

FUNDAMENTOS

HTML5 CSS3

Tabla de contenidos

Tabla de contenidos	2
Tema 1 Introducción.....	8
¿Qué es HTML?	8
Historia del HTML	8
Compatibilidad con navegadores.....	9
Estructura cliente/servidor.....	9
Protocolo HTTP	10
Etapas de una transacción HTTP.....	10
Mensajes.....	11
Métodos de petición.....	11
GET	12
POST	12
PUT.....	12
DELETE.....	12
Tema 2 Fundamentos básicos del HTML.....	13
Etiquetas y elementos.....	13
Atributos	14
Estructura básica del HTML	14
Contenido y contenedores	15
Inserción de archivos css y javascript	16
CSS	16
JavaScript.....	17
Tema 3 Elementos básicos HTML5	18
Texto.....	18
Texto importante	18
Texto enfatizado	18
Texto resaltado	19
Obtendría el resultado:.....	19
Citas	19
Subíndices	20
Superíndices	20
Elemento <i>small</i>	20
elemento <i>s</i>	20
Elemento <i>b</i>	20
Elemento <i>i</i>	21
Abreviaturas con <i>abbr</i>	21
Definiciones	21
Escritura de código o texto informático	21
Escritura de código.....	21
Resultados de operaciones en un ordenador	22
Teclas.....	22

Variables	22
Fechas y horas	22
Elementos especiales.....	23
Span	23
Elemento <i>address</i>	23
Elementos de marcado de correcciones.....	24
Párrafos.....	24
Párrafo simple.....	24
Títulos	25
Salto de línea	27
Línea horizontal	27
Listas	27
Listas con viñetas.....	27
Numéricas.....	27
Listas anidadas.....	28
Listas de términos.....	29
Contenedores	29
div	29
span	31
Estructura de cuerpo HTML5	31
Header	31
Footer	32
Main.....	32
Section	32
Nav.....	33
Article	33
Figure.....	34
Figcaption	34
Aside	34
Secciones de tipo mostrar/ocultar	35
Cuadro de diálogo.....	36
Imágenes.....	37
Inserción de imágenes	38
Atributos de la etiqueta <i>img</i>	38
Enlaces	39
Enlaces locales relativos	39
Enlaces locales absolutos.....	40
Enlaces internos.....	40
Atributos del elemento <i>a</i>	42
Tema 4 Introducción a CSS3	45
Introducción	45
Limitaciones de HTML.....	45
Ayuda de CSS	45
Versiones de CSS.....	46
Nuevos elementos CSS	46
Estandarización de características CSS	47
Compatibilidad.....	47

Sintaxis básica de CSS.....	48
Comentarios	49
Inserción de código CSS	49
En un archivo externo	50
Orden de aplicación de estilos.....	51
alteración del orden de aplicación de estilos	52
Herencias	53
Normalizar. estilo predefinido	53
Indicación de la codificación de caracteres del archivo CSS	54
Selectores CSS3.....	54
Selectores simples	54
Selección de elementos completos HTML	54
Selección por clases de elementos.....	55
Selección por identificadores	57
Selector de limitación.....	57
Selector universal	58
Selección por atributos.....	59
Selectores jerárquicos	62
Selector para un descendiente.....	63
Selector de elementos hijos	64
Otros selectores jerárquicos	64
Pseudoclases.....	69
Básicas	69
Más pseudoclases	70
Pseudoelementos.....	73
Unidades	73
Unidades relativas y absolutas	73
Unidades absolutas de medida.....	74
Unidades relativas de medida	74
Colores	76
Fuentes	78
Propiedades de las fuentes.....	78
Propiedades CSS para modificar la fuente.....	79
font-size.....	79
font-family.....	80
font-weight.....	81
font-style	81
font-variant	81
line-height	81
font	82
color.....	82
Importar tipos de letra	82
Servicios online de importación de tipos de letra	85
Formato del texto en CSS	85
word-spacing	85
letter-spacing	85
text-decoration.....	85
text-decoration-color	86
text-decoration-style.....	86
vertical-align	86

text-align	86
text-indent.....	86
text-transform	86
direction	87
text-overflow.....	87
text-shadow.....	87
word-wrap.....	87
Cajas.....	87
Formato de fondo.....	87
Color de fondo.....	87
Imagen de fondo. background-image	87
Repetición del fondo. background-repeat	88
Posición de la imagen. background-position.....	88
Desplazamiento del fondo. background-attachment	88
Tamaño del fondo. background-size	89
Área de dibujo del fondo. background-clip.....	89
Propiedad background	89
Poner varias imágenes de fondo	90
Elementos del formato de caja.....	90
Configuración del borde	91
Propiedades para el borde	91
Bordes redondeados	93
Bordes usando imágenes (CSS3)	94
Sombreado de la caja. box-shadow	96
Perfiles	96
Listas	97
Propiedades CSS para listas	97
list-style-type.....	97
list-style-image	98
list-style-position.....	98
list-style	98
Posicionamiento mediante CSS	98
Flotación	99
Propiedad shape-outside	100
Tamaño del elemento	101
Posicionamiento	101
Posicionamiento estático	101
Posicionamiento relativo.....	101
Posicionamiento fijo.....	103
Posicionamiento absoluto.....	104
Establecer coordenadas de posicionamiento	104
Coordenada z.....	104
Tema 5 Tablas.....	106
Creación de tablas simples	106
Celdas de cabecera	107
Títulos	108
Agrupación de filas	108
Agrupación de columnas	109
Elemento colgroup	109
Elemento col.....	110

Combinar celdas	111
Formato CSS para tablas.....	112
Espaciado y borde en las tablas	112
Tema 6 Formularios.....	116
Introducción	116
Atributos de formulario HTML.....	116
HTML <form> Elementos	118
Elemento <input>	118
Elemento <label>	119
Elemento <select>	119
El <textarea> Elemento	120
Elemento <button>.....	121
Elementos <fieldset> y <legend>.....	121
Elemento <datalist>.....	121
El elemento <output>.....	122
Tipos de entrada HTML. Atributo type del elemento input.	122
Text	123
Password.....	123
Submit.....	124
Reset	124
Radio	125
Checkbox	125
Button	125
Color	125
Date	126
Datetime-local	126
Email	127
File	127
Month	127
Number.....	128
Range	128
Search	129
Tel	129
Time	129
Url	129
Week.....	130
Atributos de entrada.....	130
Value.....	130
Readonly	131
Disabled	131
Size.....	131
Maxlength.....	132
Min Max.....	132
Múltiple	133
Pattern	133
Placeholder	134
Required	134
Step.....	135
Autofocus.....	135

Height Width.....	135
List.....	136
Autocomplete	136
Formulario de entrada HTML* Atributos.....	137
Form.....	137
Formaction.....	137
Formenctype.....	138
Formmethod	138
Formtarget.....	139
Formnovalidate.....	140
Tema 7 Elementos multimedia.....	141
El problema de la incompatibilidad de formatos.....	142
Vídeos en las páginas web.....	143
Formatos de vídeo	143
Inserción de vídeo.....	143
Atributos del elemento video.....	143
Uso de diferentes formatos de vídeo	144
Subtítulos	145
Audio en las páginas web	146
Elemento canvas.....	147
Atributos de canvas	147
Inserción directa de imágenes SVG	148
Ecuaciones matemáticas	148
Otros elementos multimedia.....	149
El problema del vídeo y el audio.....	149
Etiqueta embed	150
Etiqueta object	150
Elemento param	150
Elemento iframe	151

Tema 1 Introducción

¿Qué es HTML?

Según la Wikipedia:

“**HTML**, siglas en inglés de **HyperText Markup Language** (‘lenguaje de marcas de hipertexto’), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Es un estándar a cargo del *World Wide Web Consortium* (**W3C**) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. HTML se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la *World Wide Web* (WWW). Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.”

Historia del HTML

La historia de HTML ha sido larga, ya que desde su nacimiento al principio de los 90, ha evolucionado muchísimo, pasando de ser tan solo un simple sistema para compartir documentos, a ser la base completa de la WEB.

HTML fue desarrollado inicialmente por Tim Berners-Lee en 1980, que trabaja en aquel entonces en la CERN (Organización Europea para la investigación Nuclear). El propuso un sistema de Hipertexto que permitiera compartir documentos de una manera más fácil y eficiente. Pero no fue hasta que se unió con Robert Cailliau para presentar su propuesta como la World Wide Web (W3), en un concurso para definir el sistema de hipertexto que se utilizaría para internet.

Cabe mencionar que para el año de 1991 HTML se publicó bajo el nombre de HTML Tags. Fue hasta 1993 cuando la IETF (Internet Engineering Task Force) propuso a HTML para convertirse en un Estándar. Sin embargo, no fue hasta 1995 cuando HTML ya estaba en la versión 2.0 cuando logro establecer a HTML como el primer Estándar oficial de HTML.

En 1994 Tim Berners-Lee funda la W3C (*World Wide Web Consortium*) con la finalidad de generar recomendaciones y estándares para la WEB y es cuando decide en 1996 publicar bajo la W3C el estándar HTML. En 1997 se publica la versión 3.2 bajo la W3C.

En 1998 se publica la versión HTML 4.0, la cual incorpora importantes mejoras como: Hojas de estilo (CSS), scripts, mejoras en los formularios, etc. Finalmente, en 2004 se publica la última versión oficial de la W3C, correspondiente a la 4.01 la cual no contenía mejoras importantes.

Cabe mencionar que durante los años 90 se producía la denominada “Guerra de los Navegadores” en donde Internet Explorer y NetScape lidiaban una batalla campal por ser el líder de los navegadores, lo que provoco que cada navegador liberar versiones semanas tras semanas

e incluyera funcionalidad NO ESTANDAR, provocando la completa incompatibilidad de entre los navegadores. Esto provocaba que los desarrolladores tuvieran que desarrollar versiones completas para cada navegador. Un que esta guerra trabajo mucho desorden al desarrollo web, es importante mencionar que también ayude al rápido avance tecnológico de la WEB. Un que al final Internet Explorer gano la guerra, cabe mencionar que NetScape nos dejó dos legados, JavaScript y el nacimiento de la fundación Mozilla.

Depuse de la liberación de la versión HTML 4.01 la W3C anuncio que HTML ya no tenía futuro y que en cambio XHTML sería el futuro. XHTML no logro tener tanto apoyo como se esperaba y mientras tanto Apple, Opera y Mozilla se preocupaban por la falta de interés de la W3C por el estándar HTML y es cuando se organizan y crean una nueva asociación llamada WHATWG (Web Hypertext Application Technology Working Group) la cual consideraba que el estándar de HTML seguía siendo el futuro pero que necesitaba importantes mejoras. Fue cuando en 2008 la WHATWG publica el primer borrador de HTML5.

Es importante destacar que aun que HTML fue creador del estándar HTML, fue la WHATWG quien desarrollo y actualmente dirige el estándar de HTML5. Un que la W3C termino aceptando al HTML5 como el estándar.

En la actualidad HTML5 es el estándar de la WEB y propuso tantas mejoras que termino de expulsar a Flash.

Compatibilidad con navegadores

No todas las características que ofrecen HTML5 y CSS3 son admitidas por los navegadores web actuales. Por tanto, es necesario consultar algunas páginas de referencia para conocer qué elementos se pueden utilizar porque son admitidos por la mayoría de los navegadores, y qué elementos no se pueden utilizar porque no funcionarán.

Para ello las siguientes web nos pueden ayudar a saber si los navegadores con los que vamos a trabajar son compatibles:

Por versión:

[Can I use... Support tables for HTML5, CSS3, etc](#)

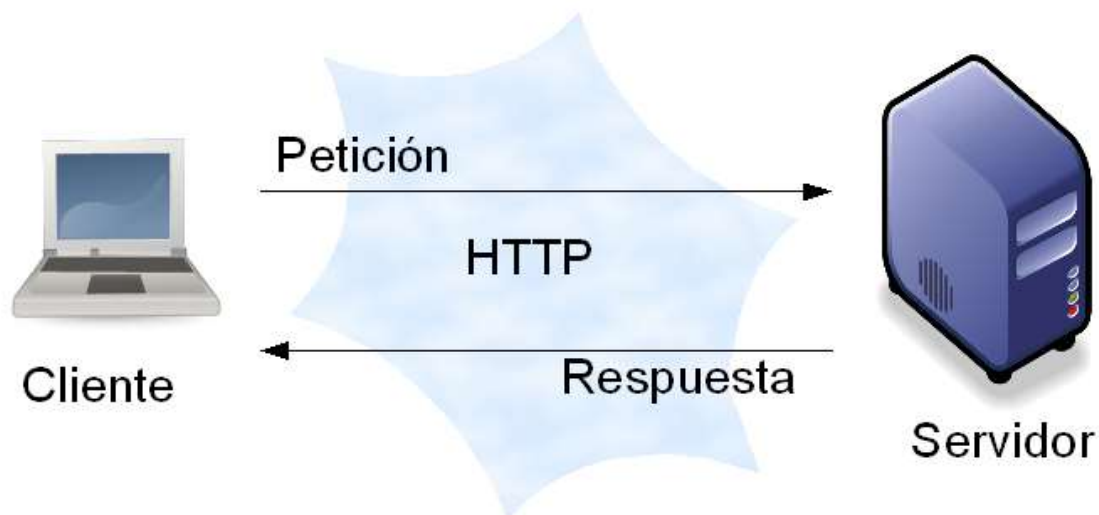
E incluso por características del HTML5:

[The HTML5 Test](#)

[Estructura cliente/servidor](#)

Según la Wikipedia:

“La **arquitectura cliente-servidor** es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados **servidores**, y los demandantes, llamados **clientes**. Un **cliente** realiza peticiones a otro programa, el **servidor**, quien le da respuesta.”



Protocolo HTTP

El Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP 1/0 está recogida en el RFC 1945. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

Etapas de una transacción HTTP.

Para profundizar más en el funcionamiento de HTTP, veremos primero un caso particular de una transacción HTTP; en los siguientes apartados se analizarán las diferentes partes de este proceso.

Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

1. Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo Location del cliente Web.
2. El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
3. Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente. Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,...), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada (casi siempre HTTP/1.0) y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor,...
4. El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
5. Se cierra la conexión TCP.

Este proceso se repite en cada acceso al servidor HTTP. Por ejemplo, si se recoge un documento HTML en cuyo interior están insertadas cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento HTML y cuatro para las imágenes.

Mensajes

Los mensajes HTTP son en texto plano lo que lo hace más legible y fácil de depurar. Esto tiene el inconveniente de hacer los mensajes más largos.

Los mensajes tienen la siguiente estructura:

- Línea inicial (termina con retorno de carro y un salto de línea) con
- Para las peticiones: la acción requerida por el servidor (**método de petición**) seguido de la **URL** del recurso y la versión HTTP que soporta el cliente.
- Para respuestas: La versión del HTTP usado seguido del **código de respuesta** (que indica que ha pasado con la petición seguido de la **URL** del recurso) y de la frase asociada a dicho retorno.
- Las **cabeceras** del mensaje que terminan con una línea en blanco. Son **metadatos**. Estas cabeceras le dan gran flexibilidad al protocolo.
- Cuerpo del mensaje. Es opcional. Su presencia depende de la línea anterior del mensaje y del tipo de recurso al que hace referencia la URL. Típicamente tiene los datos que se intercambian cliente y servidor. Por ejemplo para una petición podría contener ciertos datos que se quieren enviar al servidor para que los procese. Para una respuesta podría incluir los datos que el cliente ha solicitado.

Métodos de petición

HTTP define una serie predefinida de métodos de petición (algunas veces referido como "verbos") que pueden utilizarse. El protocolo tiene flexibilidad para ir añadiendo nuevos métodos y para así añadir nuevas funcionalidades. El número de métodos de petición se ha ido aumentando según se avanzaba en las versiones.

Cada método indica la acción que desea que se efectúe sobre el recurso identificado. Lo que este recurso representa depende de la aplicación del servidor. Por ejemplo, el recurso puede corresponderse con un archivo que reside en el servidor.

Los más importantes son

GET

El método GET solicita una representación del recurso especificado. Las solicitudes que usan GET solo deben recuperar datos y no deben tener ningún otro efecto. (Esto también es cierto para algunos otros métodos HTTP.)

POST

Envía datos para que sean procesados por el recurso identificado en la URI de la línea petición. Los datos se incluirán en el cuerpo de la petición. A nivel semántico está orientado a crear un nuevo recurso, cuya naturaleza vendrá especificada por la cabecera *Content-Type*. Ejemplos:

Para datos formularios codificados como una URL (aunque viajan en el cuerpo de la petición, no en la URL): *application/x-www-form-urlencoded*

Para bloques a subir, ej. ficheros: *multipart/form-data*

Además de los anteriores, no hay un estándar obligatorio y también podría ser otros como *text/plain*, *application/json*, *application/octet-stream*,...

PUT

Envía datos al servidor, pero a diferencia del método POST la URI de la línea de petición no hace referencia al recurso que los procesará, sino que identifica al los propios datos. Otra diferencia con POST es semántica: mientras que POST está orientado a la creación de nuevos contenidos, PUT está más orientado a la actualización de los mismos (aunque también podría crearlos).

DELETE

Borra el recurso especificado.

Tema 2 Fundamentos básicos del HTML

Etiquetas y elementos

Un documento HTML contiene texto. Ahora bien, hay texto que sirve para indicar **elementos** y propiedades de la página. Es decir el texto contiene lo que se denomina **etiquetas**.

Las etiquetas sirven para delimitar elementos de la página. Los elementos son cada uno de los elementos de la misma; por ejemplo, un párrafo es un elemento de la página, una tabla también. Incluso hay elementos que contienen otros elementos (las tablas constan de filas y las filas de celdas, por ejemplo).

Las etiquetas son textos encerrados entre los signos de mayor y menor (< >). Cuando un navegador se encuentra un texto así encerrado, entenderá que dentro de los símbolos menor y mayor lo que se indica es el inicio de un elemento. El inicio del elemento se marca con el nombre del elemento entre los símbolos < y >.

Ejemplo:

```
<strong>
```

La etiqueta **strong** sirve para indicar texto importante (se suele mostrar en negrita)

La mayoría de etiquetas afectan a un determinado texto, el cual estará *encerrado* por las etiquetas. Por lo tanto, casi siempre, existe una etiqueta de apertura y otra de cierre, que se interpretará como el inicio y fin de un determinado elemento de la página.

El cierre de una etiqueta se marca poniendo el símbolo /, seguido del nombre del elemento estamos cerrando. Ejemplo:

```
Texto normal <strong>texto negrita</strong>
```

Literalmente lo que indicamos en el ejemplo anterior es que el texto contiene un elemento de tipo *strong*, que contiene el texto: *texto negrita*. Un navegador reaccionará mostrando ese texto con un tipo de letra más fuerte. Es decir:

```
Texto normal texto negrita
```

Ya se ha comentado que un elemento puede contener dentro más elementos, por lo que entre la apertura y el cierre de una etiqueta puede haber más etiquetas. Una norma importante es que siempre debemos de cerrar primero las etiquetas abiertas más tarde.

Es decir, este código **no es correcto**.

```
Texto normal <strong><em>texto negrita y cursiva</strong></em>
```

Se está cerrando la etiqueta *em* después de *strong* y eso no es correcto porque *em* es la última que se abrió. Lo correcto es:

```
Texto normal <strong><em>texto negrita y cursiva</em></strong>
```

Es decir, se cierra primero el elemento **em** que fue el último en abrirse.



Lo cierto es que los navegadores no suelen dar problemas cuando cometemos errores en el código. Incluso un fallo como el que muestra la ilustración, no sería considerado. Lo que hacen es arreglar nuestros fallos, siempre que sea posible. La razón es que les interesa que las páginas se vean correctamente y no actuar como validadores del código.

