para formarlo).

  shape=”**poly**” coords="**x11,y11, x22,y22, x33,y33, ...**"

  Por ejemplo, si la imagen insertada tiene un ancho de 400 pixels y una altura de 300 pixels, 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) Aplicación del mapa definido a una imagen

Para ello, se utiliza el atributo usemap de la etiqueta img: en el que haremos referencia a la estructura de mapa a utilizar.

```
<img src=”url” usemap=”#nombre_mapa” >
```

Ejemplo: Definición de un mapa sensibles en una imagen de 300 x 250 pixels. Este mapa tendrá 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">
```

3.5.10 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>**

```
<table>
    <caption>Titulo</caption>
    <tr>
        <th | td colspan=”numero” rowspan=”numero”>
        </th | /td>
        ...
    </tr>

    <tr>
        ...
    </tr>

    ...

</table>
```

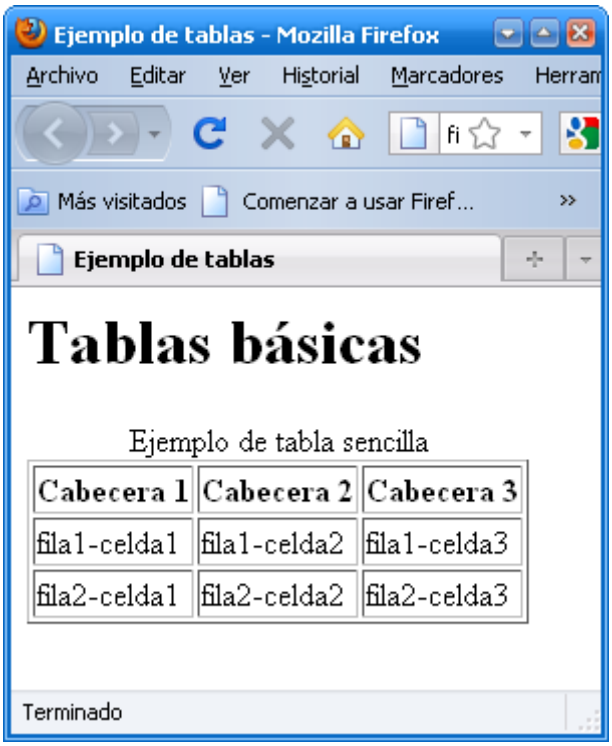
Nota: Cuando ponemos th | td *significa que los atributos valen tanto para las dos etiquetas*

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

Veamos un ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
 <title>Ejemplo de tablas</title>
</head>

<body>
    <h1>Tablas b&aacute;sicas</h1>
    <table>
        <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>
```



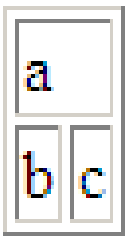
Ahora vamos a ver qué atributos pueden tener las tablas.

A partir HTML5 se han suprimido los atributos propios de las etiquetas <table>, <caption>,<tr>, <th> y <td> cuyo objetivo es modificar el aspecto de la tabla dentro de una página HMTL. Por lo tanto, para modificar la apariencia de una tabla hay que utilizar hojas de estilo.

Atributos de la etiqueta <td> que permiten combinar varias filas y columnas son:

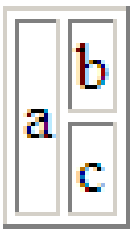
- **colspan**="n"
  Hace que la celda se combine a través del número de columnas indicadas por n.
  Así <td colspan="4"> hace que la celda se una con las tres columnas siguientes, formando una celda que ocupará 4 columnas.

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



Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset=”iso-8859-1">
    <title>Ejemplo de tabla un poco más avanzada</title>
</head>

<body>
    <h1>Tablas avanzadas</h1>

    <table >
        <tr>
         <td colspan="2">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>Dato 5</td>
        </tr>
    </table>
</body>
</html>
```

Tablas avanzadas

Dato 1		Un texto cualquiera	Dato 3
Dato 4	Dato 4	Dato 5	

3.5.11 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.

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 los datos.

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:

1. Identificar los controles con éxito. Es decir, los elementos que tienen valores y son válidos para su envío.

2. 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 controles con éxito (controles en los que se ha introducido un valor válido).

3. Codificar el conjunto de datos del formulario.
   A continuación, el conjunto de datos del formulario se codifica de acuerdo con el tipo de contenido especificado por el atributo enctype del elemento <form>.

4. 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 del elemento <form>.

La declaración del formulario se pone entre las directivas **<form> y </form>**. En el interior de la declaración se indican los controles (variables) de entrada.