Atributos

Algunas etiquetas tienen atributos. Los atributos son propiedades de cada elemento a las que podemos asignar un valor, de modo que dicho valor varía el comportamiento del elemento. La forma de utilizar atributos es:

```
<elemento nombreDeAtributo1="valor1"
nombreDeAtributo2="valor1" ... >
```

A los atributos se les asignan valores que deben ir entre comillas. El símbolo de igualdad (=) es obligatorio.

Ejemplo:

```
<p lang="es">
```

Indica que el elemento está marcado en idioma español.

La sintaxis de atributos es:

```
<etiqueta atributo1="valor1" atributo2="valor2" ...>
....
</etiqueta>
```

En el cierre sólo se cierra la etiqueta, sin indicar los atributos. Ejemplo:

```
<p class="normal">Texto del párrafo normal</p>
```

Estructura básica del HTML

Para crear una página web usando **HTML 5** la estructura básica es la siguiente:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    </body>
</html>
```

Donde el significado de los elementos es el siguiente:

- **línea DOCTYPE**. La etiqueta **DOCTYPE** indica el tipo de HTML que estamos utilizando (concretamente HTML 5). Se trata de una línea que la actualidad siguen ignorando casi todos los navegadores, pero que es muy recomendable (y muy respetuoso con la norma) escribir. En ella se indica la versión HTML que sigue el documento.
- **etiqueta html**. Todo documento HTML está encerrado dentro de la apertura y el cierre de esa etiqueta. Marca el principio y fin del mismo (marca el **elemento raíz** de un documento HTML). Se le debe añadir el atributo **lang** para indicar el idioma en el que está escrito el documento.
- **cabecera, etiqueta head**. Encierra las etiquetas de cabecera, las cuales proporcionan información que el navegador debe tener en cuenta para el contenido del documento aparezca de forma correcta y también contiene elementos que proporcionan información a los buscadores (como Google). Los elementos habituales que contiene son:
 - **etiquetas meta**. Que sirven para indicar aspectos de funcionamiento de la página web como, cabeceras del protocolo http, indicaciones sobre cada cuánto caduca la página, palabras clave para buscar la página en los navegadores, codificación de caracteres de la página,... En este caso sólo se usa la etiqueta **meta** obligatoria que indica que estamos codificando el texto usando el formato **utf-8** de **Unicode**.
 - **título, etiqueta title**. El título de la página, que los navegadores suelen colocar en la barra de título de la ventana en la que se muestra la página.
 - **declaración de archivos externos y scripts**. En el ejemplo no hay declaraciones de este tipo.
- **cuerpo**. Encerrado en la etiqueta **body**, encierra el contenido en sí de la página web. Lo que el navegador muestra por pantalla es lo que se coloca en este apartado

Contenido y contenedores

En HTML los elementos que muestran algo en la página, se colocan dentro de la sección **body**.

Además hay que tener en cuenta que hay elementos pensados para contener grandes cantidades de información (como **div**) y elementos que contienen poco texto (como **strong**). Es correcto que aparezca un elemento **strong** dentro de un elemento **p**, pero no al revés:

```
<strong><p>Hola</p></strong> -->;Incorrecto!!  
<p><strong>Hola</strong></p> -->Correcto
```

La mayoría de elementos en HTML son contenedores de otros textos y elementos. Hay que conocer que tipos de elementos tenemos. Los tipos están relacionados con la propiedad CSS **display**, que se encarga de decir cómo tiene que mostrarse un elemento. Por defecto los elementos funcionan de esta forma:

- **Elementos de tipo inline**. Son elementos que se muestran seguidos en la misma línea. Los elementos de este tipo son los que marcan texto simple. Todas las etiquetas de marcado de texto tienen este *display*: **strong**, **em**, **span**, **mark**, **abbr**, etc.

Ejemplo, el código:

```
<strong>Hola</strong> <em>Hola</em>
```

Se muestra así:

Hola *Hola*

- **Elementos de tipo block.** Son contenedores más grandes. Los elementos que marcan párrafos enteros (como `p`, `h1`, `h2` o `blockquote`) y los contenedores de secciones (como `div`, `section`, `article` o `figure`) tienen este `display`. Su característica es que, por defecto, tienen forma rectangular y abarcan toda la anchura de su contenedor. Si indicamos para ellos un color de fondo, este se muestra en forma de rectángulo (en los `inline`, el color de fondo abarca exactamente al texto, como si lo hubiéramos pintado con un rotulador). Ejemplo:

```
<h1>Hola</h1> <p>Hola</p>
```

Ese código se muestra así:

Hola

Hola

- **Elementos de tipo inline-block.** Son rectangulares como los `block`, pero no abarcan todo su contenedor. Se adaptan a su contenido. Ejemplos son los elementos `button` o `image`:

```
<strong>Hola</strong> <button>Hola</button>
```

El resultado muestra un botón (elemento rectangular) seguido del texto de tipo `strong`. Si el botón tuviera un `display` de tipo `block`, saltaría a la línea siguiente.

Hola

En todo caso, el `display` de un elemento se puede modificar desde lenguaje CSS, pero no conviene hacer combinaciones anti naturales ya que se pierde el sentido semántico del código.

Inserción de archivos css y javascript

CSS

Si deseamos aplicar en nuestro documento código CSS procedente de un archivo aparte, debemos utilizar el elemento `link` cuya sintaxis es:

```
<link href="ruta" rel="stylesheet">
```

Los archivos CSS contienen instrucciones que permiten modificar la apariencia de nuestro documento.

La etiqueta `link` debe ir en el apartado `head` de la página, normalmente después del título y las etiquetas `meta`.

JavaScript

En este caso la etiqueta que lo permite es **script**. Esta etiqueta, a diferencia de **link**, sí tiene cierre.

Sirve para incrustar código JavaScript en el documento. Ejemplo:

```
<script>
  alert("Hola");
</script>
```

Si el código JavaScript procede de otro archivos, el código de ese archivos se incrusta de esta forma:

```
<script src="archivo.js">
  alert("Hola");
</script>
```

Tema 3 Elementos básicos HTML5

Hoy en día en una página web, todo el texto está dentro de algún elemento (delimitados mediante etiquetas), en realidad hay tres contextos principales a tener en cuenta sobre el texto

Elementos capaces de contener varios párrafos. La W3C llama a estos elementos *sectioning content*. Cada elemento de esta parte es lo que hoy en día se conoce como **capa** o **sección**. Si no indicamos capa alguna, entonces todos los párrafos están contenidos en la etiqueta **body**, lo que significa que el contenedor de los párrafos será el área que el navegador deja para el texto y que puede ser casi toda la pantalla.

Elementos que contienen párrafos individuales. Aunque en HTML 5 se permite escribir texto que no esté dentro de un párrafo, no es una buena práctica, ya que no podríamos indicar más adelante el formato del párrafo (alineaciones, márgenes,...).

Lógicamente dentro de un párrafo, no podemos indicar otro párrafo; es decir no podemos meter una etiqueta **p**, dentro de una etiqueta **h1**.

Cuando un texto está dentro de un elemento de párrafo (como **p**), hará que el texto siguiente quede en otro párrafo, por lo tanto habrá un salto de párrafo entre ambos textos.

Elementos que contienen texto simple o conjuntos de caracteres. La W3C llama a estos elementos *phrasing content*. Son elementos pensados para ser interiores a un párrafo. Un mismo párrafo puede contener muchos elementos de este tipo. El mismo texto puede estar contenido en varios elementos de este tipo a la vez.

Por ejemplo, podemos indicar que un texto es **strong** y **em** a la vez.

Texto

En este caso, detallaremos los elementos que se emplean para señalar texto interior a un párrafo.

Texto importante

En HTML 5 el texto importante se marca con el elemento **strong**. En general todos los navegadores marcan el texto *strong* en negrita (letra más gruesa), pero que se muestre en negrita es un acuerdo visual que se puede modificar con el lenguaje CSS. Es decir, no es lo mismo texto en negrita que texto importante.

Sin duda, **strong** es uno de los elementos HTML fundamentales e imprescindible en los documentos HTML.

Texto enfatizado

El elemento **em** marca texto como enfatizado. El resultado visual con que le dan formato por defecto los navegadores es texto en cursiva.

La idea semántica es marcar con **em** texto con énfasis de segundo nivel (dejando a **strong** el primer nivel) que normalmente se dedica a palabras no aceptadas aun, títulos de trabajos, apodos, texto subjetivo, etc.

Texto resaltado

Se trata del tercer elemento de HTML5 para remarcar texto. Se hace a través el elemento **mark**. En este caso, su efecto visual, suele ser aplicar en el texto así marcado un fondo de color vivo que simula el uso de un subrayador. Ejemplo:

```
<p>Yo soy texto normal, <mark>y yo estoy remarcado
</mark></p>
```

Obtendría el resultado:

Yo soy texto normal, y yo estoy remarcado

Este elemento no es tan antiguo como los anteriores porque apareció con HTML5 (los navegadores antiguos no lo reconocen). La idea semántica es marcar texto que queremos realzar de forma muy llamativa indicado texto a recordar o sobre el que profundizar.

Citas

En HTML5 existen tres formas de marcar citas:

- **q**. Usada para indicar citas cortas incluidas en un párrafo con otro tipo de texto. Indica un texto que procede de otra fuente. Remarca una cita, con la connotación de “texto entrecomillado”. De hecho, los navegadores, por defecto, entrecomillan su contenido. Ejemplo de uso:

```
Como dijo Julio César, <q
cite="http://es.wikipedia.org/wiki/Alea_iacta_est">
Alea Jacta es</q>
```

No es obligatorio utilizar el atributo **cite**. Este atributo permite indicar una fuente para consultar la cita original (los navegadores, sin embargo no hacen nada con el contenido de este enlace).

- **cite**. Este elemento era el que originalmente se usaba para citas literales. Actualmente se recomienda su uso para indicar títulos de trabajos y el nombre de sus autores. Eso incluye títulos de libros, memorias, ensayos, poemas, obras musicales, sitios web, entradas de blog o foros, comentarios, tweets, discursos, etc. Es decir cualquier trabajo creativo. Ejemplo

```
<p>
Edgar Codd publicó: <cite>Un modelo relacional de datos para grandes
bancos de datos compartidos
</cite> en 1970
</p>
```

Los navegadores suelen mostrar el contenido de **cite** en cursiva.

- **blockquote**. Este es el único elemento de marcado de citas que trabaja a nivel de párrafo. Es decir, indicado párrafos completos que son citas procedente de fuentes externas al documento. Se le puede indicar (al igual que se hace con el elemento **q**) el atributo **cite** para indicar un enlace a la fuente de la cita. Ejemplo:

```
<blockquote
cite="http://noticias.juridicas.com/base_datos/Admin/constitucion.tp.html#a1">
<strong>Artículo 1 de la constitucion</strong><br>
```

```

España se constituye en un Estado social y democrático de Derecho,
que propugna como valores superiores de su ordenamiento jurídico
la libertad, la justicia, la igualdad y el pluralismo político.
</blockquote>

```

- Puede utilizar el atributo cite (como se observa en el ejemplo), para indicar la dirección URL que se ha utilizado como fuente de la cita.

Visualmente los navegadores muestran el texto **blockquote** añadiendo una sangría a la derecha y a la izquierda. Ignoran, visualmente, el uso del atributo **cite**, pero otras herramientas sí usan este contenido.

Subíndices

Permite que el texto aparezca por debajo de la línea base y en un tamaño más pequeño. Lo hace el elemento **sub**, ejemplo:

```
<p>La fórmula del agua es H<sub>2</sub>O</p>
```

Mostraría: H₂O

Superíndices

Parecida al anterior, pero ahora el texto marcado con el elemento **sup** aparecerá por encima y en pequeño. Ejemplo:

```
<p>La fórmula del agua es H<sup>2</sup>O</p>
```

Obtendría: H²O

Elemento *small*

Marcado de letra pequeña. Se usa, por ejemplo en textos que marquen copyright, derechos de uso, comentarios, notas anexas y letra pequeña (tipo la de los contratos) en general. El efecto visual es que el texto sale con un tamaño más pequeño. Es posible incluso anidar un elemento **small** dentro de otro, de modo que cada vez sale más pequeño el texto. Ejemplo:

```

Texto normal <small>Pequeño
<small>Aún más pequeño</small></small>

```

elemento *s*

Se trata de un antiguo elemento de las páginas web que permitía indicar un subrayado. En la versión de HTML 4 quedó obsoleto, no se recomendaba su uso ya que el subrayado se reserva para los enlaces. Con HTML 5 se ha redefinido su uso para marcar texto obsoleto que, visualmente, se muestra con efecto de tachado.

Ejemplo:

La capital de Alemania es <s>Bonn</s> Berlín

Elemento *b*

Se trata del elemento histórico (de las primeras versiones de HTML) utilizado para marcar texto en negrita. En HTML 4 quedó como obsoleto. En HTML 5 se volvió a permitir pero solo si

necesitamos un nivel de marcado de texto más allá del uso de **strong**, **em** o **mark**. En definitiva, elemento de uso muy poco aconsejable que se mantienen solo por la gran cantidad de creadores que lo siguen utilizando.

Elemento *i*

Tiene el mismo problema que el anterior. Es el elemento histórico (de las primeras versiones de HTML) utilizado para marcar texto en cursiva. En HTML 4 quedó como obsoleto. En HTML 5 se puede utilizar para marcar texto tipo alternativo si no encaja con otros elementos de marcado de caracteres.

Al igual que ocurre con el elemento *b*, su uso no es aconsejable.

Abreviaturas con *abbr*

Marca texto que indica una abreviatura o un acrónimo. Su uso requiere utilizar el atributo `title` para indicar el significado de la abreviatura. En general, los navegadores no muestran ningún resultado, pero sí un cartel con el contenido de **title** cuando se arrima el ratón. Ejemplo:

```
<p>Se ha coronado a
```

```
<abbr title="Su Alteza Real">SAR</abbr> el Rey Felipe VI </p>
```

En HTML 5 se eliminó el uso del elemento **acronym** que estaba pensado para acrónimos. Ahora tanto acrónimos como abreviaturas se deben utilizar con **abbr**.

Definiciones

El elemento **dfn** se usa de forma similar al anterior. Sirve para indicar definiciones para el término o palabras que encuadra.

Ejemplo:

```
<dfn title="usar dos conceptos de significado opuesto en una sola expresión">oxímoron
```

```
</dfn>
```

Los navegadores suelen mostrar el contenido del elemento en cursiva y, como siempre, en un cartelito muestran el contenido del atributo **title**:



Escritura de código o texto informático

Escritura de código

HTML proporciona un elemento llamado **code** que se utiliza cuando en un documento deseamos escribir ejemplos de código escritos en un lenguaje de programación. Por ejemplo:

```
<p>Este es un ejemplo de código en JavaScript:</p>
```

```
<code>
```

```
    window.onload=function() {  
        alert("Hola");  
    }
```

</code>

Lo cual obtendría este resultado:

Este es un ejemplo de código en JavaScript:

```
window.onload=function(){ alert("Hola"); }
```

Se puede observar como los navegadores modifican el tipo de letra (utilizan un tipo de letra monoespaciada) pero no respetan los espacios y tabuladores utilizados para indicar el código. Como es muy normal querer respetar los espacios y tabuladores cuando se escriben ejemplos de código, lo habitual es hacer esta combinación:

<p>Este es un ejemplo de código en JavaScript:</p>

<pre><code>

```
    window.onload=function() {
        alert("Hola");
    }
```

</code></pre>

Resultados de operaciones en un ordenador

Con intenciones similares al elemento anterior, tenemos el elemento **samp**. Se utiliza cuando queremos en un documento indicar ejemplos de resultados de operaciones en un ordenador. Ejemplo:

<p>Al ejecutar el comando obtuve el mensaje
<samp>Comando o nombre de archivo incorrecto </samp>
</p>

Los navegadores suelen mostrar este elemento en letra monoespaciada

Teclas

El elemento **kbd** sirve para indicar el nombre de una tecla o combinación de teclas (también comandos de voz o entradas de menú). Se puede indicar el atributo **title** para señalar el significado de la tecla.

Ejemplo:

<p>Para finalizar puse <kbd>Ctrl+C</kbd></p>

Variables

El elemento **var** sirve para indicar texto que se refiere a nombres de variables de un programa informático. Sólo tiene sentido su uso para documentos sobre programación.

Fechas y horas

El elemento **time** está presente desde HTML5 (en Internet Explorer desde la versión 9) y permite indicar una fecha y una hora (por ejemplo 10:00). Atributos:

datetime. Hora en formato estándar (yyyy-mm-dd o yyyy-mm-ddThh:mm:ssTZ) que representa la hora exacta que muestra el texto.

Ejemplo:

Hoy es `<time datetime="2017-10-13">`13 de Octubre de 2017`</time>`, se encendió el servidor a la `<time datetime="2017-10-13T01:12:00-0100">`1 y 12 de la mañana (hora local)`</time>`

Elementos especiales

- **bdi**. Se usa para aislar texto de su contenedor a fin de aplicarle formato diferenciado. Se usa si queremos marcar texto con una intención especial y atípica.
- **bdo**. Permite indicar la dirección del texto. Ejemplo:

```
<bdo dir="rtl">Este texto sale al revés</bdo>
```

Saldría séver la elas otzet etsE

Tiene un atributo llamado **dir** que posee dos valores posibles: **ltr** (el texto se muestra de izquierda a derecha) y **rtl** (el texto se muestra de derecha a izquierda)

- **u**. Hasta HTML 3.2 indicaba subrayado. Como este formato, fuera de un enlace, no es recomendable, se eliminó del estándar. HTML 5 le ha recuperado para indicar texto que tiene un estilo muy diferente del texto general, como palabras mal escritas, fallos de ortografía o combinaciones de texto sin sentido.

Los navegadores siguen mostrando este texto con un subrayado.

- **wbr**. Contiene una palabra a la que estaremos indicando que puede ser partida por un guión por parte del navegador.

Span

Sirve para marcar un trozo de texto de forma inespecífica, sin indicar semántica alguna. En definitiva, selecciona un trozo de texto por necesidades técnicas (por ejemplo, para indicar un formato CSS diferente a ese texto).

La recomendación es utilizar un elemento más semántico siempre que sea posible.

Elemento address

El elemento **address** permite indicar una dirección, especialmente indicada para mostrar información sobre el autor o los créditos del documento (con información de los autores por ejemplo) al estilo de los pies de artículos periodísticos.

Aunque, de forma clásica, se le considera un elemento de marcado de párrafos, actualmente se le ha denotado como un elemento de marcado de secciones

Ejemplo:

```
<address>
```

Texto escrito por ``

Oscar Perez`
`

```
Calle Los Informáticos 5<br>
Madrid<br>
28001<br>
España
</address>
```

Los navegadores suelen mostrar este tipo de párrafos en cursiva y centrados.

Elementos de marcado de correcciones

Permiten marcar texto para indicar como texto que se está revisando y que falta decidir su contenido final. Hay dos elementos de este tipo **ins** y **del**.

ins. Indica un texto añadido al documento. Puede utilizar dos atributos:

del. Indica un texto que se desea retirar del documento.

Ambos elementos pueden utilizar dos atributos:

cite. Con un enlace a una dirección de Internet que proveerá de más información sobre la corrección.

datetime. Indica la fecha de la modificación con cuatro cifras para el año, dos para el mes y dos para el día.

Ejemplo:

```
<p>La capital de Alemania es <del>Bonn</del>
<ins datetime="2012-03-12">Berlín</ins>
</p>
```

De manera oculta se ha indicado que el término Berlín se corrigió el 3 de diciembre de 2012. El código marcado para eliminar aparece tachado y el de inserción aparece subrayado.

Párrafos

Desde hace años todo texto en un documento HTML debe ir dentro de una etiqueta que sirva para decir qué tipo de texto es. Y eso significa, que el texto debe de ir dentro de una etiqueta de párrafo. Se comentan las etiquetas disponibles para marcar párrafos.

Párrafo simple

La norma HTML estándar establece que el texto de la página debe estar contenido dentro de un elemento. Es decir, no se puede poner texto directamente dentro de la etiqueta *body*; ha de haber un contenedor para ese texto.

El elemento más sencillo lo marca la etiqueta **p**, que indica un elemento de párrafo normal. Todo texto dentro de una etiqueta *p*, queda marcado como texto dentro de un párrafo normal.

Normalmente los navegadores utilizan fuentes tipo *Times* de tamaño *11pt* para la letra de párrafo normal. Ejemplo de uso:

```
<body>
<p>
```



```

        Párrafo con un poco de texto
    </p>
    <p>
    Este es un párrafo completo englosado dentro de una etiqueta
    <em>p</em>,
    aunque
        el texto se escriba
        con todos estos saltos de línea      y      espacios
        se mostrará seguido,
        sin saltos      de
        línea interiores
                                y entre cada
palabra dejando un solo espacio
    </p>
</body>

```

En la imagen se observa el modo en el que el navegador muestra el párrafo. Por defecto los párrafos tipo p dejan espacio arriba y abajo (normalmente media línea en cada dirección).

Títulos

Hay una serie de siete etiquetas que comienzan con la letra *h* a la que le sigue un número del 1 al 7. Sirven para marcar párrafos de forma que se considerarán títulos del texto. De modo que el elemento **h1** marcará títulos de primer nivel, **h2** de segundo nivel,... y así hasta **h6** (sexto nivel).

<h2>Sistema Solar</h2> <h3>Planetas</h3> <h4>Mercurio</h4> <p>Mercurio es el planeta del Sistema Solar más próximo al Sol y el más pequeño (a excepción de los planetas enanos). Forma parte de los denominados planetas interiores o rocosos y carece de satélites</p> <h4>Venus</h4> <p>Venus es el segundo planeta del Sistema Solar en orden de distancia desde el Sol, y el tercero en cuanto a tamaño, de menor a mayor</p> <p>...</p> <h3>Sol</h3> <p>El Sol es la estrella del sistema planetario en el que se encuentra la Tierra; por tanto, es la más cercana a la Tierra y el astro con mayor brillo aparente</p> <h3>Satélites</h3> <h4>Luna</h4> <p>La Luna es el único satélite natural de la Tierra y el quinto satélite más grande del Sistema Solar.</p> <h4>Ío</h4> <p>Fue descubierto por Galileo Galilei en 1610 y recibió inicialmente el nombre de Júpiter I como primer satélite de Júpiter.</p> <p>...</p> <h2>Otros Sistemas</h2> <h3>Fomalhaut</h3> <p>Estrella conocida desde la prehistoria ahora se le han descubierto planetas</p> <h3>Vega</h3> <p>Antigua estrella polar, muy venerada. Posee un disco de polvo que podría contener planetas o bien formarse pronto</p>
--

Ejemplo:

```

<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Estrellas</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <h1>Sistema Solar</h1>
    <h2> Planetas</h2>
    <h3>Mercurio</h3>
    <p>
      Mercurio es el planeta del Sistema Solar más
próximo al Sol
y el más pequeño (a excepción de los planetas enanos). Forma parte
de los
denominados planetas interiores o rocosos y carece de satélites
    </p>
    <h3>Venus</h3>
    <p>
      Venus es el segundo planeta del Sistema Solar en orden de
distancia
desde el Sol, y el tercero en cuanto a tamaño, de menor a mayo
    </p>
    <p>...</p>
    <h2>Sol</h2>
    <p>
      El Sol es la estrella del sistema planetario en el que se
encuentra la
Tierra; por tanto, es la más cercana a la Tierra y el astro con
mayor
brillo aparente
    </p>
    <h2>Satélites</h2>
    <h3>Luna</h3>
    <p>
      La Luna es el único satélite natural de la Tierra y el
quinto
satélite más grande del Sistema Solar.
    </p>
    <h3>Ío</h3>
    <p>
      Fue descubierto por Galileo Galilei en 1610 y recibió inicialmente
el nombre de Júpiter I como primer satélite de Júpiter.
    </p>
    <p>...</p>
    <h1>Otros Sistemas</h1>
    <h2>Fomalhaut</h2>
  <p>
    Estrella conocida desde la prehistoria ahora se le han descubierto
planetas
  </p>
  <h2>Vega</h2>
  <p>Antigua estrella polar, muy venerada. Posee un disco de polvo
que podría
  contener planetas o bien formarse pronto
  </p>
</body>
</html>

```

Salto de línea

A veces es necesario dentro del texto de un determinado párrafo hacer un salto de línea. El elemento que lo realiza no tiene cierre y se llama **br**. Ejemplo:

```
<p>Primera línea <br>Segunda línea</p>
```

Línea horizontal

Otra posibilidad es hacer un salto pero dejando una línea horizontal en el hueco de las palabras. Esto lo hace la etiqueta **hr** (que tampoco tiene cierre):

```
<p>Primera línea <hr>Segunda línea</p>
```

Aunque los navegadores entienden este código. En realidad **hr** tiene que estar fuera de las etiquetas de párrafo, es decir lo correcto es:

```
<p>Primer párrafo</p>
<hr>
<p>Segundo párrafo</p>
```

Listas

Las listas permiten crear párrafos agrupados y alineados mediante símbolos como viñetas o números para facilitar la lectura y organización de las ideas del documento.

Listas con viñetas

Las listas con viñetas se deben englobar dentro de un elemento **ul** (acrónimo de *unordered list*, lista no ordenada), después cada párrafo de la lista estará dentro de elementos de tipo **li** (de *list item*, elemento de lista).

Ejemplo:

```
<p>Lista de la compra</p>
<ul>
  <li>Agua</li>
  <li>Vino</li>
  <li>Cerveza</li>
</ul>
```

Resultado:

Lista de la compra

- Agua
- Vino
- Cerveza

Numéricas

Las listas numéricas aparecen dentro del elemento **ol** (de *ordered list*, lista ordenada), después cada párrafo de la lista estará dentro de elementos de tipo **li**, al igual que las anteriores. La diferencia ahora es que cada párrafo con **li**, aparece con un número y no con una viñeta.

Ejemplo:

```
<p>Lista de la compra</p>
<ol>
  <li>Agua</li>
  <li>Vino</li>
  <li>Cerveza</li>
</ol>
```

Resultado:

Lista de la compra

1. Agua
2. Vino
3. Cerveza

Listas anidadas

Es posible meter una lista dentro de otra, por ejemplo:

```
<p>Lista de la compra</p>
<ul>
  <li>
    No alcohólicas
    <ul>
      <li>Agua</li>
    </ul>
  </li>
  <li>
    Alcohólicas
    <ul>
      <li>Vino</li>
      <li>Cerveza</li>
    </ul>
  </li>
</ul>
```

También es posible anidar mezclando tipos de listas:

```
<p>Lista de la compra</p>
<ol>
  <li>
    No alcohólicas
    <ul>
      <li>Agua</li>
    </ul>
  </li>
  <li>
    Alcohólicas
    <ul>
      <li>Vino</li>
      <li>Cerveza</li>
    </ul>
  </li>
</ol>
```

Listas de términos

Permite crear una lista de definiciones de términos. En ellas se indica el término a definir y su significado. Ejemplo:

```
<dl>
  <dt>Windows</dt>
  <dd>
    Sistema operativo de <strong>Microsoft</strong> disponible
para PC
    disponible en versiones de 32 y 64 bits y para servidores,
ordenadores e incluso tabletas y móviles.<br>
    La última versión es la versión 10 y la 2016 para
servidores.
  </dd>
  <dt>Linux</dt>
  <dd>
    Sistema operativo de código abierto disponible
en numerosas distribuciones gratuitas y de pago.
    Es la base del sistema <strong>Android</strong>.
  </dd>
  <dt>Mac OS</dt>
  <dd>
    Sistema operativo de los ordenadores de la empresa
<strong>Apple</strong>
    <br>
    La última versión es <strong>Mojave</strong>
  </dd>
</dl>
```

Resultado:

Windows	Sistema operativo de Microsoft disponible para PC disponible en versiones de 32 y 64 bits y para servidores, ordenadores e incluso tabletas y móviles. La última versión es la versión 10 y la 2016 para servidores.
Linux	Sistema operativo de código abierto disponible en numerosas distribuciones gratuitas y de pago. Es la base del sistema Android .
Mac OS	Sistema operativo de los ordenadores de la empresa Apple La última versión es Mojave

Contenedores

div

Se trata de un elemento clásico, ya estaba presente en las primeras versiones de HTML, que se utiliza como contenedor de otros elementos a fin de poder manipularlos todos a la vez. Dentro de **div** se pueden colocar todo tipo de etiquetas (tablas, párrafos, imágenes,...).

Desde hace años se ha convertido en el elemento utilizado, combinado con CSS, para crear **capas**. Las capas son elementos visuales que se pueden posicionar a voluntad en la página web. De esta forma, a través de las capas se consiguen maquetaciones muy artísticas.

Por defecto, los navegadores no dan formato al contenido de un elemento **div**, con lo que visualmente no se diferencia del resto de elementos, salvo por el hecho de que un elemento **div** tiene un **display** de tipo block. Es decir, el contenido de estos elementos, por defecto, se muestra de forma separada al resto.

En el ejemplo siguiente se destaca el elemento **div** coloreándolo de rojo:

```
<h1>Título principal</h1>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing  
elit. Quos, architecto, sapiente corporis debitis  
placeat libero ipsa labore molestias omnis facere  
odit fugiat. Dolore, quam, tempore molestias  
architecto veritatis magnam atque?
```

```
</p>
```

```
<div style="background-color:red;">
```

```
<h2>Título secundario</h2>
```

```
<p>Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Odio, fugit, sint maxime unde  
laudantium architecto asperiores libero sit  
accusamus itaque doloribus quae error aliquam in  
soluta commodi porro mollitia excepturi!
```

```
</p>
```

```
<p>Commodi, eaque, voluptate a culpa explicabo  
deserunt sequi molestias nemo quibusdam  
consequuntur architecto cum iste et nobis  
praesentium minima illum. Ex laborum consectetur  
voluptatem deleniti magnam iste tenetur est  
facilis.
```

```
</p>
```

```
</div>
```

```
<p>Velit, ex tenetur architecto culpa officia  
soluta exercitationem recusandae aut ea  
praesentium. Recusandae, qui, repudiandae, eaque  
amet doloribus quod ad illum nihil pariatur iste  
natus velit nostrum voluptatum nemo dicta.
```

```
</p>
```

El resultado en el navegador sería:

Título principal

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quos, architecto, sapiente corporis debitis placeat libero ipsa labore molestias omnis facere odit fugiat. Dolore, quam, tempore molestias architecto veritatis magnam atque?

Título secundario

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Odio, fugit, sint maxime unde laudantium architecto asperiores libero sit accusamus itaque doloribus quae error aliquam in soluta conimodi porro mollitia excepturi!

Commodi, eaque, voluptate a culpa explicabo deserunt sequi molestias nemo quibusdam consequuntur architecto cum iste et nobis praesentium minima illum. Ex laborum consectetur voluptatem deleniti magnam iste tenetur est facilis.

Velit, ex tenetur architecto culpa officia soluta exercitationem recusandae aut ea praesentium. Recusandae, qui, repudiandae, eaque amet doloribus quod ad illum nihil pariatur iste natus velit nostrum voluptatum nemo dicta.

span

Es muy similar al anterior, pero como los navegadores no agregan nada (ni siquiera espacios) a esta etiqueta cuando se muestra por pantalla, en lugar de para definir capas (aunque se podría) se usa para marcar contenido interior a un párrafo, a fin de que a ese contenido se le pueda dar un formato especial mediante CSS. Es decir, con `div` marcamos capas (contenedores grandes) y con `span` marcamos bloques pequeños de texto.

Ejemplo:

`<p>`Dentro de este texto ``esta frase sale de color rojo``. Esto vuelve a salir normal`</p>`

Estructura de cuerpo HTML5

La idea en HTML5 (a diferencia de las versiones anteriores) es que los elementos HTML aporten valor semántico al contenido. Es decir, se trata de que el código HTML indique qué tipo de contenido es el que está dentro de cada elemento. Eso vale para casi todos los elementos ya conocidos como `p` (párrafo), `abbr` (abreviatura), `strong` (remarcado fuerte), etc.

Sin embargo los estándares anteriores a la versión 5 tenían una carencia muy importante. El único elemento para marcar contenedores grandes era `div`. Y este elemento no aporta ninguna semántica a su contenido.

En HTML 5 hay una serie muy importante de elementos que no dan ningún formato al texto, salvo el hecho de que aparezcan con un display de bloque, pero permiten darle un significado. Será CSS el encargado de que esos elementos aparezcan con un formato apropiado a su significado en la página final. La idea final es que estos elementos sustituyan el sobreuso que, tradicionalmente, se hacía de la etiqueta `div`.

No es fácil esta idea, puesto que existen numerosas plantillas y patrones de trabajo profesionales (como ocurría por ejemplo con `Bootstrap`) que se basan en `div` para trabajar con ellas. La mayoría de plantillas actuales (incluida la propia `Bootstrap`) se han adaptado a esta idea.

Header

Permite marcar contenido como cabecera del documento. Hay que señalar que este elemento no tiene nada que ver con el elemento `head`. Header marca una serie párrafos para indicar que pertenecen a la cabecera de la página.

La cabecera de la página suele contener el logotipo de la página, el menú, el título principal, etc.

Realmente una página puede tener varios elementos `header`. Si está al nivel de la etiqueta `body` indica que su contenido es la cabecera de la página completa. Pero dentro de un elemento de sección, por ejemplo, `article`, indicaría que su contenido es la cabecera de ese artículo.

No se debe indicar un elemento header dentro de otro header ni dentro de un elemento `footer` o `address`.

Footer

Similar al anterior, pero sirve para marcar el pie de una página, sección, artículo etc. En el caso de que un elemento footer se coloque al nivel del elemento `body`, servirá para indicar el pie general de la página que suele contener información sobre el autor, copyright, términos de uso de la página, contacto, etc.

Al igual que ocurría con header, no se puede colocar un footer dentro de otro ni puede estar dentro de header o address.

Main

Es incluso más moderno que los anteriores. Permite indicar una sección (que podrá incluir artículos, cabeceras, títulos, párrafos, secciones de imagen,...) de forma que se entenderá que esta es la sección principal del documento.

Solo puede haber un elemento main en una página web. Se usa para diferenciar de las secciones de menú, publicidad o cualquier tipo de información en la página considerada como secundaria.

`main` no puede colocarse dentro de secciones de tipo `article`, `footer`, `header`, `aside` o `nav`. Sí se admite dentro de `section`, para marcar elementos principales dentro de una sección.

Section

Es un elemento que permite dividir en diferentes partes o secciones un documento. Ejemplo de página dividida en secciones:

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <header>
      <h1>Historia de HTML</h1>
      <p>Desde 1989 hasta nuestros días</p>
    </header>
```



```
<section id="inicio">
  <h2>Inicios en HTML. Tim Bernes Lee</h2>
  <p>.....</p>
</section>
<section id="creacion">
  <h2>Creación de la web. Primeros navegadores
</h2>
  <p>.....</p>
</section>
<section id="guerra">
  <h2>La guerra de los navegadores</h2>
  <p>.....</p>
</section>

<section id="versiones">
  <h2>De HTML 4 a HTML 5 pasando por XHTML</h2>
  <p>...</p>
</section>
<footer>
  <p>Realizado por Jorge Sánchez</p>
</footer>
</body>
</html>
```

El atributo **id** no es obligatorio, pero a veces se usa con la finalidad de identificar de modo único a cada sección, por ejemplo para darle un formato especial a una sección concreta.

Nav

Se trata de un elemento que marca a su contenido como una sección de enlaces o menú, es decir una barra de navegación. Más adelante con CSS se puede dar un formato especial a dichos enlaces. **nav** se puede escribir dentro de cualquier elemento HTML estructural (**section**, **article**, **header**, **footer**,...), pero tampoco se deben marcar todos los enlaces de navegación como parte de un elemento nav, solo los que formen un menú real y completo de navegación.

Los elementos marcados por **nav** pueden ser omitidos por los lectores digitales de páginas web que utilizan, por ejemplo, las **personas** invidentes. Lo cual facilita la comprensión del texto) y así que dicho contenido quede marcado solo para utilizar los enlaces interiores a **nav**.

Lógicamente dentro de **nav** se suelen incluir numerosos elementos de tipo **a**.

Article

Si se observan los elementos descritos anteriormente parece claro que HTML 5 utiliza, como metáfora, la forma de distribuir contenidos de un periódico. Así hay cabeceras, pies, secciones y, con este elemento, artículos.

La idea es colocar dentro de este elemento, que tiene sentido que aparezca dentro del elemento **section** o incluso aparecer de forma independiente, contenido que pueda ser entendido como un **todo que** describa un tema de forma íntegra.

Es decir dentro de `article` se colocan los elementos que sirvan para describir un mismo tema.

Es posible (aunque poco habitual) colocar elementos `article` dentro de otros elementos `article`, pero solo cuando queremos indicar artículos externos relacionados con el artículo principal.

La autoría del artículo (nombre del autor y créditos) se puede indicar con el elemento `address`.

Figure

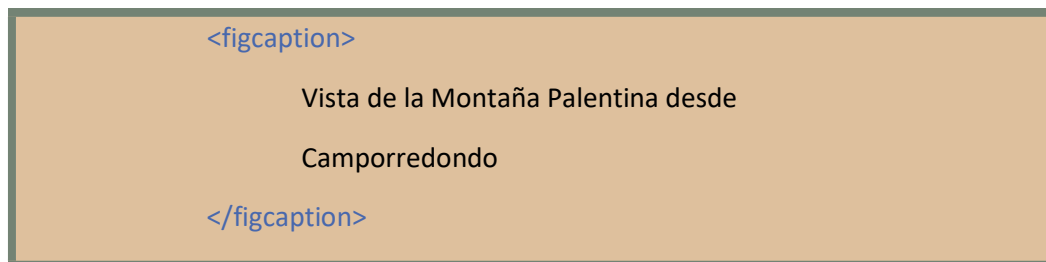
Sirve para agrupar los elementos relativos a una imagen como la propia imagen (elemento `img`), el título de la misma, el pie, los párrafos relativos, etc. Contenido relacionado con el que tiene alrededor, pero que se podría utilizar de forma independiente en otro documento.

Figcaption

Permite indicar el título de una imagen, dentro de un elemento `figure`, haciendo el efecto de un pie de imagen. Por defecto la imagen y el texto quedarán alineados de la misma forma.