```
<form action=”programa”
      method=”método_envío”
      enctype=”método_codificación”
      name=”nombre_formulario”
      autocomplete=”on|off”
      novalidate >

    <!-- controles -->

</form>
```

Atributos de <form> son:

- **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 HTTP según el que se transferirán los datos al agente procesador.
Este atributo puede tener dos valores:

- **get** indica que los datos enviados se adjuntarán en la barra de direcciones del cliente, al final de la url correspondiente y después de un signo de interrogación de cierre. Si se envía más de un dato, éstos irán separados por el símbolo &
- **post** indica que el método de envío no se hará a través de la url, sino formando parte del cuerpo de la petición.

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 éstos no se muestran al usuario.

Por esto, 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.

Mientras que 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.

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.

- **enctype**=”metodo”

  Especifica, cuando el atributo method es "post”, la forma en que se enviará al servidor la información contenida en el formulario. Por defecto su valor es application/x-www-form-urlencoded. Otros valores posibles son: text/plain y multipart/form-data (conjunto de caracteres ISO10646)

- **name**=”nombre_formulario”

  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>
```

El siguiente ejemplo muestra como enviar la información recogida en un formulario a una dirección de correo electrónico

```
<form action="mailto:usuario@correo.es" method="post" enctype="text/plain">
  ...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">
```

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

Campos de Entrada de datos

Para la captura de información 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 campo tiene sus propios parámetros.

```
<input type="tipo_dato" name="variable" ...>
```

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. ***Los nuevos tipos de campos introducidos a partir de HTML5 no solo especifican qué clase de entrada es esperada sino también le dicen 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 (**type=”**text”)
  Sus parámetros son:
  - **maxlength=**“numero”: 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.

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

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

- Entrada de una dirección de email (**type=”**email”)
  Indica que el campo será una dirección de correo electrónico. El texto insertado será controlado por el navegador y validado como una dirección de email. Si la validación falla, el formulario no será enviado.
  Ejemplo: * Introduce tu email: <input type="email" name="email"> *

- Entrada de una búsqueda (**type=”**search”)
  El tipo search (búsqueda) no controla la entrada, es sólo 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.

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

- Entrada de una URL (**type=”**url”)
  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”)
  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 sólo 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”)
  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 en HTML5 para este campo:

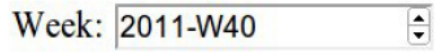
- **min** Indica el mínimo valor aceptado para el campo.
- **max** Indica el máximo valor aceptado para el campo.
- **step** Indica el tamaño en el cual el valor del campo será incrementado o disminuido en cada paso. El valor por defecto es 1
  Por ejemplo, si el atributo step declara un valor de 5 en un campo que tiene un valor mínimo de 0 y máximo de 10, el navegador no permitirá introducir valores entre 0 y 5 o entre 5 y 10.

- Entrada a partir de una selección o rango (**type=”**range”)
  Este 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 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.

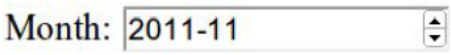
- Entrada de una fecha (**type=”**date”)
  Algunos navegadores muestran en pantalla un calendario que aparece cada vez que el usuario hace clic sobre el campo. El calendario le permite al usuario seleccionar un día que será ingresado en el campo.

  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”)
  Este tipo de campo ofrece una interface similar a date, pero sólo para seleccionar una semana completa. Normalmente el valor esperado tiene la sintaxis 2011-W40 donde 2011 es al año y 40 es el número de la semana.



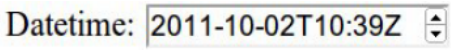
- Entrada de un mes **(type=”**month**”)**
  Similar al tipo de campo previo, éste es específico para seleccionar meses. Normalmente el valor esperado tiene la sintaxis año-mes.



- Entrada de una hora **(type=”**time**”)**
  El tipo de campo time es similar a date , pero solo para la hora. Toma el formato de horas y minutos.

Su comportamiento depende de cada navegador. 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**")**
  El tipo de campo datetime permite ingresar fecha y hora completa, incluyendo la zona horaria.



- Entrada de una fecha y hora **(type="**datetime-local**")**
  El tipo de campo datetime-local es como el tipo datetime sin la zona horaria.

- Entrada de un color **(type="**color**")**
  Proporciona una interface predefinida para seleccionar colores. Normalmente el valor esperado para este campo es un número hexadecimal, como #00FF00.
  Ninguna interface ha sido especificada como estándar en HTML5 para el tipo color.

- Selección de ficheros (**type="**file")
  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 method tiene que ser igual a "post" y el atributo enctype debe ser "multipart/form-data"

```
<form method="post" enctype="multipart/form-data">
  <div>
    <label for="image_uploads">Choose images to upload (PNG, JPG)</label>
    <input type="file" id="image_uploads" name="image_uploads" accept=".jpg, .jpeg, .png"
multiple>
  </div>
  <div class="preview">
    <p>No files currently selected for upload</p>
  </div>
  <div>
    <button>Submit</button>
  </div>
</form>
```

- Casillas de verificación (*CheckBoxes*) (**type="**checkbox")
  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.
  Atributos aplicables serían:
  - **value=**"valor" Valor que se envía cuando se selecciona el control. Se puede omitir ya que se puede saber el estado de una casilla verificando su presencia o ausencia entre los campos enviados.
  - **checked** Indica si la casilla aparecerá marcada por defecto