Ejemplo de uso:

```
<article>
  <hgroup>
    <h2>Palencia</h2>
    <h3>Paisajes</h3>
  </hgroup>
  <figure>
    
```



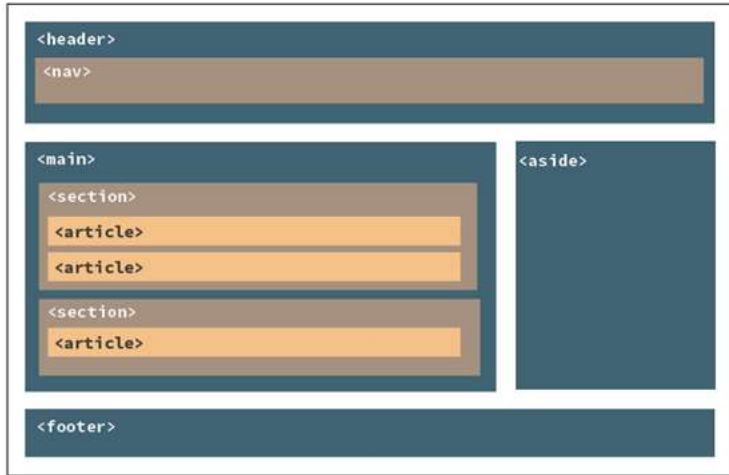
```
</figure>
<p>
  Los paisajes en la provincia de Palencia.....
</p>
....
</article>
```

Aside

Permite marcar texto dentro de un artículo para que no se tenga en cuenta como parte del texto del artículo, sino como un texto aparte que permite realizar aclaraciones al artículo, referencias, resúmenes remarcados y sobre todo cuadros de texto de estilo periodístico para destacar partes del artículo. Para que este texto aparezca de manera especial, debe dársele formato con CSS.

La idea es simular el típico recuadro que aparece en los periódicos en los que se expresan ideas interesantes o que resumen el artículo.

En definitiva, `aside` define texto que no forma parte del flujo principal de la página, sino que posee contenido que debe de fluir aparte.



Secciones de tipo mostrar/ocultar

Se trata de secciones que están dentro de un elemento llamado **details**. Dentro de ese elemento se puede poner una sección de tipo **summary** la cual muestra un texto de resumen de la sección. Al hacer clic en el apartado **summary**, se muestra todo el contenido de la sección.

Es decir en definitiva consigue crear en un documento HTML un apartado de tipo resumen/detalle. Ejemplo:

Ejemplo:

```
<body>
```

```
<details>
```

```
<summary>
```

La ciudad de Palencia

```
</summary>
```

```
<p>Localidad de 80.000 habitantes situada entre
```

los Valles de Cerrato y la Tierra de Campos

en plena meseta castellana</p>

```
<p>Más información en
```

```
<a href="https://es.wikipedia.org/wiki/Palencia">
```

Wikipedia-Ciudad de Palencia

```
</a>
```

```
</p>
</details>
</body>
```

summary y **details** forman parte de la especificación HTML 5.2, aunque ya todos los navegadores (salvo los de Microsoft: Edge y Explorer) los reconocen.

element **summary**

Se coloca dentro del anterior. Es un texto que se puede poner en los elementos de tipo **detail**. Ese texto aparece en lugar del texto *Detalles* que aparece por defecto en una sección de tipo *detail*.

Cuadro de diálogo

Se suelen llamar cuadros de diálogo a las ventanas que los sistemas muestran a los usuarios para darles mensajes, o para permitirles configurar una determinada respuesta u operación.

En HTML 5 se ha incorporado una nueva etiqueta llamada **dialog**, precisamente con esta finalidad. Todavía está en fase de pruebas. El contenido del elemento dialog se muestra en un recuadro. La gracia es que desde JavaScript tiene muchas opciones de manipulación.

Hay un atributo llamado **open** (no se le pone valor alguno cuando se usa) que hace que se muestre el cuadro de diálogo (sin ese atributo, el cuadro no se muestra). Ejemplo:

```
<dialog id="favDialog" open>
  <form method="dialog">
    <fieldset>
      <legend>Sexo</legend>
      <input type="radio" id="hombre"
        name="sexo">
      <label for="hombre">Hombre</label><br>
      <input type="radio" id="mujer"
        name="sexo">
      <label for="mujer">Mujer</label>
    </fieldset>
    <br>
    <label for="nombre">Nombre</label>
    <input type="text" name="nombre"
      id="hombre">
    <button id="cancel" type="reset">
```

Cancelar

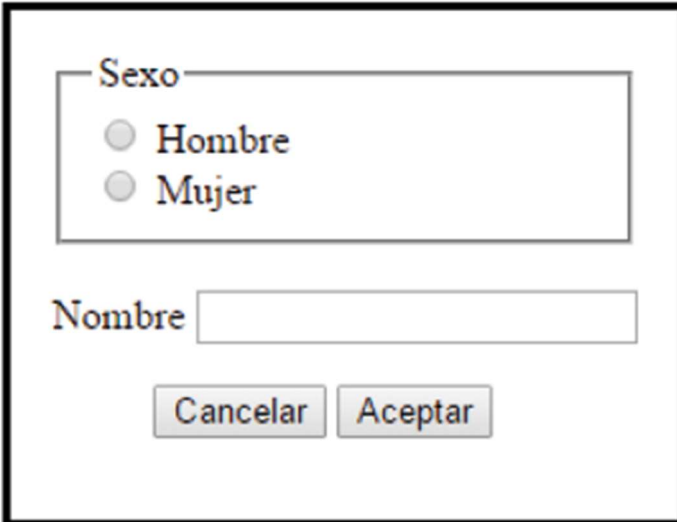
```
</button>
```

```
<button type="submit">Aceptar</button>
```

```
</form>
```

```
</dialog>
```

Resultado:



Realmente sus principales posibilidades se consiguen combinando su uso con el lenguaje JavaScript.

El elemento **dialog** es parte de la recomendación HTML 5.1 y todavía está en proceso de adopción por parte de la mayoría de navegadores.

Imágenes

Las imágenes son fundamentales para que una página web sea más atractiva. Son el primer contenido no textual que se planteó poder incorporar al estándar HTML. Prácticamente no hay páginas sin imágenes.

Los navegadores tienen capacidad de mostrar imágenes, pero sólo las que pertenezcan a tipos concretos. Los tipos de imágenes reconocidos por la mayoría de navegadores son:

- **Formato jpg.** Son imágenes que ocupan muy poco gracias a su alta compresión. No admiten animaciones ni zonas marcadas con transparencia. Son el formato habitual de la imagen digital en una página web.
- **Formato gif.** Imágenes con hasta 256 colores, por lo que son buenas para dibujos y logotipos, pero no para fotografías. Pueden incluso contener animaciones (uno de sus usos habituales) y colores marcados como transparentes, a través de los cuales se mostraría lo que esté *por debajo* de la imagen.

- **Formato png.** Imágenes fotográficas comprimidas al estilo de jpg pero con más calidad (ocupan más, normalmente) Permite la posibilidad además de utilizar canales alfa, es decir marcar opacidades con hasta 255 niveles. De esta forma habrá partes de la imagen que se mezclarán con el fondo. También hay posibilidad de hacer animaciones estilo gif.
- **Formato webp.** Formato de imagen auspiciado por **Google** para conseguir mejorar a PNG y JPG haciendo que ocupen menos sin pérdida de calidad respecto a estos formatos. Sin embargo, solo los navegadores **Chrome y Opera** son capaces de interpretar este formato.
- **Formato SVG.** Es un formato vectorial de imágenes. Las imágenes vectoriales tienen la ventaja de que nunca pierden calidad independientemente de cuanto las ampliemos o las reduzcamos. La desventaja es que no se pueden utilizar para imágenes fotográficas. Tenía problemas de compatibilidad en algunos navegadores, pero en los últimos años ha ganado mucha popularidad y es compatible con todos los navegadores.

Descartando a las imágenes **SVG**, el tamaño en disco de las imágenes puede ser mayor o menor dependiendo de su tamaño y su compresión. De modo que un tamaño grande implica más tardanza al cargar la página, pero una mayor nitidez en la imagen.

Las imágenes son fundamentales en las páginas web, su elección resulta vital para la estética de la misma, nunca se considera un mero acompañamiento ya que el impacto visual y las sensaciones sobre la profesionalidad de la página siempre le consiguen las imágenes con ayuda de la tipografía (pero ahí ya entra el lenguaje CSS) y la disposición o maquetación (para lo que también necesitamos CSS).

Para minimizar el efecto de la incompatibilidad de algunos formatos de imagen se dispone del elemento **picture** que se explica más adelante.

Inserción de imágenes

Las imágenes se colocan mediante la etiqueta **img**. El atributo **src** indica la URL (relativa o absoluta) a la imagen. Ejemplo:

```

```

img es una etiqueta sin cierre, la imagen se coloca directamente en la posición de esta etiqueta. Se la considera como texto al maquetarla. De modo que aparece entre el texto si ponemos la etiqueta entre el texto. Colocar de forma adecuada las imágenes respecto al texto es todo un arte que se consigue (una vez más) con ayuda de CSS.

Atributos de la etiqueta *img*

atributo	significado
alt	<p>Obligatorio. Indica un texto alternativo. Ese texto aparece cuando la imagen no se ha podido cargar (o durante la carga). También suele aparecer cuando arrimamos el cursor a la imagen a fin de informarnos sobre ella.</p> <p>Es un texto también tenido en cuenta por los buscadores a fin de identificar lo que muestra la imagen.</p>

atributo	significado
	Deberíamos tomarnos este atributo como obligatorio
width	<p>Anchura de la imagen. No es aconsejable su uso, ya que si la ampliamos no se verá en buena calidad y si la reducimos estaremos cargando una imagen grande para luego mostrarla en pequeño; sería más inteligente reducirla primero con un editor de imágenes.</p> <p>En cualquier caso es importante utilizar este atributo (junto con height) para que el navegador sepa de antemano el tamaño de la imagen y así que prepare la página correctamente. De este modo si la imagen no se carga, al menos veremos el rectángulo que la misma ocuparía y la página no se desbarata.</p>
height	Altura de la imagen. Tiene las mismas connotaciones que el atributo anterior.

Enlaces

Los enlaces permiten colocar un texto (u otro elemento, como una imagen o un botón) resaltado de forma especial, de modo que cuando se le hace clic, el navegador web nos llevará al destino de la URL (si está disponible).

La etiqueta que permite realizar enlaces es la etiqueta **a**. El atributo **href** permite indicar la URL a la que se realiza el salto. Ejemplo:

A Augusto le sucedió el emperador

```
<a href="http://es.wikipedia.org/wiki/Tiberio">Tiberio
</a>
```

La palabra Tiberio estará remarcada de modo que al hacer clic en ella saltaremos a la URL <http://es.wikipedia.org/wiki/Tiberio>.

Ese ejemplo muestra un salto absoluto, es decir el enlace nos lleva a una dirección URL global usando la notación explicada en el punto anterior.

Enlaces locales relativos

La mayor parte de enlaces de un sitio web en Internet no saltan a otros sitios, sino que son saltos a recursos del mismo servicio web. Es decir se salta muy a menudo a recursos del mismo servidor web.

En ese caso no se indica una URL global tal cual se explicó anteriormente, sino que se indica un salto local. Es decir, un salto a un recurso en nuestro propio servidor web que toma como punto de partida el directorio en el que se encuentra la página web que realiza el salto.

Por ejemplo imaginemos que estamos creando una página web relacionada con la URL <http://www.miservidor.com/noticias/n1.html> y queremos en esa página hacer un enlace a la URL indicada mediante <http://www.miservidor.com/noticias/n2.html> Podríamos hacerlo con el código:

```
<a href="http://www.miservidor.com/noticias/n2.html">
```

Ver segunda noticia

El enlace funcionaría perfectamente. Pero hay que tener en cuenta que la página a la que saltamos está en el mismo directorio. Si algún día movemos todo nuestro sitio web al dominio *nuevodominio.com* resulta que tendremos que cambiar todo el código para hacer frente a la nueva situación. Por ello lo lógico es usar rutas locales.

El enlace usando una ruta local sería:

```
<a href="n2.html">Ver segunda noticia</a>
```

Hay que observar que no se ha indicado protocolo ni ruta (tampoco se indicaría puerto), simplemente se empieza indicando la ruta. Se asume que la ruta

Ejemplos de enlaces con URL local son:

```
<!-- Salto a la página tiberio.html que estará en el mismo directorio  
que la actual -->
```

```
<a href="tiberio.html">Tiberio</a>
```

```
<!-- Salto a la página tiberio.html que estará dentro del directorio emperadores que  
del directorio actual -->
```

```
<a href="emperadores/tiberio.html">Tiberio</a>
```

```
<!-- Salto a la página tiberio.html que estará en el directorio padre, es decir el di.  
actual -->
```

```
<a href="../tiberio.html">Tiberio</a>
```

```
<!-- Salto a la página tiberio.html que estará en el directorio emperadores, dentro c  
actual-->
```

```
<a href="../emperadores/tiberio.html">Tiberio</a>
```

Enlaces locales absolutos

Hay otra posibilidad de enlace local. Por ejemplo:

```
<a href="/tiberio.html">Tiberio</a>
```

Ese enlace (*/tiberio.html*) buscaría la página *tiberio.html* en el directorio raíz del servidor ya que la ruta comienza con el símbolo */*.

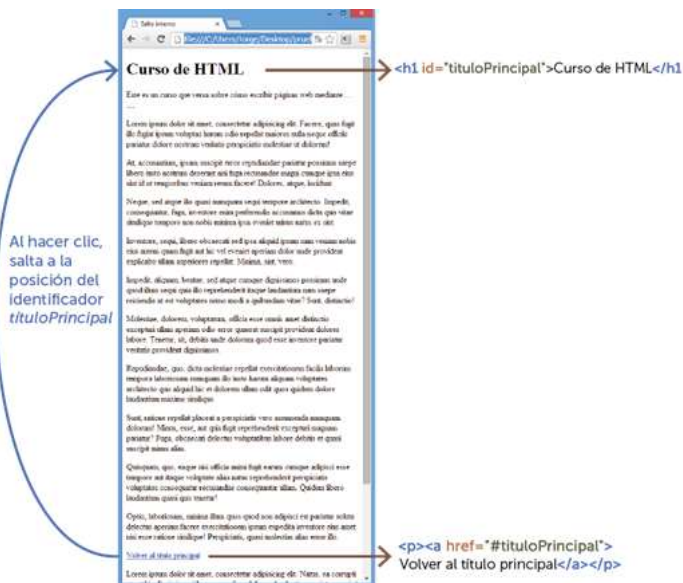
Permiten ahorrar mucho texto y son más fáciles de usar que los enlaces relativos, pero en la práctica no se usan mucho, ya que, si movemos de sitio nuestro sitio web, por ejemplo a un directorio interno del servidor, la raíz ya no sería la misma y el enlace no funcionaría.

Enlaces internos

Hay un tipo de enlace que permite posicionarnos en un punto concreto del documento. Este tipo de enlace se denomina interno, ya que salta dentro del propio documento.

Este salto requiere primero marcar la posición a la que deseamos saltar. Antes de HTML 5 se hacía usando el atributo **name** del propio elemento **a**. Pero con HTML 5 se prohíbe el uso de este atributo con este fin y, en su lugar, se utiliza el atributo **id**.

id es un atributo presente en cualquier elemento de la página web. Sirve para identificar a dicho elemento dentro de la página, por ello no podemos repetir valor para ese atributo en ningún otro elemento.



Supongamos que tenemos este código:

```
<h1 id="tituloPrincipal">Curso de HTML</h1>
```

```
<p>
```

Este es un curso que versa sobre cómo escribir páginas web mediante...

...

```
</p>
```

El párrafo de tipo **h1** está identificado con el identificador *tituloPrincipal*. Gracias a ello si, en otra parte de la página, tenemos el código:

```
<p><a href="#tituloPrincipal">
Volver al título principal</a></p>
```

Al hacer clic en ese enlace (*Volver al título principal*), se usa el identificador como un marcador de posición y así nos colocaremos en dicho párrafo.

También podemos realizar un salto a una posición interna a otro documento. Por ejemplo si este enlace:

```
<a href="manual.html">Abrir manual</a>
```

Nos permite abrir una página web llamada **manual.html** (que estará en el mismo directorio que la página actual). Siempre que se abre un documento se muestra el inicio del mismo. Sin embargo este otro enlace:

```
<a href="manual.html#comentarios">
```

```
Abrir manual por la zona de comentarios</a>
```

Abre el mismo documento, pero se intenta colocar en la zona marcada por el identificador *comentarios*. Lógicamente, para que esto sea posible, debe existir un elemento marcado con ese identificador en el documento.

Es posible indicar una posición interna en una URL absoluta. Por ejemplo este enlace permite abrir la página dedicada a la ciudad de Palencia de la Wikipedia y colocarse en la zona dedicada a la historia:

```
<a href="http://es.wikipedia.org/wiki/Palencia#Historia">
```

```
Historia de Palencia</a>
```

Atributos del elemento *a*

Realmente el único de obligado uso es *href* para indicar el destino del enlace, pero el elemento *a* dispone de unos cuantos atributos (además de los comunes a cualquier elemento HTML) que hacen tareas muy interesantes.

atributo	significado
hreflang	Permite indicar un código de lenguaje (es, fr, en,...) indicando el lenguaje en el que está escrito el destino del enlace. Ejemplo: <pre> Información sobre Londres</pre>
media	Sólo válido en HTML5, permite indicar el medio idóneo para mostrar el contenido del enlace. Ejemplo: <pre> Descargar manual</pre>
target	Permite indicar cómo se muestra la página de destino. Posibilidades: <u>_blank</u> . Abre el enlace en una nueva página. Es una opción muy utilizada al crear enlaces. <u>_parent</u> . Abre el enlace en el marco de la página padre de ella. <u>_top</u> . Abre la página en el marco superior <u>_self</u> . Abre la página en el marco actual <i>nombre</i> . EL nombre indicado será el del marco en el que se abrirá la página Salvo el primero, el resto no se usan por referirse a marcos.
type	Atributo añadido en HTML 5, soportado por la mayoría de navegadores. Permite indicar el tipo de contenido (según la normativa oficial de tipos de la IANA, véase: http://www.iana.org/assignments/media-types/media-types.xhtml

atributo	significado
	<p>Ejemplo:</p> <pre data-bbox="427 331 1366 360">Abrir i</pre> <p>En el ejemplo se indicaría que el enlace hace referencia a una imagen de tipo PNG</p>
rel	<p>Informa sobre la función del enlace. Puede ser:</p> <p>alternate. Enlace alternativo</p> <p>author.</p> <p>bookmark. Página de marcadores</p> <p>help. Página de ayuda</p> <p>license. Información sobre la licencia</p> <p>next. Si nuestra página pertenece a una serie ordenada, el enlace nos lleva al siguiente elemento dentro de la serie.</p> <p>nofollow. Marca que los robots de búsqueda de empresas como Google no tengan en cuenta los enlaces externos y así evitar que dichos enlaces en las páginas se utilicen para subir su calificación en los buscadores. Así se ignoran por los robots los enlaces marcados de esta forma.</p> <p>noreferrer. Un <i>referrer</i> es la información relativa a la página desde la que procede el visitante a un sitio. Con este valor en los enlaces, no se indicará al destino URL la página desde la que procedía el usuario.</p> <p>prefetch. Permite descargar el enlace antes de que el usuario haga clic en él y así acelerar su carga. Se usa (aunque pocos navegadores soportan este valor) en enlaces de uso habitual.</p> <p>prev. Si nuestra página pertenece a una serie ordenada, el enlace nos lleva al elemento anterior dentro de la serie.</p> <p>search. Página de búsqueda dentro de nuestro sitio web.</p> <p>tag. Página con etiquetas de temas (tags) de nuestro sitio web.</p>
download	<p>Nuevo atributo de HTML 5 (no funciona ni en Internet Explorer ni en Safari, ni en cualquier versión de navegador que no sea bastante moderna) que permite indicar que al hacer clic en el enlace, no se abrirá en el navegador, sino que se descargará en nuestro directorio predeterminado de descargas. Ejemplo:</p> <pre data-bbox="427 1794 1366 1854"> Descargar manual</pre> <p>Se puede indicar un valor para el atributo que se tomará como el nuevo nombre del archivo descargado:</p> <pre data-bbox="427 1977 1366 2007">D</pre>

atributo	significado
	Aunque el nombre original del archivo que se descarga en el servidor es <i>manual3232001.pdf</i> , al descargar el nombre del archivo será <i>manual.pdf</i>

Tema 4 Introducción a CSS3

Introducción

Limitaciones de HTML

HTML, en especial en las últimas versiones, es un lenguaje descriptivo. Es decir; indica qué elementos hay en una página web, pero no la forma de presentarlos en pantalla. En sus primeras versiones todas las posibilidades de formato se conseguían mediante etiquetas. Así existían etiquetas para modificar el tipo de letra (**font**), uso de maquetaciones complejas (**frame**), etc. Es decir, ante una nueva necesidad de formato, se inventaba una nueva etiqueta.