Ejemplo,

```
<p> Intereses:<br/>
   <input type="checkbox" name="cb-autos" value="gusta"> Autos<br/>
   <input type="checkbox" name="cb-deportes" value="gusta"> Deportes<br/>
   <input type="checkbox" name="cb-videojuegos" value="gusta"> Videojuegos
</p>
```

Para que el texto asociado a cada casilla responda también a los clics de ratón, se puede usar la etiqueta <label> aplicada al texto y al control. Como, en el siguiente ejemplo:

```
<p> Intereses:<br/>
   <label><input type="checkbox" name="cb-autos"> Autos</label><br/>
   <label><input type="checkbox" name="cb-deportes"> Deportes</label><br/>
   <label><input type="checkbox" name="cb-videojuegos"> Videojuegos</label>
</p>
```

- Radiobotones (*Radio Buttons*) (**type=”**radio”)
  Permite definir una casilla de elección. El campo se elegirá marcando una casilla. Sólo se permite marcar una de las casillas.
  Atributos aplicables serían:
  - **value=**"valor"Valor que se envía cuando se selecciona
  - **checked** Indica que la opción aparecerá marcada por defecto

  IMPORTANTE: Para que varios radiobutton pertenezcan al mismo grupo deben tener el mismo nombre.

```
<p> Empleo actual:<br/>
   <input type="radio" name="empleoactual" value="completo"> Tiempo completo<br/>
   <input type="radio" name="empleoactual" value="mediodia"> Medio día<br/>
   <input type="radio" name="empleoactual" value="sinempleo"> Sin empleo
</p>
```

- Controles ocultos (**type=”**hidden”)
  El usuario no puede modificar su valor, ya que el campo no es visible. Se manda siempre con el valor indicado por el parámetro value.

  Sirve para enviar información al servidor sin que el usuario la vea. También 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.

```
<input id="prodId" name="prodId" type="hidden" value="xm234jq">
```

- Entrada de texto en área de texto multilínea
  Representa un campo de texto de múltiples líneas. Normalmente se utiliza para introducir comentarios.

  La etiqueta usada es **<textarea>**

```
<textarea name=”variable” cols=”columnas” rows=”filas” > </textarea>
```

Algunos de sus atributos son:

- **name=**”variable” Nombre del campo
- **cols=**”num” Número de columnas de texto visibles
- **rows=**”num” Número de filas de texto visibles

Visualización de valores

La etiqueta **<output>** se utiliza para visualizar un valor o el resultado de un cálculo. Este cálculo puede estar basado en valores de campos de un formulario. No funciona en todos los navegadores.

Por ejemplo, para visualizar el valor de un campo llamado *rango*. Aquí se utiliza la propiedad **onformchange** para actualizar su valor.

```
<p>Valor de rango: <output onformchange=”value=rango.value”>0</output></p>
```

En este otro ejemplo, se muestra el resultado de una operación de potencia entre dos números que son entradas del formulario. La operación será llevada a cabo por un programa llamado mostrarPotencia del lado cliente definido en otro lugar. Los controles deben tener sus atributos **id** definidos para que el programa pueda recuperar sus valores y se puede usar el atributo **for** para reflejar los elementos o controles que tomaron parte en el cálculo.

```
<p>Base: <input type="number" id="base" value="0" min="0" max="30"></p>
<p>Exponente: <input type="number" id="exponente" value="0" min="0" max="30"></p>
<p><input type="button" value="Calcular" onclick="mostrarPotencia()"></p>
<p>Resultado: <output id="potencia" for="base exponente"></output></p>
```

Campos de Selección (Menús desplegables)

Despliegan una lista de opciones entre las que debemos escoger una o varias.

```
<select name=”nombre” size=”visibles” multiple disabled >
       <option value=”valor” selected disabled > texto </option>
       .....
</select>
```

Sus atributos de la etiqueta **<select>** son:

- **name=**”campo” Nombre del campo
- **multiple** Permite seleccionar más de un valor para el campo (pulsando la tecla CTRL y las distintas opciones que se quieren seleccionar)
- **disabled** La lista solo se visualiza. El usuario no puede seleccionar.
- **size=**”valor” Número de opciones visibles

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

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 (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 (Asia) -->
       <option value="jp">Japón</option>
       <option value="ch">China</option>
       <option value="in">India</option>
</optgroup> <!-- Cierre del grupo "Asia"-->
</select>
```

Botones

Se pueden crear botones con el elemento **<input>** (los botones se definen como tipos especiales de cadenas de entrada) o con el elemento **<button>**. Su funcionalidad es similar

- Botón de envío (**type=**"submit")
  Al pulsar este botón la información de todos los campos se envía al programa indicado en el atributo action de la etiqueta <form>.
  El texto que aparecerá en el botón se especifica con el atributo value.