Hoy en día (y ya desde hace muchos años) se ha entendido que ese no es el modelo lógico para desarrollar páginas web. Se ha entendido que HTML no es el lenguaje que se debe de encargar de indicar cómo se deben formatear los datos. Las razones de no usar HTML para realmente dar formato a la página son:

Utilizando un lenguaje diferente que se encargue del formato, podremos reutilizar dicho formato fácilmente para diferentes páginas, manteniendo una apariencia coherente y armoniosa en todas las páginas de nuestro sitio web.

Hace que HTML sea un lenguaje puramente semántico, simplemente para dotar de significado al contenido. De esa forma nuestro texto tiene más calidad para ser buscado de forma apropiada mediante los buscadores de Internet u otras herramientas.

De usar sólo HTML, habría infinidad de etiquetas y una gran probabilidad de que cada navegador use las suyas propias (como ha ocurrido durante tantos años).

Se independiza el formato y el contenido. Así podemos usar un lenguaje para dar formato mucho más poderoso. Es más, si en el futuro se inventan mejores lenguajes que el actual dominante CSS, sería fácil adaptar nuestras páginas a esos lenguajes

Usando como único lenguaje HTML, el formato de la página web la deciden los navegadores. El tipo de letra, colores, tamaño, espacios entre párrafo,... todo eso escapa al control de HTML. En general todos los navegadores usan los mismos formatos básicos (por ejemplo la letra del párrafo normal es de tipo **Times** en todos los navegadores), pero hay pequeñas diferencias que a veces hacen que el aspecto de una página web sea muy diferente de un navegador a otro. CSS permite una apariencia más exacta en cada navegador.

Ayuda de CSS

CSS es la abreviatura de **Cascade Style Sheets** (*Hojas de Estilo en Cascada*) y se trata de un lenguaje de texto que se incrusta en las páginas web para definir el formato de la página. Actúa sobre los elementos HTML definiendo la forma en la que se mostrarán en pantalla (o en otros dispositivos).

Es capaz de actuar sobre todas las etiquetas del mismo tipo o sobre unas concretas. Se puede almacenar en un archivo aparte que después se puede usar para varias páginas a la vez. De modo que si cambiamos algo en el estilo, al instante se reflejará en todas las páginas.

CSS por lo tanto facilita la homogeneidad de las páginas y su mantenimiento. Hoy en día se considera una técnica imprescindible para dar formato a las páginas web. Además se puede aplicar también a código XML.

No es posible hoy en día crear páginas web sin utilizar CSS.

Versiones de CSS

CSS se ideó a mediados de los años 90 y se ha ido estandarizando. A día de hoy se habla de tres versiones de CSS:

CSS1. Es la versión original de CSS. Estandarizada en 1996 por la [W3C](#) (organismo de estándares oficial de Internet) incluye formatos de texto, párrafo, márgenes, lista, tamaños de imágenes,...

CSS2. Es estándar desde 1998. Amplía el CSS anterior para incluir sobre todo posicionamiento (manejo de capas), además de tipos de medios (que permite definir distintos tipos de páginas web según los diferentes medios que la usen, pantallas, impresoras, reconocedores de voz...).

La especificación [2.1](#) es el último estándar. Modificó errores de la anterior.

CSS3. Se lleva trabajando en ella desde 1998 y es un estándar *de facto*, pero que aún no ha sido aprobado por la W3C (se sigue trabajando en él) . En realidad se compone de una serie de módulos que definen diferentes especificaciones que, sumadas a CSS2 (con la que sigue siendo compatible), dan lugar a posibilidades muy avanzadas de formato. De hecho, en total hay unos 30 módulos, varios de ellos son ya considerados recomendación oficial.

En conjunto, la norma aún está en fase de borrador. Pero la mayoría de sus capacidades (estén o no aprobadas por la W3C) están ya implementadas en los navegadores. Se puede observar el estado de estandarización de sus componentes en www.w3.org/Style/CSS/current-work

CSS3 ha supuesto un auténtico pistoletazo de salida para webs con una apariencia más espectacular y vistosa, por lo que ha tenido un éxito inmediato. La mayoría de las nuevas páginas web se crean mediante CSS3 y está claro que no corre peligro de que se deje de usar en el futuro. Más bien al contrario: se seguirá ampliando y será cada vez más usado.

Nuevos elementos CSS

Se puede considerar que las normas CSS1 y CSS2 (hasta CSS 2.1) están ya soportadas por todos los navegadores actuales. Sin embargo CSS3 no está del todo adoptado. De hecho, jamás lo estará ya que la norma no está cerrada y aparecen nuevas capacidades (nuevos módulos) constantemente.

El proceso de adopción de nuevas propiedades para CSS sigue este proceso:

[1] Desde foros o medios influyentes se propone una posible nueva propiedad CSS.

[2] La empresa fabricante de un navegador influyente ([Mozilla](#), [Google](#), [Microsoft](#), etc.) adopta de forma personal dicha propiedad que solo funcionará en algún navegador.

[3] La propiedad se hace popular entre los desarrolladores y cada vez más navegadores la soportan (aunque sea de forma personal).

[4] Si las entidades normalizadoras (como el [W3C](#)) lo estiman conveniente pasa a ser parte del estándar.

Estandarización de características CSS

Que una característica forme parte del estándar tiene un largo proceso en el que esa característica pasa por diferentes fases. El estado actual de estandarización de los diferentes elementos de CSS está disponible en la URL <https://www.w3.org/Style/CSS/current-work>.

Para conseguir que una característica (o más bien un módulo de CSS, una serie de características) sea estándar, hay un proceso rígido en el que se debe pasar por estas fases:

[1] *First Public Working Draft*. **Primer borrador de trabajo**. Un grupo de trabajo del organismo W3C recoge propuestas detectadas en la comunidad sobre nuevas características a incluir en la norma CSS y publica un primer borrador con la propuesta. Se suelen agrupar un conjunto de características en **módulos CSS**, la propuesta hace referencia a un módulo concreto. El documento-borrador lo publicita a entidades, otros grupos de trabajo de la W3C y al público en general.

[2] *Working Draft*. **Borrador de trabajo**. Cuando ya se ha trabajado en el borrador anterior. Se recoge este trabajo y el nuevo módulo CSS pasa a estar en estado de Borrador de Trabajo (*Working Draft*). Este documento se puede revisar por el público y organismos técnicos antes de decidir que pase al estado siguiente.

[3] *Last Call*, **Última llamada**. Se pasa a este estado cuando el W3C indica que una fecha en la que el módulo CSS pasará a ser una característica recomendada por el organismo. En este estado de última llamada se avisa de que los comentarios de la comunidad deben hacerse antes de esa fecha.

[4] *Candidate recommendation*. **Recomendación candidata**. La propuesta pasa a poseer un documento técnico revisado que cumple la forma de la W3C para las recomendaciones. En este estado ya hay plazos para que la propuesta pase a ser una propuesta formal del organismo, salvo que se detecten problemas en la misma.

[5] *Proposed Recommendation*. **Recomendación propuesta**. En este estado, la propuesta ha sido totalmente aceptada por el W3C. En este caso habrá ya una fecha definitiva para que la propuesta pase a ser una recomendación completa de la W3C. No se pueden hacer cambios significativos en este estado de la propuesta, salvo que se publique otra recomendación candidata u otro borrador.

[6] **Recomendación**. Cuando una propuesta pasa a este estado, se considera que ya es estándar y utilizable por todos los desarrolladores y herramientas.

[7] *Rescinded recommendation*. **Recomendación escindida**. A este estado pasan recomendaciones obsoletas que no están en uso, bien por la aparición de otros elementos que la mejoran o bien porque se detecta que su uso no ha sido positivo.

Compatibilidad

Independientemente de si una característica de CSS pasa a ser estándar o no, para determinar si una página web creada por nosotros debe de utilizar esa característica, conviene tener claro qué navegadores la soportan y cuáles no.

Algunas páginas como <http://css3test.com/> y <http://caniuse.com> permiten comprobar qué parte (incluso qué porcentaje) de CSS3 reconoce cada navegador (especialmente *caniuse.com*).

Muchos navegadores incorporan elementos CSS nuevos, antes de que el estándar los recoja. Para ello, las propiedades CSS nuevas que incorpora un navegador concreto y que no son estándares, se acompañan de un prefijo.

Los prefijos habituales son:

-moz-. Para navegadores que usan el motor [Mozilla](#), como [Firefox](#).

-webkit-. Para navegadores que usan el motor [Webkit](#), como [Safari](#) o [Chrome](#).

-o-. Para el navegador Opera. Aunque desde el año 2013 utiliza webkit como motor de renderizado.

-ms-. Para navegadores con motor de [Microsoft](#), como [Internet Explorer](#).

Así para algunas propiedades que están en proceso de desarrollo podemos encontrarnos código CSS como este:

```
div{  
    -moz-columns:3;  
    -webkit-columns:3;  
    columns:3;}  
}
```

La razón es que los navegadores Mozilla y los que utilizan el motor Webkit (como Chrome) en algunas versiones implementaban las columnas (característica bastante reciente de CSS) de forma propia. Por ello se indica la propiedad con el prefijo y finalmente sin él para que los navegadores que han adoptado esta propiedad de forma estándar la lean correctamente. Además la forma estándar siempre es la última que se indica para que sea la que prevalece (los navegadores que no la reconozcan, no la tendrán en cuenta).

Otra posibilidad es detectar el navegador de usuario (para lo que hace falta código en JavaScript) e indicarle que se actualice a uno más adecuado para nuestra página web. Esto último se puede hacer también con utilidades gratuitas como por ejemplo la disponible en outdatedbrowser.com/es.

Finalmente hay que señalar que hay navegadores que permiten probar nuevas características de CSS antes de que sean adoptadas por los navegadores comerciales (para que los desarrolladores se familiaricen con ellas).

Así, Google posee el navegador [Chrome Canary](#) y Mozilla posee hasta tres: [Firefox Nightly](#) (permite probar de forma prematura nuevas características), [Firefox Developer Edition](#) (que posee toda una caja de herramientas para desarrolladores, además de incorporar CSS aun no disponible en la versión final) y [Firefox Beta](#) (que permite probar la siguiente versión de Firefox antes de que sea definitiva).

Sintaxis básica de CSS

CSS es un lenguaje distinto de HTML. Es muy sencillo, pero su dificultad radica en que es muy extenso.

CSS sigue esta sintaxis fundamental:


```
selector {  
    propiedad1:valor1;  
    propiedad2:valor2;  
    ...  
}
```

El **selector** nos permite indicar a qué elemento (o elementos) de la página web se va a aplicar el formato CSS.

El **formato** se indica mediante un conjunto de propiedades y valores. Esas propiedades permiten indicar qué formato estamos cambiando y los valores indican la modificación que hacemos a esa propiedad.

Ejemplo:

```
p{  
    font-size:14pt;  
    color:red;  
}
```

En ese código CSS, el selector es el elemento **p** (que seleccionaría todos los párrafos normales del documento). Las propiedades que se cambian es el tamaño de letra (*font-size*) y el color. Se conseguirá por tanto que los párrafos salgan con tamaño de letra de 14 puntos y color rojo.

Comentarios

Dentro del código CSS se pueden colocar comentarios. Para ello el texto del comentario se encierra entre los símbolos **/*** y ***/**. Ejemplo:

```
p {  
    line-height:10pt;  
    /*El siguiente código marcará el texto subrayado*/  
    text-decoration:underline;  
    text-align:center;  
}
```

Inserción de código CSS

Utilizado de esta forma, el código CSS se aplicará a la etiqueta en la que se ha incluido dicho código. Eso se consigue gracias a que todos los elementos de HTML (*p*, *strong*, *abbr*, *h1*,...) pueden utilizar un atributo llamado **style**, dentro del cual se coloca el código CSS. Ejemplo:

```
<!doctype html>  
<html lang="es">  
  <head>  
    <meta charset="UTF-8">  
    <title></title>  
  </head>  
  <body>  
    <p>Este texto sale de color negro (normal)</p>  
    <p style="color:red;">Este texto sale de color rojo </p>  
  </body>  
</html>
```

En este caso el código CSS no lleva selector alguno, porque el estilo se aplica al elemento en el que se incrustó el código CSS. El resto del documento no queda afectado por el código CSS.

En este caso el código CSS se inserta debajo del elemento HTML `style` que se colocará en la zona de cabecera (`head`). Este código afectará a toda la página web. Ejemplo:

```
<!doctype html>
<html lang="es">
  <head>
    <title></title>

    <style type="text/css">
      p{
        color:red;
      }
    </style>
  </head>
  <body>
    <p>Este texto sale de color rojo</p>
    <p>Este texto también sale de color rojo </p>
  </body>
</html>
```

Con ese código todos los elementos de tipo `p` se mostrarán en color rojo ya que es a esa etiqueta (mediante el selector `p`) al que se ha aplicado el formato.

El elemento HTML `style` usa fundamentalmente dos atributos:

type. Que hoy en día siempre contiene el valor `text/css`. En el caso de que apareciera otro lenguaje de estilos su contenido sería el tipo MIME correspondiente a ese lenguaje.

En la versión HTML 5 ha dejado de ser obligatorio, por lo que hoy en día no se usa (se da por hecho que el lenguaje de formato de páginas es CSS).

media. Identifica para qué tipo de dispositivo de salida se aplican los estilos. Podemos diseñar diferentes estilos de salida, dependiendo del medio por el cual estamos viendo el contenido de la página. Sus posibilidades se explican en capítulos posteriores. Si no se indica este atributo, se entiende que la hoja de estilos se aplica para cualquier tipo de medio de salida.

En un archivo externo

Es el caso más habitual. Se trata de crear un archivo de texto con extensión `.css` al que se le inserta código CSS. La ventaja es que el mismo archivo nos sirve para aplicar su código a diferentes documentos. De esa forma conseguimos una mayor coherencia estética y una centralización del formato; ya que, si modificamos el archivo CSS, todos los documentos que utilicen esa hoja de estilo aplicaran automáticamente los cambios.

La forma de incorporar el código CSS de un archivo externo es mediante el elemento `link` (se sitúa dentro del elemento `head`).

Por ejemplo, si hemos creado un archivo de texto css llamado *estilo1.css* con este contenido:

```
/*Contenido del archivo estilo1.css */
```

```
p{
    color:red;
}
```

Suponiendo que lo guardemos en el mismo directorio que la página web a la que se aplica, el código que permite aplicar dicho código CSS es:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title></title>
    <link rel="stylesheet" href="estilo1.css"
        type="text/css">
  </head>
  <body>
    <p>Este texto sale de color rojo</p>
    <p>Este texto sale de color rojo también </p>
  </body>
</html>
```

La etiqueta `link` tiene los atributos:

href. Con el que se indica la ruta de la hoja de estilos que se está incluyendo.

rel. Contiene siempre el valor `stylesheet` (para indicar que estamos incluyendo una hoja de estilos, ya que este atributo se puede utilizar para otros fines).

type. Hoy en día no se suele usar. Se aparece, se indica el valor `text/css`.

media. Comentado en el apartado anterior, indica el tipo de medio de salida para el que se aplicará el código CSS (se explica más adelante).

Orden de aplicación de estilos

Es posible que algunas páginas web utilicen varios estilos referidos al mismo elemento. Ejemplo:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Página con estilos</title>
    <style type="text/css" media="screen">
      p{
        color:blue;
      }
    </style>
  </head>
  <body>
    <p style="color:red;">Hola</p>
  </body>
</html>
```

En el código anterior, la cuestión es en qué color saldrá el texto *Hola* , teniendo en cuenta que por un lado saldría de color azul (ya que la etiqueta *p* hemos marcado que tenga ese color y hemos indicado rojo en el atributo *style* de dicha etiqueta).

La respuesta es: de color rojo, porque tiene prioridad la definición más restrictiva. Las reglas de aplicación de estilos son (se indica cuáles se aplican a antes y cuáles después):

[1] Los **estilos del navegador**. Es decir el formato predefinido del navegador. Todos los navegadores poseen un estilo predefinido, que dicta con que tamaño por defecto se muestra el texto, los colores, el tipo de letra, etc. Estos estilos son los que se aplican en primer lugar.

[2] Después se aplican los **estilos externos** (los que se incorporan con la etiqueta *link*).

[3] Después los que proceden de la **etiqueta style** en la cabecera del documento.

[4] Después los que se definan internamente en el elemento (mediante el **atributo style**).

[5] En caso de dos estilos referidos al mismo elemento y definidos en el mismo ámbito (por ejemplo ambos procedentes de archivos externos e incluidos con el elemento *link*) tiene preferencia **el último que aparece** en el código.

[6] Tienen preferencia siempre los **estilos definidos sobre los elementos más internos**

Ejemplo:

```
<style>
    strong{
        color:red;
    }
    p{
        color:green;
    }
</style>
<body>
    <p>Soy verde<strong>Soy rojo</strong></p>
</body>
```

El elemento **strong** aparece de color rojo ya que es más interno que el elemento **p**.

alteración del orden de aplicación de estilos

Se puede alterar la preferencia por defecto utilizando una palabra clave: **!important**. Los estilos marcados con **!important** tienen preferencia sobre cualquier otro. Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
```

```

<title>Prueba important</title>

<style>

    strong{
        color:green !important;
    }

</style>

</head>

<body>

    <h1>Color</h1>

    <p>Esto es una
    <strong style="color:blue">prueba</strong>

    </p>

</body>

</html>

```

El texto *prueba* aparece de color verde, aunque el atributo `style` tiene prioridad sobre el elemento del mismo nombre. La razón es que `!important` altera ese orden.

Herencias

Hay que tener en cuenta que hay etiquetas que son *padre* de otras. Es decir etiquetas que contienen a otras. En el ejemplo:

```
<p>Arturo Herrero: <em>Los años veinte</em></p>
```

La etiqueta `p` es padre de la etiqueta `em` (`em` está dentro de `p`). Esto hace que `em` herede todo el estilo que posea `p` y además añada el suyo propio. Por ejemplo, si hemos definido:

```

p{
    color:blue;
    font-size:12pt;
}
em{
    font-size:14pt;
}

```

En el ejemplo anterior, los años veinte tendrán color azul, heredado del párrafo en el que se encuentra el texto de tipo `em`, y tamaño 14, ya que tiene prioridad el tamaño asignado al elemento `em`.

Normalizar. estilo predefinido

Una página HTML que no use código CSS, mostrará el contenido utilizando el formato predefinido en el navegador.

Esto significa que el navegador decide el estilo de la página. Corremos, por tanto, el riesgo de que nuestra página se muestre distinta en diferentes navegadores porque apliquen diferente estilo inicial.

Para no correr riesgos, muchos diseñadores de páginas web, crean un primer estilo *normalizador*. Esto se realiza creando una primera hoja de estilos que contenga el formato inicial deseado para cada elemento HTML (o al menos para cada elemento que utilicemos en nuestras páginas).

Un opción muy habitual es utilizar como normalizador, la librería [normalize.css](http://necolas.github.io/normalize.css/) disponible en <http://necolas.github.io/normalize.css/> que es una hoja de estilos ya preparada para normalizar el funcionamiento de todos los elementos HTML5 en cualquier navegador. Con ella se asegura que el aspecto inicial es el mismo en cualquier navegador, además de conseguir que la apariencia inicial sea más moderna.

Indicación de la codificación de caracteres del archivo CSS

Como se ha visto, en HTML 5 la etiqueta:

```
<meta charset="utf-8">
```

Indica que el documento HTML utiliza la codificación de caracteres UTF-8. Si tenemos un archivo CSS que dentro utiliza caracteres fuera del ASCII (algo que no ocurre a menudo), disponemos de un elemento en el lenguaje CSS que permite avisar de la codificación del archivo. Se trata de `@charset`.

Esta instrucción CSS debe de ser la primera del archivo y basta simplemente con indicarla y, seguidamente, añadir la codificación utilizada. Por ejemplo:

```
@charset "utf-8"
```

Selectores CSS3

Los estilos CSS se aplican hacia el elemento (o elementos) HTML que indiquemos, por ello es muy importante utilizar correctamente los selectores.

Un selector es el instrumento que proporciona CSS para determinar qué partes de la página web se van a ver afectados por el código CSS que indiquemos.

Podemos indicar los elementos a los que aplicar un determinado código CSS, por ejemplo, con el nombre del elemento (por ejemplo, `h1`), pero también podemos hacer selecciones más elaboradas para seleccionar un elemento concreto mediante su identificador, seleccionar elementos mediante su jerarquía en la página, mediante el valor de sus atributos, etc..

Hacer buenas selecciones nos permitirá conseguir hojas de estilos muy sofisticadas, coherentes y fáciles de mantener. Lo ideal es que HTML y CSS funcionen de forma independiente, de forma que cuando queramos dar formato a nuestros documentos, no tengamos que tocar el código HTML.

Eso solo se puede conseguir estructurando bien a nivel semántico nuestro HTML y dominando el uso de selectores.

Selectores simples

Selección de elementos completos HTML

Podemos aplicar un estilo a los elementos de un tipo de etiqueta concreta de HTML. Ejemplo:

```
p{  
  color:blue;
```

```
font-size:12pt;
}
```

En principio, con ese código todos los elementos de tipo `p` de la página saldrán de color azul.

Podemos incluso aplicar el estilo a varias etiquetas a la vez:

```
h1,h2,h3{
  color:blue;
}
```

Los títulos de tipo `h1`, `h2` o `h3` saldrán de color azul.

Selección por clases de elementos

Una de las primeras formas que tiene CSS para diferenciar elementos del mismo tipo (por ejemplo un párrafo de otro) son las clases.

Una clase es un nombre que asignamos a una serie de propiedades y valores CSS. Posteriormente simplemente aplicaremos el nombre de la clase a aquellos elementos que deseamos se modifiquen. La forma de configurar, es:

```
selector.nombreclase{
  propiedad1:valor1;
  propiedad2:valor2;
  ...
}
```

Por ejemplo:

```
p.rojo{
  color:
}
```

Para que un párrafo (necesariamente marcado con la etiqueta `p`) adopte este estilo (y por lo tanto coloree la letra de color rojo) hay que indicarlo gracias a un atributo presente en todos los elemento HTML; se trata del atributo `class`. Ejemplo:

```
<p
  Este                texto                sale                de
</p>
```

Podemos definir la clase sin indicar a qué tipo de elementos se aplica, de ese modo podremos aplicar la clase a cualquier elemento:

```
.rojo{
  color:red;
}
```

Cualquier elemento HTML podrá utilizar esa clase. Ejemplo:

```
<h1 class="rojo">  
    Título con letras rojas
```

```
</h1>
```

```
<p class="rojo">  
    Párrafo con letras rojas
```

```
</p>
```

En el ejemplo tanto los elementos `h1` como los `p` de clase `rojo`, aparecerán con la letra en rojo.

Hay que tener en cuenta que podemos aplicar más de una clase al mismo elemento:

```
/* CSS */
```

```
.rojo{
```

```
    color:red;
```

```
}
```

```
.borde{
```

```
    border-bottom:1px solid black;
```

```
}
```

```
.....
```

```
...
```

```
<h1 class="rojo borde">  
    Título de color rojo y con borde
```

```
</h1>
```

En este último caso se suman al elemento las propiedades y valores de ambas clases definidas. En el caso de que las dos clases modifiquen las mismas propiedades (por ejemplo el color de la letra), se aplican las propiedades de la última clase definida en el código.

También en CSS es posible aplicar formato cuando un determinado elemento tenga dos clases a la vez. Eso se hace usando como selector las dos clases seguidas (sin separarlas con un espacio):

```
/* CSS */
```

```
.rojo.borde{
```

```
    color:red;
```

```
    border-bottom:1px solid black;
```

```
}
```

```
.....
```

```
...
```



```
<h1 class="rojo">
```

No tendrá borde ni color rojo

```
</h1>
```

```
<h1 class="rojo borde">
```

Este sí es título de color rojo y con borde

```
</h1>
```

Selección por identificadores

La idea es parecida a la de las clases, pero en este caso se usa el valor del atributo **id** (disponible en todos los elementos del lenguaje HTML). El valor de este atributo no se puede repetir en un documento (aunque los navegadores lo permitan).

Para indicar el valor se usa el símbolo # seguido del identificador.

Ejemplo:

```
/* CSS */
```

```
#parrafo1{
  color: #339999;
  background-color: #D6D6D6;
}
```

```
.....
```

```
...
```

```
<p id="parrafo1">
  texto del párrafo coloreado
```

```
</p>
```

Selector de limitación

Permite indicar que el estilo definido se aplica a una determinada etiqueta pero cuando esté contenida en otra. Ejemplo:

```
td p{
  color:red;
}
```

Se aplica a los elementos de tipo **p** cuando están dentro de elementos de celda (**td**). Ejemplo:

```
<p>Este texto sale normal <p>
```

```
<table>
```

```
  <tr>
```

```
    <td>Sale normal</td>
```

```
  <td><p>Sale de color rojo</p></td>
```

Podemos hacer limitaciones más complejas, por ejemplo:

```
section article p{
    color:red;
}
```

El código anterior se aplicará a los párrafos que estén dentro de un artículo que, a su vez, estén dentro de una sección.

Esta técnica vale para cualquier selector:

```
section .rojo{
    color:red;
}
```

En este caso se colorean de rojo los elementos de clase **rojo** interiores a un elemento **section**.

Selector universal

Permite aplicar un estilo a todas las etiquetas del documento. Esto lo consigue el asterisco. Ejemplo:

```
*{
    color:black;
}
```

No se suele utilizar de esa forma ya que es demasiado indiscriminada. Pero sí se utiliza para elementos de este tipo:

```
<!doctype html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <title>Documento</title>

    <style>

        table *{

            color:red;

        }

    </style>

</head>

<body>

    <p>Sale normal</p>

    <table>

        <tr>
```

```
        <th>Sale rojo</th>
    </tr>
    <tr>
        <td>Sale rojo</td>
    </tr>
</table>
</body>
</html>
```

Permite colorear en rojo el texto que esté dentro de una tabla; no importará que etiqueta sea la que está dentro de la tabla. Otro ejemplo:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Documento</title>
    <style>
        p *{ color:red;}
    </style>
</head>
<body>
    <p>Sale normal <span>Sale rojo</span></p>
</body>
</html>
```

Ahora sale de color rojo el texto dentro de la etiqueta *span*.

Selección por atributos

Permite aplicar estilos a un elemento cuando este tiene un determinado valor sobre un atributo. Para ello se indica el atributo entre corchetes, seguido del signo de igualdad y el valor entre comillas. Ejemplo:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
```

```

<title>Documento</title>

<style>

    p[lang="en"]{
        font-style: italic;
    }

</style>
</head>
<body>
    <p lang="es">texto que sale de forma normal</p>
    <p lang="en">texto que sale en cursiva</p>
</body>

```

Se puede mezclar este tipo de definiciones con clases o definiciones por identificador:

```

p.clase1[lang="en"]{
    font-style: italic;
}

```

En este caso el estilo definido se aplica a párrafos de **clase 1** que estén marcados con el valor *en* en el atributo *lang* (es decir que estén en inglés).

Se puede incluso utilizar más de un atributo:

```

p[lang="en"][spellcheck="true"]{
    font-style: italic;
}

```

En este caso se aplica el estilo para los elementos de tipo *p* que usen los atributos *lang* y *spellcheck* con los valores indicados.

Podemos incluso usar varios valores para el mismo atributo:

```

p[lang="en"][lang="fr"]{
    font-style: italic;
}

```

Aplica la cursiva a párrafos cuyo lenguaje sea inglés o francés.

También podemos indicar el estilo simplemente para los elementos que usen el atributo independientemente de su valor:

```

<!doctype html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <title>Documento</title>

    <style>

```

```

        p[lang]{
            font-style: italic;
        }
    </style>
</head>
<body>
    <p>Este texto se muestra de forma normal</p>
    <p lang="en">This text is shown with italics (se muestra con
        cursiva)</p>
    <p lang="es">Este texto también se muestra en cursiva</p>
</body>

```

Por otro lado gracias a CSS3 disponemos de estas posibilidades:

sintaxis	significado
<i>elemento[atributo~="valor"]</i>	Elementos que usen el atributo indicado que contengan el valor aunque separado de otros valores por espacios
<i>elemento[atributo\$="valor"]</i>	Elementos que utilicen el atributo y cuyo contenido finalice con el valor indicado
<i>elemento[atributo^="valor"]</i>	Elementos que utilicen el atributo y cuyo contenido empiece con el valor indicado
<i>elemento[atributo = "valor"]</i>	Elementos que utilicen el atributo, cuyo contenido empiece con el valor indicado y además ese valor sea una palabra
<i>elemento[atributo*="valor"]</i>	Elementos que utilicen el atributo indicado y contengan (en cualquier parte) el atributo indicado

Ejemplo:

```

<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Documento</title>
    <style>
        p[lang*="es"]{
            font-style: italic;
        }
    </style>

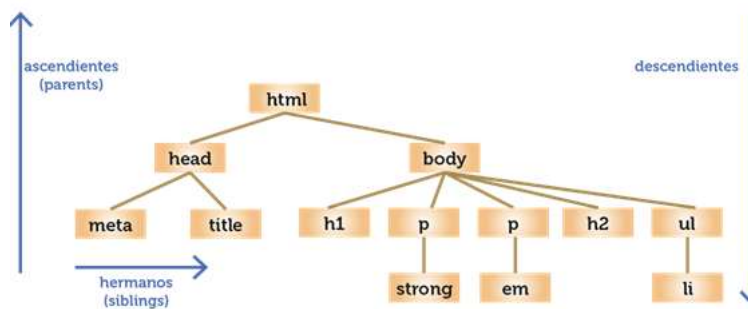
```

```

    }
  </style>
</head>
<body>
  <p lang="es-AR">Este texto se muestra en cursiva
</p>
  <p lang="es-ES">Este texto también se muestra en
  cursiva</p>
  <p lang="es">Incluso este también</p>
  <p lang="en">Este no</p>
</body>

```

Selectores jerárquicos



Podemos entender HTML como un documento formado de manera jerárquica, donde hay elementos que contienen otros elementos formando una estructura de árbol. Esta es una parte importante del funcionamiento tanto de una página web HTML como de un documento XML. Esta visión permite seleccionar elementos que casen con una cierta jerarquía.

De este modo esta página web:

```

<!doctype html>
<html lang="es-ES">
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <h1>Monumentos de Palencia</h1>

```

<p>Palencia dispone de numerosos monumentos interesantes. Debido a su importante actividad medieval posee una gran cantidad de edificios religiosos entre los que destaca la Catedral de Palencia así como la iglesia de San Miguel

con su peculiar torre de defensa y el monumento más conocido de la ciudad: El Cristo del Otero</p>

<p>A finales del siglo XIX y principios del XX aparecieron edificios suntuosos y civiles que han embellecido una buena parte de la ciudad, en especial la transitada Calle Mayor.</p> <h2>Edificios religiosos</h2>

```
<ul>
  <li>Cristo del Otero</li>
  <li>Catedral Mayor</li>
  <li>Iglesia de San Miguel</li>
  <li>Iglesia de San Lázaro</li>
  <li>Convento de San Pablo</li>
  <li>Iglesia de la Compañía</li>
</ul>

</body>

</html>
```

En él se observa como los elementos *body* y *head* son hijos de *html*. Mientras que *meta* y *title* son hijos de *head* (luego nietos de *html*). Los nodos al mismo nivel forman hermanos (*h1*, *h2*, *p* y *ul* en este esquema son hermanos).

Selector para un descendiente

Es posible aplicar un formato cuando un elemento está contenido en otro (es su descendiente).

La forma (vista anteriormente es):

```
section p{
  color:red; /* Selecciona párrafos p si están dentro de una tabla
}
```

Ese código colorea de rojo el texto contenido en elementos de párrafo (de tipo *p*) cuando son **descendientes** de algún elemento **section** (sea *hijo* o *nieto*).

Así usando el CSS anterior este código HTML:

```
<section>
  <h1>Título</h1>
  <p id="p1">Sale de color rojo</p>
</section>
<article>
  <p id="p2">También sale de color rojo</p>
</article>
</section>
```

Se colorean ambos párrafos de tipo *p*.

Selector de elementos hijos

En este caso, solo funciona para elementos que son hijos directos. Se hace de esta forma:

```
section > p {
  color:red;
}
```

El signo **>** indica una relación jerárquica de padre a hijo. En el texto HTML del ejemplo anterior solo se colorearía el primer párrafo:

```
<section>
```

```
  <h1>Título</h1>
```

```
  <p id="p1">Sale de color rojo</p>
```

```
</article>
```

```
  <p id="p2">No sale de color rojo</p>
```

```
</article>
```

```
</section>
```

Otros selectores jerárquicos

sintaxis	significado
elemento1 + elemento2	El estilo se aplica al elemento2 cuando es hermano del elemento1 y además el elemento1 precede inmediatamente al elemento2 .
elemento1 ~ elemento2	Se aplica al elemento2 cuando es hermano del elemento1 y éste le precede, aunque no sea inmediatamente.
elemento:empty	Se aplica cuando el elemento está vacío
elemento:nth-child(número)	<p>Se aplica al elemento indicado cuando sea el hijo con el número indicado (por ejemplo número sería 3, para el tercer hijo).</p> <p>Se pueden utilizar expresiones más complejas mediante el uso de la variable n para conseguir fórmulas más complejas (la n en el navegador la sustituirá automáticamente por cada número entero para calcular qué filas se seleccionan)</p> <p>Se permite también usar las palabras clave odd (<i>impar</i>) y even (<i>par</i>)</p> <p>Ejemplos de uso:</p>

sintaxis	significado
	<p><code>tr:nth-child(3)</code>. Selecciona la tercera fila de una tabla</p> <p><code>tr:nth-child(odd)</code>. Selecciona las filas impares</p> <p><code>td:nth-child(even)</code>. A los pares</p> <p><code>tr:nth-child(2n+1)</code>. Selecciona las filas 1,3,5,... (las impares)</p> <p><code>tr:nth-child(3n+1)</code>. Selecciona las filas 1, 4, 7, 10, 13... (de tres en tres)</p>
<code>elemento:nth-last-child(n)</code>	Funciona igual que el anterior pero cuenta los elementos hijos a partir del último (es decir, el orden de atrás hacia delante).
<code>elemento:first-child</code>	Se aplica al elemento cuando es el primer hijo
<code>elemento:last-child</code>	Se aplica al elemento cuando es el último hijo
<code>elemento:only-child</code>	Se aplica cuando el elemento es el único hijo
<code>elemento:first-of-type</code>	Primer descendiente de su tipo
<code>elemento:last-of-type</code>	Último descendiente de su hijo
<code>elemento:nth-of-type(n)</code>	<p>Funciona como <code>nth-child</code>, pero se refiere al número de hijo de ese tipo. No cuenta hijos, sino hijos del tipo indicado.</p> <p>Ejemplo:</p> <pre><style> article p:nth-of-type(2){ color:red; } </style> </head> <body> <article> <h1> Primer hijo, no se colorea </h1> <p> Segundo hijo, primero de tipo p. No se colorea</pre>

sintaxis	significado
	<pre> </p> <p> Tercer hijo, segundo de tipo p. SE COLOREA </p> <p> Cuarto hijo, tercero de tipo p. No se colorea </p> </article> </body> </pre>
elemento: <code>nth-last-of-type(n)</code>	Como el anterior pero cuenta <i>n</i> desde el final.
elemento: <code>only-of-type</code>	Se aplica cuando el elemento es el único hijo de ese tipo

Ejemplo de uso de formatos jerárquicos

```
<style>
```

```

h1 + p{
    color:red;
}

```

```
</style>
```

```
</head>
```

```
<body>
```

```
<section>
```

```
<h1>Este es un título</h1>
```

```
<p>Párrafo que sale de color rojo</p>
```

```
<p>Párrafo que sale normal</p>
```

```
<h1>Otro título</h1>
```

```
<address>C/ Mayor...</address>
```

```
<p>Este sale normal</p>
```

```
</section>
```

```
</body>
```

En el ejemplo, el operador `+`, hace que se coloree la letra de color rojo, para todos los párrafos de tipo `p` que estén justo detrás de un párrafo `h1`. Por ello se colorea solo el párrafo con el texto *Párrafo que sale de color rojo* ya que es el único que cumple la condición. Sin embargo si se hubiera definido de esta forma:

```
<style>
```

```
h1 ~ p{
    color:red;
}
```

```
</style>
```

Ahora todos los párrafos de tipo `p` aparecen de color rojo ya que no es necesario que estén inmediatamente detrás de un `h1`, basta con que sean hermanos; es decir, que estén dentro del mismo contenedor y `h1` aparezca antes que `p`.

Ejemplo de uso de `nth-child`:

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

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>Prueba nth-child</title>
```

```
<style>
```

```
table{
    width:450px;
    border-collapse:collapse;
}
td,th{
    border:1px solid black;
}
```

```
td:nth-child(2){
    background-color:red;
```

```
</style>
```

```
</head>
```

```
<body>
  <table>
    <tr>
      <th>&nbsp;</th>
      <td>&nbsp;</td>
      <td>&nbsp;</td>
      <td>&nbsp;</td>
    </tr>
    <tr>
      <th>&nbsp;</th>
      <td>&nbsp;</td>
      <td>&nbsp;</td>
      <td>&nbsp;</td>
    </tr>
  </table>
</body>
```

La segunda columna sale con fondo rojo, ya que en ella la etiqueta `td` es el segundo hijo de la etiqueta `tr`. Pero si el código CSS fuera este:

```
<style>
  table{
    width:450px;
    border-collapse:collapse;
  }
  td,th{
    border:1px solid black;
  }
  td:nth-of-type(2){
    background-color:red;
  }
```

Porque ahora se colorea la etiqueta `td` cuando es el segundo hijo de su tipo (la primera columna es de tipo `th` y no `td`).