```
<input type="submit" value="texto del botón">
```

- Botón de envío con imagen (**type=**"image")
  Permite crear un botón de envío con una imagen, aunque no se suele usar.
  Se usa un atributo src para indicar el fichero con la imagen.

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

- Botón de reinicialización **(type=**"reset")
  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.

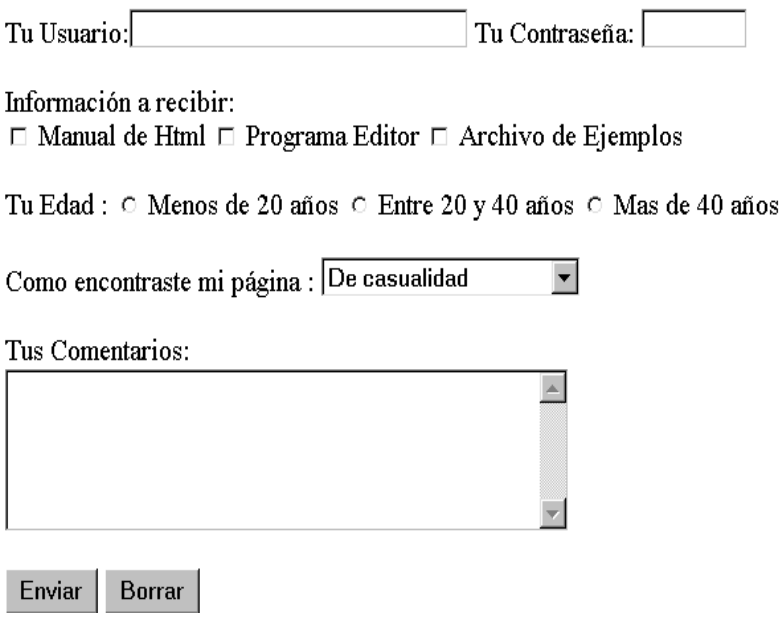
```
<input type="reset" value="texto del botón">
```

- Botón genérico (**type**="button"**)**
  Permite definir un botón al que se le puede asociar una acción utilizando algún lenguaje de script en el cliente.

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

  En este ejemplo vemos 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.

Veamos un ejemplo, 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="usuario" size="30" >
Tu Contrase&ntilde;a: <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&ntilde;os
   <input type="radio" name="edad" value="20-40"> Entre 20 y 40 a&ntilde;os
   <input type="radio" name="edad" value="+40"> M&acute;s de 40 a&ntilde;os
</p>

<p><input type="hidden" name="lugar" value="p&aacute:gina personal"> </p>

Como encontraste mi p&aacute;gina :
<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>
```

Atributos comunes para los elementos de formularios

Algunos tipos de campo requieren de la ayuda de atributos para mejorar su rendimiento o determinar su importancia en el proceso de validación. Por ejemplo, el atributo novalidate se puede usar para evitar que el formulario completo sea validado o formnovalidate para hacer lo mismo con elementos individuales. El atributo autocomplete facilita medidas de seguridad adicionales para el formulario completo o elementos individuales.

Otros atributos que se pueden aplicar a diferentes tipos de campos de entrada son:

- Atributo **placeholder**=”texto”

  Su valor se presenta en pantalla dentro del campo, como una sugerencia para ayudar al usuario a introducir la información correcta. Se suele utilizar en campos tipo search y entradas de texto

  Ejemplo:

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

- Atributo **required**

  Cuando este atributo booleano es incluido en un campo, evitará que el formulario sea enviado si el campo se encuentra vacío. 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. Por ejemplo, si el atributo required no está presente cuando usamos el tipo email para recibir una dirección de correo, el navegador comprueba si la entrada es un email válido o no, pero validará la entrada si el campo está vacío.

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

  Otro ejemplo de uso del atributo required

  ```
  <p> Debe aceptar nuestros términos del servicio<br>
     <label><input type="checkbox" name="cb-terminosservicio" required>
         Acepto los términos del servicio
     </label>
  </p>
  ```

- Atributo **multiple**

  Este atributo booleano **multiple** es 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.

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

- Atributo **autofocus**

  Cuando el documento se cargue, el campo tendrá automáticamente el foco. Anteriormente a HTML5 esta funcionalidad se conseguía utilizando el método focus() de Javascript.

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

- Atributo **pattern**

Usa expresiones regulares para personalizar reglas de validación. Algunos tipos de campo validan cadenas específicas, pero no permiten hacer validaciones personalizadas. El atributo **pattern** nos permite crear nuestro propio tipo de campo para controlar valores no ordinarios.

Se puede incluir un atributo **title** para personalizar mensajes de error.

Por ejemplo, si queremos introducir un código postal que consiste en 5 números

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

IMPORTANTE: Para obtener información adicional sobre expresiones regulares: http://es.wikipedia.org/wiki/Expresi%C3%B3n_regular

- Atributo **form**

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**

Esto permite ubicar campos de entrada en diferentes ubicaciones dentro de una misma página. Por ejemplo,

```
<!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**

Permite al usuario seleccionar un valor de una lista de opciones o escribir uno que no esté en ella (este tipo de elemento se suele llamar Combo Box)

Hay que utilizar el elemento **<datalist>** para construir una lista de ítems que será usada como sugerencia en un campo del formulario. Este elemento utiliza el elemento **<option>** en su interior para crear la lista de items a sugerir.

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

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">
```

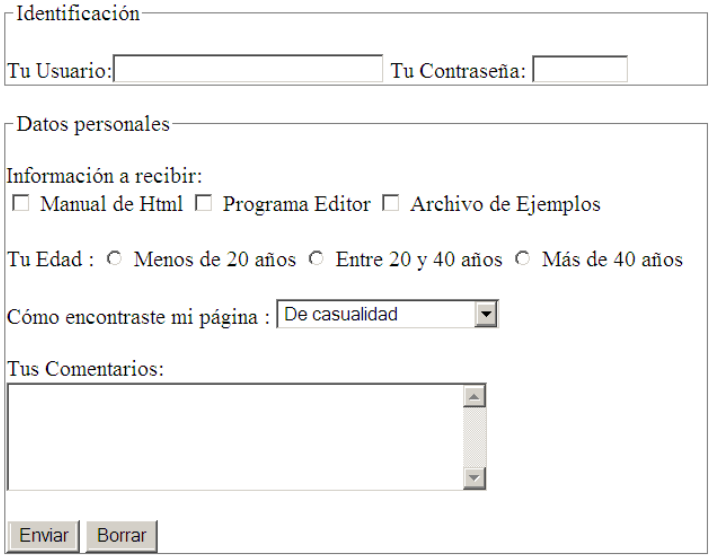
Ejemplo:

```
<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>
```

Añadir estructura a los formularios: los elementos <fieldset> y <legend>

El elemento **<fieldset>** (grupo de campos) permite 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 grupos de campos (fieldset). La leyenda mejora la accesibilidad cuando el fieldset no se representa visualmente.



```
<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=”archive” 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>
        <option>Por el buscador Google</option>
        <option>Por el buscador Yahoo</option>
        <option>Me la comentaron</option>
     </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>
```

Navegación con tabulador: atributo 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.

En versiones anteriores de HTML, el atributo tabindex sólo se podía usar con: <a>, <area>, <button>, <input>, <object>, <select>, and <textarea>. En cambio, a partir de HTML5, puede ser usando con cualquier elemento HTML

Ejemplo:

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

3.5.12 Marcos

A partir HTML5 las etiquetas <frameset>, <frame> y <noframes> no pertenecen al estándar.

La etiqueta <iframe> se usa para visualizar una página web dentro de la página que le hace referencia.

Permite insertar un marco dentro de un bloque de texto. Los iframes no se pueden redimensionar. Cuando se usan iframes primero se carga la página y después el marco.

Su uso habitual es para mostrar publicidad o webs de colaboración.

```
<iframe src="url" name=”nombre” heigth=”altura” width=”anchura” ... ></iframe>
```

Sus atributos son:

- **name**=”nombre” Nombre que identifica al iframe
- **id**=”identificador” Identificador del iframe usado para referenciarlo
- **src**=”url” URL del documento HTML que se visualiza en el iframe
- **height**=”altura” Altura que ocupará el iframe en el documento
- **width**=”anchura” Anchura que ocupará el iframe en el documento
- **srcdoc**=”codigoHTML” Código HTML de la página que se va a mostrar en el iframe
- **sandbox**=”allow-forms|allow-pointer-lock|allow-popups| allow-same-origin|allow-scripts|allow-top-navigation” Conjunto de restricciones que se permiten para el contenido del iframe.

```
<iframe width="560" height="315"
        src="https://www.youtube.com/watch?v=paRh8O5bc18" ></iframe>
```

Por defecto, los iframes tiene bordes. Si se quiere quitar el borde habría que usar la propiedad border de las hojas de estilo. Por ejemplo,

```
<iframe src="demo.html" style="border:none;"></iframe>
```

Haciendo uso de esta propiedad también se podrían modificar las características (color, grosor, apariencia, ...) del borde.

```
<iframe src="demo.html" style="border:2px solid red;"></iframe>
```

3.5.13 Elementos multimedia

A partir de HTML5 se han incorporado etiquetas que indican cómo utilizar elementos multimedia de forma homogénea en los diferentes navegadores. Para ello se incorporan nuevas etiquetas, <canvas>, <video> y <audio>, y se han modificado la definición de otras, como <object>, <param>, etc.

Aunque, a continuación, se presentan estas etiquetas, su uso está fuera del alcance de este módulo ya que es necesario conocer el lenguaje de guiones Javascript para poder manejar audios, videos y canvas con imágenes dinámicas.

Etiqueta <canvas>

Con la etiqueta <canvas> se pueden crear imágenes dinámicas. Utilizando código JavaScript se puede manipular el canvas para dibujar en él y crear gráficos dinámicos de todo tipo (incluidas interfaces de aplicaciones web completas).

```
<canvas id="nombreCanvas" width="anchoPixels" height="altoPixels">
      Texto cuando el navegador no maneja canvas
</canvas>
```

Para empezar a usarlo lo único que hay que especificar son sus dimensiones. El texto que se escribe entre la apertura y cierre de la etiqueta <canvas> sólo es interpretado por navegadores que no soportan esta etiqueta. El resto de trabajo con el canvas está fuera del ámbito de este curso ya que se realiza con código JavaScript.