Podemos incluso hacer combinaciones avanzadas. Ejemplo:

```
ul > li + li {  
    color:green;  
}
```

Se aplica a los elementos `li` que estén dentro de elementos `ul` y además estén inmediatamente precedidos por otro elemento `li`. Ejemplo:

```
<ul>  
    <li>Yo salgo normal</li>  
    <li>Yo salgo verde</li>  
    <li>Yo también salgo verde</li>  
</ul>
```

Pseudoclases

Básicas

Las pseudoclases permiten asociar estilos a un selector cuando le ocurre una determinada circunstancia. Inicialmente las pseudoclases se aplicaban a los enlaces (elemento `a`), pero ahora se aplican a cualquier elemento. Las pseudoclases clásicas son:

`:link`. Se aplica para los enlaces no visitados

`:visited`. Enlaces visitados

`:active`. Enlaces activos (aquellos sobre los que hacemos clic)

`:hover`. Se aplica cuando el ratón pasa por encima del elemento (sea un enlace o no)

Ejemplo:

```
<style>  
a{  
    text-decoration:none;  
}  
a:visited{  
    color:gray;  
}  
a:hover{  
    text-decoration:underline;
```

```
}
```

```
</style>
```

En el ejemplo los enlaces a los que se aplique esta hoja de estilo saldrán sin subrayar, si están visitados saldrán de color gris y si se arrima el ratón en ellos será cuando se muestren subrayados.

Estas pseudoclases permiten un cierto dinamismo en la página HTML. Actualmente además es posible utilizarlas en otros elementos distintos de la etiqueta **a** (como **p**, **div**, etc.) lo que da enormes posibilidades.

Más pseudoclases

CSS3 ha incorporado muchas más pseudoclases:

pseudoclase	selecciona...
:focus	Cuando el elemento obtiene el foco, es decir cuando el elemento ha capturado la entrada del teclado. Muy útil en formularios.
:lang(código)	Se aplica cuando el elemento esté marcado con el lenguaje indicado por su código de idioma en el atributo lang (es para español, en para inglés,...)
:enabled	Cuando el elemento está habilitado (útil en formularios)
:disabled	Cuando el elemento está deshabilitado (útil en formularios)
:checked	En controles de formulario de tipo radio o checkbox , cuando el elemento pasa a estar activado.
:in-range	En controles de formulario (de tipo input) cuando el valor que contienen está dentro de rango. Esto significa que los atributos min y max (que solo se aplican a algunos tipos de controles input) han delimitado valores el contenido del control cumple. No funciona en Internet Explorer.
:out-of-range	Al revés del anterior. Aplica el contenido CSS a controles cuyos valores están fuera de rango.
:target	Se aplica cuando el elemento al que se refiere ha sido destino de un enlace. Ejemplo: <pre><style> #intro:target{ color:blue; } ... <h1 id="intro">Introducción</h1></pre>

pseudoclase	selecciona...
	<p>...</p> <p><code>Ir a la introducción</code></p> <p>Cuando hagamos clic sobre el enlace <i>ir a la introducción</i>, el título <i>Introducción</i> se pondrá de color azul.</p>
:valid	Controles de formulario que contienen valores válidos (lo serán o no dependiendo del tipo de elemento). Por ejemplo no será válido un texto en un cuadro <code>input</code> de tipo <code>numérico</code> (<code>number</code>).
:invalid	Inverso al anterior. Es decir, selecciona controles cuyos valores no sean válidos.
:read-only	Controles de formulario que están en modo de solo lectura (debido a usar el atributo HTML <code>readonly</code>). No funciona en Internet Explorer y en Mozilla Firefox solo si se usa con el prefijo <code>-moz-</code>
:read-write	Inverso al anterior. Controles que se puedan leer y escribir.
:required	Controles de formulario que están marcados (gracias al atributo HTML <code>required</code>) para ser rellenados obligatoriamente.
:optional	Contrario al anterior. Controles que no están obligados a ser rellenados.
:default	Selecciona los elementos de un formulario que son las opciones marcadas por defecto. Por ejemplo el botón de radio que actualmente esté activado. No funciona en Internet Explorer.
:first-line	La primera línea del elemento.
:first-letter	La primera letra del elemento.
:before	<p>Contenido anterior al elemento. Siempre se suele usar con la propiedad <code>content</code> para añadir contenido al elemento. Ejemplo:</p> <pre>p:before{ content:url('flecha.gif'); }</pre> <p>Coloca la imagen de una flecha delante del párrafo</p>
:after	Para indicar contenido después del elemento.
:lang(codigoLenguaje)	Selecciona los elementos que hayan utilizado el atributo <code>lang</code> con el código indicado. Es equivalente a <code>[lang="codigoLenguaje"]</code> .
:not(selector)	Se aplica cuando el elemento no cumple ser del tipo indicado por el selector dentro de la palabra <code>not</code> . Ejemplo:

pseudoclase	selecciona...
	<pre>div >:not(p){ color:red; }</pre> <p>El color rojo se aplicará al contenido de los elementos div salvo que estén dentro de elementos p.</p> <p>Otro ejemplo:</p> <pre>p:not(:first-of-type){ color:green; }</pre> <p>Se aplica a todos los párrafos excepto al primero de cada grupo de hermanos.</p>
:root	Solo tiene utilidad cuando se aplica CSS sobre documentos XML. Aplica el estilo al elemento cuando es raíz del documento. En un documento HTML la raíz es siempre el elemento html , por lo que carece de utilidad.
:dir(direccion)	Aunque es estándar, no lo soporta aun casi ningún navegador. Selecciona en base a la dirección en la que se escribe el texto. Esa dirección puede tomar el valor ltr (izquierda a derecha) o rtl (derecha a izquierda)
:fullscreen	Selector interesante, pero experimental que se aplica a los elementos que se muestran cuando la página HTML está en modo de pantalla completa. Todos los navegadores actuales la soportan pero si se usa el prefijo del navegador (por ejemplo -moz-fullscreen). Solo el navegador Edge de Microsoft la reconoce sin prefijo.
:left	Se aplica cuando la página se imprime. Selecciona la página izquierda (en impresión a doble cara).
:right	Se aplica cuando la página se imprime. Selecciona la página derecha (en impresión a doble cara).

Las pseudoclases se pueden combinar, de modo que se pueden construir cosas como:

```
input:focus:hover {
    background-color: yellow;
}
```

Se pondrá color de fondo amarillo, cuando el cuadro de tipo input tenga el foco y el ratón pase por encima de él. O cosas más complejas como:

```
input.clase1:focus:hover[type="password"] {
    background-color: yellow;
}
```


En este caso se aplica el fondo amarillo cuando el cursor esté encima de un control de texto de tipo contraseña que además tenga el foco y sea de *clase1*.

Pseudoelementos

Se trata una mejora de CSS3 que permite aplicar estilos a elementos, aunque estos no existan en el árbol de elementos del documento. La razón para que un elemento no esté en el árbol es que no existe en el código hasta que se ejecute, por ejemplo, una determinada sentencia en lenguaje JavaScript.

Los pseudoelementos utilizan los símbolos `::` en lugar de simplemente `:` como ocurre con las pseudoclases.

pseudoelemento	significado
<code>::first-line</code>	Primera línea del elemento
<code>::first-letter</code>	Primera letra del elemento
<code>::after</code>	Se aplica para colocar código tras el elemento
<code>::before</code>	Se aplica para colocar código antes del elemento
<code>::selection</code>	Se aplica al texto seleccionado del elemento. En el caso de los navegadores de Mozilla , requiere usar el prefijo <code>-moz-</code>
<code>::placeholder</code>	Pseudoclase experimental (solo funciona si se usa el prefijo de cada navegador) que se aplica al texto de tipo <i>placeholder</i> de los controles de los formularios.

Unidades

Unidades relativas y absolutas

CSS proporciona en realidad dos tipos de medidas: **absolutas** y **relativas**. Las absolutas utilizan magnitudes que se indican de forma independientes. El caso más claro es usar como unidad de medida el píxel.

Si por ejemplo indicamos que el texto tenga un tamaño de 12px, ocupará 12 píxeles, independientemente del medio, o de las características de la letra base.

Las unidades absolutas producen efectos no deseados: 12 píxeles puede ser un tamaño suficiente para un dispositivo de poca resolución, como un móvil de 480x320 píxeles de pantalla; pero en un dispositivo de alta calidad, como por ejemplo un *iPad retina* por ejemplo, sería un tamaño apenas legible en esa pantalla que tiene más de 300 píxeles por pulgada (12 píxeles ocupan muy poco tamaño).

Por contra, las unidades relativas hacen que el tamaño de la letra no dependa de la resolución sino de un tamaño base. Ese tamaño base depende en cada dispositivo (se supone que se adapta de forma apropiada al dispositivo) con la finalidad de que los elementos de la página se adapten a dicho dispositivo.

La desventaja de las unidades relativas es que ya no disponemos de la posibilidad de saber con exactitud el tamaño de un elemento; medirá más o menos según el tamaño base que se esté utilizando en cada dispositivo de usuario. En realidad hoy en día es una gran ventaja, ya que estamos obligados a diseñar de forma adaptativa (*responsive design*), pero es difícil de utilizar.

Unidades absolutas de medida

Esta es la lista de las unidades de medida utilizables en CSS:

in. Pulgadas. Medida muy habitual en el mundo anglosajón, equivale a 25,4 milímetros.

cm. Centímetros. No son muy útiles en la pantalla, pero sí en medios de impresión

mm. Milímetros.

pt. Puntos. Muy utilizada en tipografía. Un punto tipográfico equivale a 1/72 pulgadas. Debido a la popularidad de los procesadores de texto (que usan esta unidad de medida por defecto), es una medida que, referida a la letra, la mayoría de las personas conocen.

pc. Pica. Una pica la forman doce puntos. Seis picas es una pulgada.

px. Píxeles. Esta medida es relativa respecto al dispositivo de salida. Ya que en cada dispositivo el tamaño del píxel varía. Se usa mucho en elementos grandes (capas, tablas,)

Para tener una percepción más directa de la relación entre unidades absolutas y relativas, podemos acudir a la dirección:

http://www.w3schools.com/cssref/css_pxtoemconversion.asp

Unidades relativas de medida

em. Tamaño relativo de letra. Toma el tamaño de la M mayúscula como base. Así por ejemplo si indicamos *2em*, nuestro tamaño de letra será el doble de lo que ocupa la letra M mayúscula del tamaño base. Hoy en día es la forma recomendada para indicar medidas en el diseño de páginas web. Esta unidad toma como tamaño base, el del elemento padre del actual.

Por ejemplo, si indicamos:

```
<style>
    small{
        font-size:2em;
    }
</style>
</head>
<body>
    <h1>Título <small>Letra pequeña</small></h1>
</body>
```

Como tamaño base de *small*, se tomará el tamaño por defecto de la letra de *h1* y por lo tanto, se obtiene:

Título Letra pequeña

La letra pequeña es el doble de grande que la normal del título, ya que se toma como base el tamaño de letra del elemento **h1**.

ex. Tamaño relativo respecto a la letra equis minúscula (**x**). Así 2x indica que el tipo de letra aumenta hasta ocupar el doble la letra x mayúscula. Con 0.5x ocuparía la mitad

ch. Altura relativa al número cero (0) en la tipografía actual. No se debe utilizar, ya que la mayoría de navegadores no le reconoce y no aporta ninguna ventaja sobre las dos medidas anteriores.

%. Porcentaje. Es relativo respecto del tamaño del elemento padre del elemento al que le ponemos esta medida. Así 50% para una tabla dentro del elemento **body** ocuparía la mitad de la pantalla, pero dentro de una capa ocuparía la mitad del tamaño de la capa. Para una letra, si indicamos un 50%, la letra será la mitad de alta respecto al tamaño base del elemento en el que nos encontramos.

rem. Es muy exitosa actualmente, ya que elimina algunos problemas que tienen los diseñadores al utilizar **em** como unidad de medida. Lo malo es que no funciona en navegadores antiguos (requiere Internet Explorer 8 o superior y en los demás navegadores una versión bastante actualizada). Se basa en un tamaño relativo respecto de la raíz del documento (*root em*). Es decir funciona como la unidad *em*, pero tomando como base de la fuente la letra asignada por defecto a la raíz (al elemento *html*).

Ejemplo:

```
<style>
    small{
        font-size:2rem;
    }
</style>
</head>
<body>
    <h1>Título <small>Letra pequeña</small></h1>
</body>
```

Se obtiene:

Título Letra pequeña

Ahora la letra pequeña es del mismo tamaño, ya que se utiliza como base es el de la raíz de la página (elemento `html`) y esa base será la misma para todos los elementos. Como el elemento `h1` tiene por defecto una letra de tamaño `2em`, el tamaño de `small` y de `h1` es el mismo.

vw. Es una medida muy moderna (sólo los navegadores más actualizados la reconocen) que equivale al 1% del tamaño del ancho del área visible de la página (el llamado *viewport*). Aporta como ventaja el hecho de que se tiene en cuenta lo que mide el dispositivo, lo que le hace una medida relativa de verdadera objetividad.

vh. Como la anterior, pero se refiere al 1% de la altura total del área visible de la página.

vmin. Usa la menor medida entre **vw** y **vh**.

vmax. Usa la mayor medida entre **vw** y **vh**.

Colores

Hay numerosas etiquetas con capacidad de mostrar colores. Para indicar un color, CSS dispone de estas posibilidades:

Notación hexadecimal. Se trata de la notación más utilizada en CSS. Consiste en una cifra hexadecimal, precedida del símbolo `#`. Las dos primeras cifras hexadecimales indican el nivel de rojo, las dos siguientes el nivel de verde y las dos últimas de verde; por ejemplo `#FF0000` es el código del rojo puro. Ejemplo¹:

color	código hexadecimal	código RGB
	<code>#000000</code>	<code>rgb(0,0,0)</code>
	<code>#FF0000</code>	<code>rgb(255,0,0)</code>
	<code>#00FF00</code>	<code>rgb(0,255,0)</code>
	<code>#0000FF</code>	<code>rgb(0,0,255)</code>
	<code>#FFFF00</code>	<code>rgb(255,255,0)</code>
	<code>#00FFFF</code>	<code>rgb(0,255,255)</code>
	<code>#FF00FF</code>	<code>rgb(255,0,255)</code>
	<code>#C0C0C0</code>	<code>rgb(192,192,192)</code>
	<code>#FFFFFF</code>	<code>rgb(255,255,255)</code>

Mediante función RGB. Consiste en usar el la función `rgb(r,g,b)` donde *r* es el nivel de rojo (entre 0 y 255), *r* es el nivel de rojo y *b* el de azul. Ejemplo:

```
font-color: rgb(255,0,0);
/* Color de fuente rojo */
```

Mediante RGB con porcentaje. Funciona como la anterior, pero el nivel re rojo, verde y azul se indican con tanto por ciento. Ejemplo:

```
font-color:rgb(50%,25%,12%);  
/*el color es un rojo anaranjado*/
```

Mediante función RGB y transparencia. Permite utilizar una función llamada **rgba** que funciona como la función **rgb**, pero que añade un cuarto parámetro que es un valor de transparencia (conocido como parámetro **alpha**). Este parámetro puede tener un valor entre 0 (transparencia total y 1.0 opacidad total). De modo que el color se funde con el fondo. Es parte de CSS3 y sólo está disponible en las últimas versiones de los navegadores. Ejemplo:

```
font-color:rgba(50%,25%,12%,.5);  
/*el color es un rojo anaranjado con 50% de transparencia*/
```

Mediante función hsl. Permite seleccionar colores utilizando tres valores:

Tono (Hue). Un valor de 0 a 360 que indica el giro en la rueda de colores. Por ejemplo el cero es el rojo, el 120 el verde y el 240 el azul.

Saturación. Un número del 0% al 100%, que indica el nivel de saturación. Más saturación indica un color más vivo. Una saturación del 0% significa pasar el color a escala de grises.

Luminosidad. Un número del 0% al 100%. Indica el nivel de luminosidad del color: más luminoso significa más claridad para el color (0% significa negro).

Este modo está disponible en los últimos navegadores compatibles con CS3. Es el modelo más parecido a la forma que tiene el ser humano de percibir el color, pero no se utiliza mucho dentro de las páginas web.

Mediante función hsla. Añade la función anterior un cuarto valor (un número decimal entre 0.0 y 1.0) que sirve para indicar transparencia.

Por el nombre. Permite indicar el color por su nombre estándar Hay diecisiete colores estándares para especificar colores son:

Nombre	Código Hexadecimal
White	#FFFFFF
Silver	#C0C0C0
Gray	#808080
Black	#000000
Red	#FF0000
Maroon	#800000
Yellow	#FFFF00
Olive	#808000
Lime	#00FF00
Green	#008000

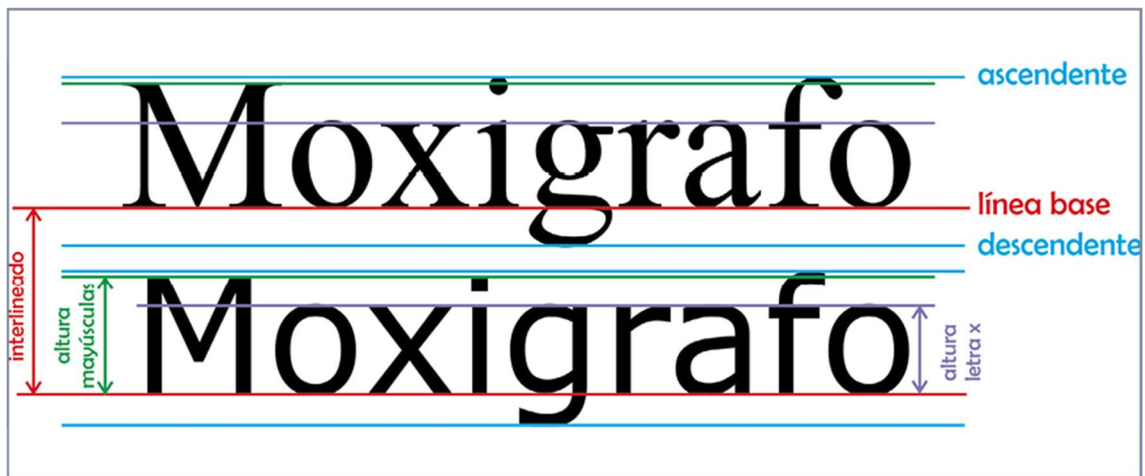
Nombre	Código Hexadecimal
Aqua	#00FFFF
Teal	#008080
Blue	#0000FF
Navy	#000080
Fuchsia	#FF00FF
Purple	#800080

Estos colores se reconocen por cualquier navegador web. Además la especificación CSS 3 referente a los colores (*CSS3 color module*), reconoce los colores **X11**. Esta lista de colores se creó para el sistema X Windows de Unix y se fue adoptando por los navegadores y por diversos lenguajes gráficos (como SVG). En la actualidad son reconocidos por la mayoría de navegadores.

- Hay diferencias entre el original X11 y la recomendación de nombres estándar de la W3C (que es la adoptada en CSS3).

Fuentes

Propiedades de las fuentes



Indudablemente la tipografía (llamada también **formato de fuente** o, simplemente, **fuente**) es uno de los aspectos más importantes de la estética de un documento escrito. Manejar adecuadamente el estilo de la letra puede hacer que nuestra página sea mucho más atractiva. De hecho hay diseñadores que consideran que esta es la primera decisión a tomar por la tremenda influencia que tiene sobre el resto de elementos. Lo ideal es dedicar mucho tiempo a esta decisión.

CSS dispone de numerosas propiedades para modificar los tipos de letra. Muchas de sus propiedades se pueden observar en la

En la imagen anterior se observan las líneas básicas que permiten encajar las letras. Los elementos a tener en cuenta son:

Línea base. Es la línea principal donde se colocan las letras. Es la línea que permite colocar adecuadamente en la horizontal al texto.

Altura de la letra x. Mide el tamaño de la letra equis minúscula, que sirve de referencia para todas las letras minúsculas (aunque algunas como la *g* y la *a* del primer ejemplo la puedan rebasar).

Ascendente. Línea superior que indica el máximo que puede ocupar las letras (en la imagen es el límite superior de la letra *f*).