Etiqueta <video>

HTML5 soporta la inclusión de vídeo empotrado en las páginas web de forma nativa. Con la etiqueta <video> no se especifica el formato del mismo sino que el uso de uno u otro forma vendrá impuesto por el navegador

```
<video id=”identificador” src="url_archivo_video"
       height=”pixeles” width=”pixeles”
       poster=”url_imagen ”  ... >
    Texto cuando el navegador no soporta video
</video>
```

El texto que se escribe entre la apertura y cierre de la etiqueta <video> sólo es interpretado por navegadores que no soportan esta etiqueta.

Sus atributos son:

- **src**=”url” Indica la url del fichero de video
- **height=”**pixeles**”** Fija la altura del reproductor de video
- **width=”**pixeles**”** Fija el ancho del reproductor de video
- **poster**=”url_imagen”
  Indica la imagen que se mostrará mientras se descarga el video, o hasta que el usuario presione el botón de reproducción
- **autoplay** Indica que se iniciará la reproducción tan pronto como esté disponible
- **controls** Se presentan controles de video, como botón de reproducción/pausa, etc.
- **loop** Indica que el video se volverá a iniciar, cada vez que se termine
- **muted** Especifica que la salida de video será silenciada.
- **preload=”**auto | metadata | none**”**
  Especifica si y cómo se cargará el video al cargar la página. Valores posibles indicarían que el navegador cargará todo el fichero de video (**auto**), sólo cargará metadatos (**metadata**) o no cargará el video (**none**) cuando se carga la página.

Etiqueta <audio>

La etiqueta <audio> permite insertar archivos sonoros en diferentes formatos. Además provee de una interfaz de control sobre la reproducción del audio haciendo uso de funciones JavaScript sin necesidad de ningún plugins.

```
<audio id=”identificador” src="url" ... >
      Texto cuando el navegador no soporta audio
</audio>
```

El texto que se escribe entre la apertura y cierre de la etiqueta <audio> sólo es interpretado por navegadores que no soportan esta etiqueta.

Sus atributos son:

- **src**=”url” Indica la url del fichero de audio
- **autoplay** Indica que se iniciará la reproducción tan pronto como esté disponible
- **controls** Visualiza controles de audio, como un botón de reproducción/pausa, etc.
- **loop** Indica que el audio se volverá a iniciar, cada vez que se termine
- **muted** Especifica que la salida de audio debe estar silenciada.
- **preload=”**auto | metadata | none**”**
  Especifica si y cómo se cargará el audio cuando se carga la página. Valores posibles indicarían que el navegador cargará todo el fichero de audio (**auto**), sólo cargará metadatos (**metadata**) o no cargará el fichero de audio (**none**) cuando se carga la página.

Etiqueta <source>

La etiqueta <source> se usa para especificar múltiples recursos para elementos multimedia, como <video> y <audio>. Permite especificar alternativas de video ó audio que el navegador puede elegir, en función de su tipo de medio, soporte de códec, etc.

Por ejemplo,
```
<audio controls>
   <source src="fichero.ogg" type="audio/ogg">
   <source src="fichero.mp3" type="audio/mpeg">
   Tu navegador no dispone de soporte para audios
</audio>
```

Etiquetas <object> y <param>

La etiqueta <object> define un objeto incrustado dentro de un documento HTML. Use este elemento para incrustar multimedia (como audio, video, applets de Java, ActiveX, PDF y Flash) en sus páginas web. También se puede usar para incrustar otra página web en su documento HTML.

Por ejemplo, `<object width="400" height="400" data="fichero.swf"></object>`

La etiqueta <param> se puede usar para pasar parámetros a los complementos que se han incrustado con la etiqueta <object>.

Por ejemplo,
```
<object data="fichero.wav">
   <param name="autoplay" value="true">
</object>
```

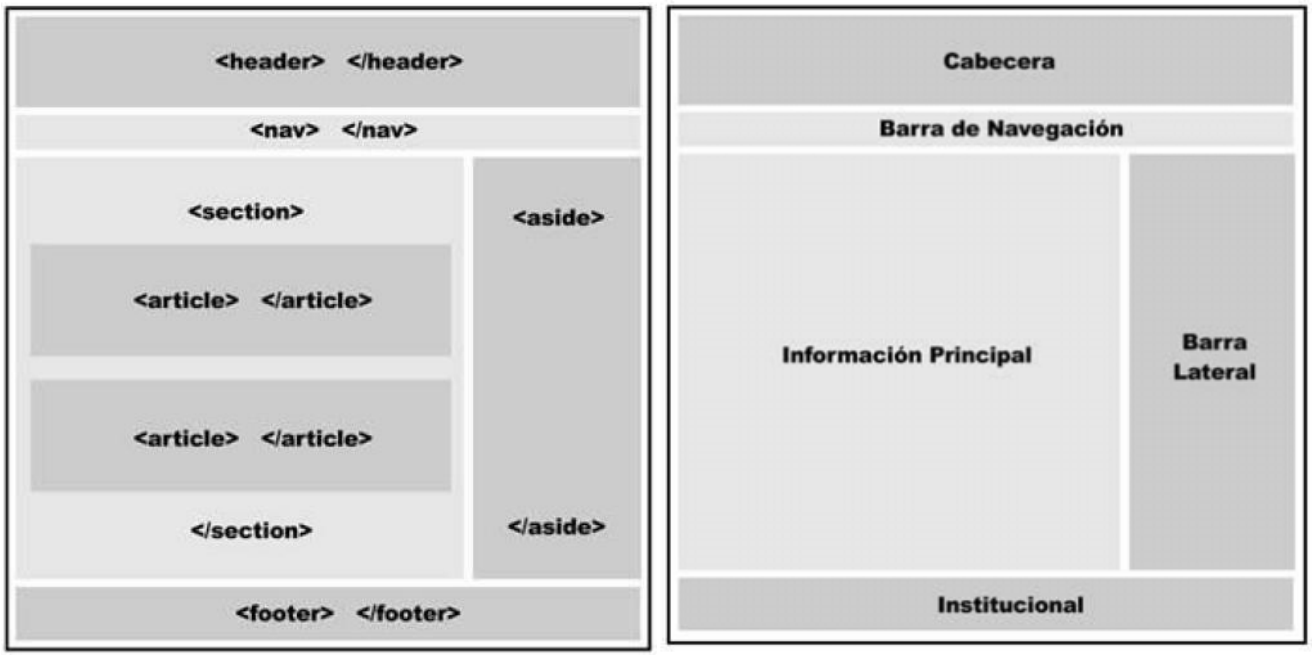
3.5.14 Estructura del cuerpo. Etiquetas semánticas

La idea de HTML5 es que las etiquetas HTML, más que para dar formato, sirvan para dar valor semántico al contenido. Es decir, para indicar qué tipo de contenido es. Con las etiquetas se marca la semántica y con las hojas de estilo CSS que veremos en la siguiente unidad didáctica se le da formato a los contenidos. Con todo

ello se consigue que los navegadores puedan darle más importancia a determinadas secciones, facilitándole además la tarea a los buscadores, así como cualquier otra aplicación que interprete sitios Web.

HTML5 proporciona nuevas etiquetas que introducen un nuevo nivel semántico que hace que la estructura de las páginas web sea más coherente y fácil de entender.

En la figura la imagen de la izquierda representa 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>**

  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>**

  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>**

  Contiene la información más relevante del documento y puede ser encontrada en diferentes formas (por ejemplo, dividida en varios bloques o columnas).

- **<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.

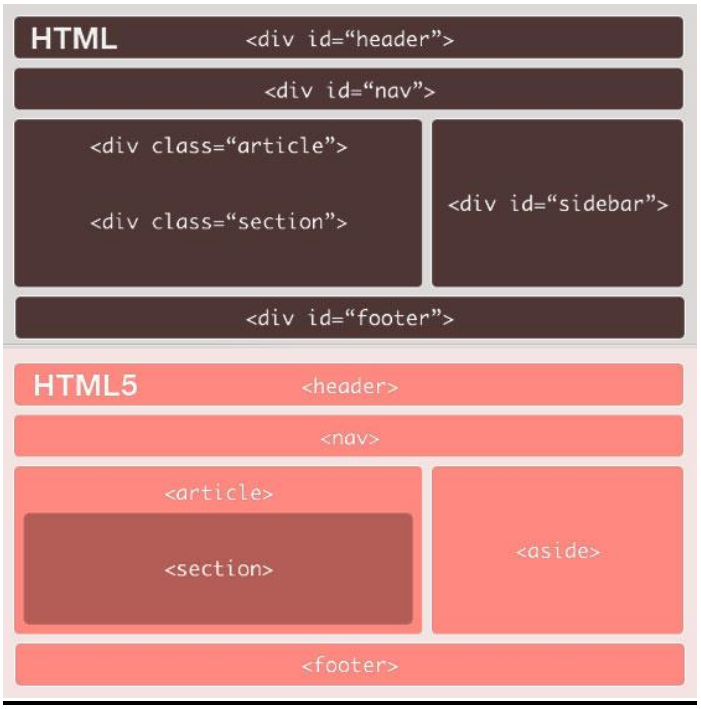
- **<footer>**

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

- **<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.

Hasta ahora se utilizaba la etiqueta y las tablas para estructurar una web en bloques. En la siguiente imagen se puede ver una comparación entre la estructuración realizada con versiones anteriores de HTML y con HTML 5



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).

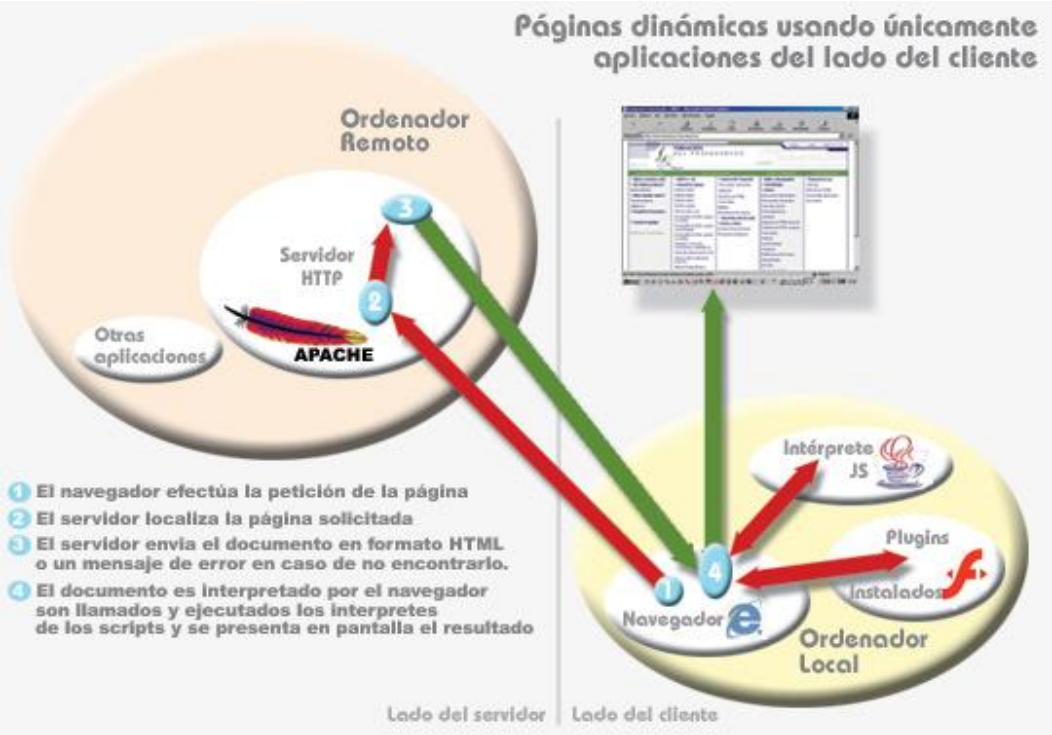
Existen dos tipos de lenguajes de script:

- **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.

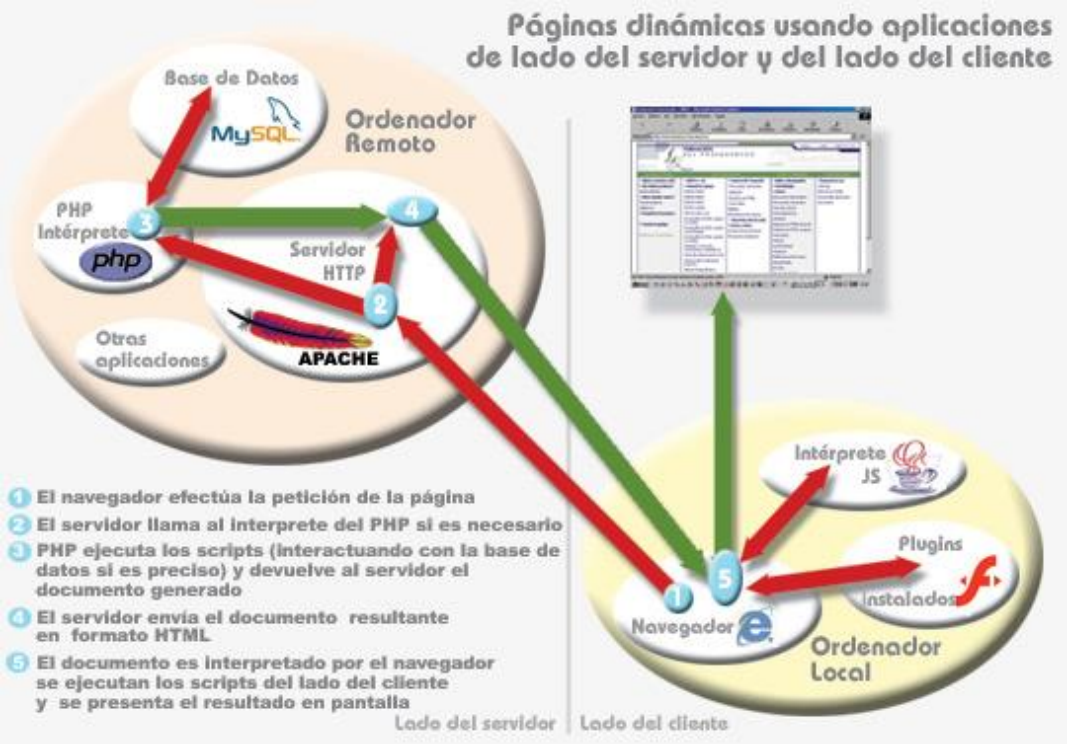
Cada uno de estos tipos tiene sus ventajas y sus inconvenientes:

- **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