Descendente. Línea inferior, posición máxima inferior que puede ocupar el texto (la letra *g* en el ejemplo es la que marca esa posición).

Altura mayúsculas. Línea superior máxima de las letras mayúsculas (se suele tomar la de la letra *M* mayúscula).

Interlineado. Distancia entre dos líneas, se toma utilizando las líneas base.

Interletrado. Distancia horizontal entre letra y letra. Ampliarla significaría distancias más cada letra de una palabra.

Kerning. Es la distancia entre letra y letra pero aplicando distancias distintas en función de las letras. Así en la imagen anterior se observa que la distancia entre la *M* y la letra *o*, no es la misma que entre la *o* y la *x* (la *x* se mete un poco más dentro de la *o*), ya que de esa forma el texto es más estético.

Cuerpo. La mayor parte de las tipografías actuales tiene tamaños (en horizontal) distintos de las letras. Es decir no ocupa lo mismo la letra *M* que la letra *N* (la *N* es más corta). Los llamados tipos de letra monoespaciados (como la letra *Courier*), hacen que cada carácter ocupe lo mismo (al estilo de las máquinas de escribir antiguas) en ellas la *M* ocupa lo mismo que la *N*.

Letras vectoriales y letras en mapa de bits. Hoy en día todas las tipografías son de tipo vectorial, esto significa que no se guarda una imagen de cada letra, sino que de cada letra se guarda la fórmula para dibujarla. Esto permite que al ampliar las letras nunca perdamos resolución (al contrario de lo que ocurría en los ordenadores antiguos).

Las primeras letras vectoriales fueron las pertenecientes al lenguaje **PostScript** de **Adobe** incluidos en los ordenadores **Apple MacIntosh**. **Microsoft** creó tipos de letra **TrueType** para Windows y ambas compañías actualmente desarrollan los tipos **OpenType** que se espera se convierta en un estándar abierto.

En la actualidad se da por hecho que todas las fuentes son vectoriales.

Familia. Hay dos grandes familias de letras las letras de tipo Serif y las Sans-Serif. Las primeras se distinguen por colocar *patas* en algunas letras. Es el caso de la letra **Times**, que escribiría así: MÉTODO (obsérvense las patas de la *M* y la *T*), mientras que en forma sans serif, por ejemplo como la letra Arial, se escribiría así: MÉTODO.

Propiedades CSS para modificar la fuente

font-size

Modifica el tamaño de la fuente. Se puede especificar de tres maneras:

Usando palabras especiales. Hacen referencia a tamaños predefinidos.

Son:

valor	significado
xx-small	Fuente muy pequeña
x-small	Fuente pequeña
small	Fuente un poco pequeña
medium	Fuente normal.
large	Fuente un poco grande
x-large	Fuente grande
xx-large	Fuente muy grande

Aumentando o disminuyendo el tamaño. En este caso se aumenta o disminuye el tamaño de la letra sobre el tamaño que le correspondería a la letra, por herencia. Valores:

valor significado

smaller Más pequeña

larger Más grande

Tamaño heredado. Lo consigue la palabra **inherit**. Deja la letra con el tamaño que hereda de su contenedor o elemento padre.

Tamaño inicial. Lo consigue la palabra **initial**. Coloca la letra a su tamaño por defecto (**Internet Explorer** no reconoce este valor).

Modo exacto. En este caso se indica el tamaño de la letra con su valor numérico. Inmediatamente tras este número se indica la medida en la que se debe medir el número (sea de forma absoluta o relativa). Ejemplos: *12px, 2mm, 12pt, 1.2em, 120%,...*

Ejemplo (sube la letra un 40% sobre el tamaño que le correspondería normalmente al elemento h1):

```
h1 {
    font-size: 1.4em;
}
```

font-family

Indica el tipo de letra. El problema es que no todas las fuentes están disponibles en todos los sistemas. Por ello, se suelen indicar varias opciones separadas por comas. Si la primera no está disponible, se usa la siguiente.

Ejemplo:

```
p{
    font-family: "AvantGarde Bk", Arial, Helvetica,
```



```
sans-serif;

}
```

En este código se intenta asignar la fuente **AvantGarde Bk** a todos los párrafos. Si el sistema del usuario no posee este tipo de letra, se prueba con **Arial**. Si tampoco está disponible, se prueba con **Helvetica**. Si tampoco, se prueba con una letra **sans-serif** genérica.

Se usan comillas dobles para los nombres de fuentes compuestas por varias palabras.

La norma obliga a dar como última opción de la lista de valores de **font-family** una de estas familias genéricas:

familia	uso
serif	Letra genérica utilizada en tipos de letras con <i>patitas</i> . Como por ejemplo Times New Roman.
sans-serif	Para letras genéricas sin patas como Arial O Helvética.
fantasy	Para letras más informales como Impact.
monospaced	Letras donde cada carácter ocupa lo mismo. En Windows suele ser Arial.
cursive	Para letras que simulan escritura manual como Brush Script

font-weight

Peso de la fuente (grosor). Valores posibles:

normal. Espesor normal.

bold. Negrita

Número. Se trata de indicar un número, que puede ser: **100, 200, 300, 400, 500, 600, 700, 800, 900 ó 1000**. Para que funcionen esos valores, debemos tener tipos de letra que incluyan esos grosores.

bolder. Más gruesa. Aumenta el grosor actual de la letra (al siguiente disponible).

lighter. Más delgada. Disminuye el grosor actual de la letra (al siguiente disponible).

También son válidos los valores **initial** e **inherit**.

font-style

Estilo de letra. Puede ser: **normal**, **italic** (cursiva) u **oblique** (normalmente se representa igual que la anterior). Como siempre, también son válidos los valores **initial** e **inherit**.

font-variant

Versales (small-caps). Valores: **normal** y **small-caps** (además de **initial** e **inherit**)

line-height

Permite calibrar el interlineado (la distancia entre cada línea). Se puede especificar de estas formas:

Un número. En ese caso dicta la distancia multiplicando este número por la distancia normal. Es decir si indicamos 2, el interlineado será doble, si indicamos 1.5 será un 50% mayor de lo normal.

Un número seguido de una unidad de medida. Si indicamos *16px*, entonces estamos indicando la distancia exacta entre cada línea (que será de 16 píxeles).

font

Permite desde una sola propiedad cambiar en un solo golpe todas las anteriores. Su sintaxis es:

```
font: font-style font-variant font-weight font-size/line-height  
font-family;
```

El orden tiene que ser estrictamente ese, pero algunas propiedades se pueden dejar sin utilizar.

Ejemplos:

```
/*Letra cursiva y negrita de tipo Comic con opciones a Arial y  
Helvetica*/  
  
font: italic bold 16pt "Comic Sans MS", Arial,  
Helvetica, sans-serif;  
  
/* Letra cursiva con versalitas de tamaño 18 puntos y 24 de puntos  
de distancia entre cada línea */  
  
font: italic small-caps 18pt/24pt;  
  
/* Letra cursiva */  
  
font: italic;
```

Esta propiedad también nos permite utilizar las fuentes de los elementos del sistema operativo en el que estemos, concretamente:

caption. Letra utilizada para los títulos.

icon. Letra de los iconos

menu. Letra de los menús

message-box. Letra de los cuadros de diálogo

small-caption. Letra de los controles pequeños.

status-bar. Letra de las barras de estado.

color

Color de la fuente. Utilizando cualquiera de los códigos de color explicados en el apartado dedicado a los colores.

Importar tipos de letra

@font-face

Uno de los problemas más importantes sobre la tipografía es querer utilizar un tipo de letra sobre la que no hay seguridad que exista en el dispositivo de la persona que está viendo la página HTML.

CSS3 ha introducido el uso de esta directiva en el código CSS para precargar el tipo de letra necesario. Para ello deberemos utilizar el archivo que contiene el tipo de letra en sí. El formato posible para los archivos con el tipo de letra es:

ttf. Letras de tipo **TrueType**, las clásicas de Windows. Utilizadas por Mozilla y Chrome/Safari (y otros).

otf. Letras de tipo **Open Type**, son las que se usan en los sistemas Windows actuales. Está basada en el formato anterior. Utilizadas por Mozilla y Chrome/Safari (y otros).

eot. Propiedad de Microsoft (y por lo tanto compatible con los navegadores de esta empresa)

woff. **Web Open Font Format**, Formato abierto para la web impulsado por Mozilla y compatible en casi todas las versiones de los últimos navegadores. Se creó para evitar los problemas de propiedad de los formatos anteriores.

svg. Formato estándar vectorial de gráficos, que permite también diseñar tipos de letras. Lo reconocen algunos navegadores (especialmente para el móvil).

Con tantos formatos e incompatibilidades podemos tener el problema de que la fuente elegida al final no se vea correctamente. Por ello servicios como el disponible en la página **Font Squirrel** <http://www.fontsquirrel.com/tools/webfont-generator> permite subir la letra en formato **Open Type** (suponiendo que la fuente no posea derechos reservados o bien les hayamos pagado) y nos descarga todos los archivos en los demás formatos e incluso el código que hay que escribir en nuestra hoja de estilos CSS para cargar la fuente correctamente en cualquier navegador compatible con CSS3.

Para cargar la fuente se usa la instrucción **@font-face** a la que se le incorporan varias propiedades entre las que están al menos **font-family** (para d

Por ejemplo, supongamos que queremos utilizar el tipo de letra *Cantarell* y disponemos (teniendo en cuenta que los tipos de letra pueden tener derechos exclusivos de uso, por lo que habrá que asegurarse de que uso es libre o bien pagar los derechos para su uso) de archivo *Artistik.ttf* que contiene el tipo de letra en sí. El código para que todos los párrafos la usen (con la seguridad de que sí saldría) es:

font-family. Para dar nombre de familia a la letra que estamos importando. Ese nombre es el que luego utilizarán los selectores para elegir la letra que hemos cargado (en el ejemplo es lo que usa **p** para dar formato con esa letra a los párrafos).

src. En la que se indica una o más URL (separadas por comas) que apuntan a archivos de letras que se cargarán en la página para que se vea correctamente la letra. El hecho de dar varias rutas con archivos de tipos de letra es para proporcionar opciones, para en caso de no funcionar el primer archivo cargar el segundo y así sucesivamente.

Opcionalmente podemos utilizar las propiedades:

font-weight.

font-style.

font-stretch.

Todas ellas vistas anteriormente y que ahora sirven para indicar de qué tipo es la fuente que se está cargando; si es negrita, cursiva, etc.

Ejemplo:

```
@font-face {  
    font-family: cantarellbold;  
    /* primera opción, tipo de letra EOT, no admite  
       usar opciones */  
    src: url('font/cantarell-bold-webfont.eot');  
    /* opciones en orden de prioridad deseada para  
       importar los tipos de letra */  
    src:  
        local("Cantarell Bold"),  
        url('font/cantarell-bold-webfont.eot?#iefix')  
            format('embedded-opentype'),  
        url('font/cantarell-bold-webfont.woff')  
            format('woff'),  
        url('font/cantarell-bold-webfont.ttf')  
            format('truetype'),  
        url('font/cantarell-bold-webfont.svg#cantarellbold')  
            format('svg');  
    font-weight: bold;  
    font-style: normal;  
}  
p{  
    font-family:cantarellbold;  
    ...  
}
```

En el ejemplo anterior en la propiedad **src** se indican varias rutas mediante la función **url** y en cada una de ellas, se indica el formato del archivo mediante la función **format**. Antes de usar las rutas, se buscará la fuente *Cantarell Bold* en el ordenador local gracias a la función **local** y así si el usuario ya la tiene en el sistema no hará falta cargarla desde el archivo externo.

Aunque esta técnica es muy interesante porque nos permite utilizar fuentes personales y no depender de que lo el usuario o usuaria tiene instalado en su ordenador, hay que cargar el archivo de fuentes y eso puede llevar cierto tiempo, durante el cual no se ve texto alguno o se ve con un formato diferente, lo cual puede ser problemático. No hay que abusar de esta técnica cargando demasiadas fuentes porque si no la carga de la página web sería muy lenta.

Servicios online de importación de tipos de letra

El problema de los tipos de letra poco a poco se va solucionando gracias al elemento **@font-face**, el problema es que su manejo no es nada sencillo.

Por ello han aparecido servicios de tipos de letra online, que nos permiten utilizar tipos de letra de librerías (algunas gratuitas y otras de pago) disponibles en Internet. Para ello se suele requerir añadir en las páginas un enlace a un archivo CSS o JavaScript, que en realidad se encarga por nosotros de importar adecuadamente el tipo de letra.

Importar muchos tipos de letra puede retrasar muchísimo la carga de la página web por lo que se debe utilizar con cautela. Las instrucciones para importar estos tipos de letra les proporcionan los propios servicios online. Los más conocidos son:

Google Fonts. Disponible en <https://www.google.com/fonts> y perteneciente a Google, es claramente el más utilizado por incorporar un buen número de tipos de letra gratuitas.

Adobe Typekit. El más utilizado a nivel profesional. Hay posibilidad de suscripción gratuita a un número limitado de fuentes, para utilizar muchas más hay que pagar una cuota (o disponer de licencia **Creative Cloud** de **Adobe**). Disponible en <https://typekit.com/>

Adobe Edge Fonts. Servicio gratuito de Adobe (realmente servido desde **Typekit**) que permite utilizar en las páginas web en torno a 500 tipos de letra. <https://edgewebfonts.adobe.com/fonts>

fontdeck. Se trata de la web <http://fontdeck.com/> que dispone un catálogo de tipos de letra de calidad. Se cobra por el uso de las letras, si usamos pocas se paga poco si usamos muchas, se paga más. Sólo se cobra además si hay páginas web que creemos que sigan utilizando esas fuentes.

webtype. En <http://www.webtype.com>, servicio parecido al anterior.

fonts.com. Posee el catálogo más amplio de tipos de letra. Tiene un plan gratuito con acceso a 200 fuentes (el de pago accede a más de 20.000) limitado a dos proyectos y 20.000 páginas vistas en ellos al mes.

Formato del texto en CSS

Son propiedades que afectan al texto, fundamentalmente cambian el formato de los párrafos y aspectos del texto que no se refieren a su tipografía.

word-spacing

Indica la distancia entre las palabras del texto. Usa las mismas medidas que la propiedad **font-size**.

letter-spacing

Indica la distancia entre las letras del texto. Es similar a la anterior, pero ahora referida a la distancia horizontal entre caracteres. Ejemplo con letter-spacing:20px;

text-decoration

Se indican posibles efectos en el texto. Valores:

underline. Subrayado (línea por debajo del texto)

overline. Línea por encima del texto.

line-through. Tachado, línea que atraviesa el texto.

blink. Parpadeo. No funciona en casi ningún navegador (sí en Firefox).

text-decoration-color

Permite especificar el color del subrayado cuando se utilizan líneas mediante **text-decoration**. Solo funciona, por ahora, en **Mozilla Firefox** (y requiere utilizar el prefijo **-moz-**).

text-decoration-style

Permite especificar el estilo del subrayado. Solo funciona, por ahora, en **Mozilla Firefox** (y requiere utilizar el prefijo **-moz-**). Valores: **solid** (sólida), **double** (línea doble), **dotted** (línea punteada), **dashed** (línea rayada) y **wavy** (línea ondulada).

vertical-align

Posición vertical del texto (o imagen) respecto a su contenedor. Es muy versátil porque permite tanto alinear en vertical un texto respecto, por ejemplo, a la celda de la tabla en la que se encuentre; como indicar superíndices y subíndices. Posibilidades:

baseline. En la línea base inferior del texto

sub. Subíndice

super. Superíndice

top. Arriba respecto al elemento más alto de la línea

text-top. En la línea superior del texto

middle. Medio respecto a la altura del texto o contenedor en el que estemos

bottom. Abajo respecto al elemento más alto de la línea

text-bottom. En la línea inferior del texto

Porcentaje. Porcentaje respecto al texto (por ejemplo 120%)

Lo malo es que sólo funciona en texto dentro de celdas de tablas. Si deseamos alinear el texto dentro de una etiqueta div por ejemplo, habría que utilizar la propiedad **display** con el valor **table-cell**.

text-align

Alineación horizontal del texto. Puede ser: **left** (izquierda), **right** (derecha), **center** (centrada) o **justify** (justificada a derecha e izquierda).

text-indent

Sangría de la primera línea del párrafo. Distancia extra que se deja a la primera línea respecto al resto de líneas. Por ejemplo si indicamos, **text-indent:50px**; el resultado sería (para un párrafo al que se aplique ese código):

text-transform

Permite modificar el texto para que se muestre en mayúsculas o minúsculas.

capitalize. La primera letra en Mayúsculas

uppercase. Mayúsculas

lowercase. Minúsculas

none. No hace ninguna transformación

direction

Procede de CSS2, especifica la dirección en la que se escribe el texto. Posibilidades:

ltr. *Left to right*, de izquierda a derecha

rtl. *Right to left*, de derecha a izquierda (utilizada en lenguas como el árabe)

text-overflow

Parte de CSS3. Indica que hacer con el texto cuando está dentro de un contenedor (como una capa) y no tiene el tamaño suficiente para mostrar todo el texto. Posibilidades:

clip. El texto sale recortado. Sólo se ve el texto que cabe en la capa, el resto no se muestra

ellipsis. Como la anterior, pero se ponen puntos suspensivos al final del texto recortado.

text-shadow

Se trata de una propiedad CSS3. Permite colocar una sombra al texto para darle efecto de volumen. Tiene esta sintaxis:

```
text-shadow: color distanciaX distanciaY desenfoque;
```

color. Es el color de la sombra

distanciaX. Es el desplazamiento horizontal que tendrá la sombra (puede ser positivo o negativo)

distanciaY. Es el desplazamiento vertical que tendrá la sombra (puede ser positivo o negativo)

desenfoque. Es opcional e indica cuánto se va a desenfocar la sombra. Se indica una cantidad que cuanto mayor sea, más desenfocará el fondo.

word-wrap

Se trata de una nueva propiedad de CSS3 que permite partir el texto cuando tenemos palabras muy largas.

word-wrap con valor **break-word**, parte el texto en aquellas palabras que, de otro modo, se salen fuera del margen.

Cajas

Formato de fondo

Se denomina fondo al espacio que queda “por debajo” del texto e imágenes de un elemento. Normalmente el fondo es transparente, pero podemos colorearlo o usar imágenes como fondo.

Color de fondo

La propiedad **background-color** permite establecer un color de fondo al elemento al que se aplique la propiedad. Si se aplica al elemento **body**, toda la página tendrá ese color de fondo; aplicado a otro elemento (como **div** o **p** por ejemplo), dicho elemento tendrá como fondo el color aplicado.

Imagen de fondo. background-image

La propiedad **background-image** permite establecer una imagen de fondo. Esta imagen se superpone al color de fondo (si lo hay, de modo que si la imagen no se puede cargar (porque la ruta a ella no se ha indicado), entonces aparece el color de fondo.

Por defecto, la imagen de fondo se repite las veces necesarias (creando un mosaico), tanto en horizontal como en vertical, hasta que se rellena el elemento.

Repetición del fondo. background-repeat

En principio, el fondo se repite en todas las direcciones hasta rellenar el elemento al que se aplica la imagen de fondo. Pero podemos hacer que la imagen se repita en una sola dirección (horizontal o vertical), e incluso hacer que no se repita con la propiedad **background-repeat**

Posibles valores de la propiedad:

repeat. Es el valor por defecto. la imagen se repite en todas las direcciones (efecto mosaico), hasta rellenar completamente el elemento.

repeat-x. La repetición de la imagen se hace solo en horizontal.

repeat-y. La repetición de la imagen se hace solo en vertical.

no-repeat. La imagen no se repite aparecerá solo una vez

Posición de la imagen. background-position

Por defecto la imagen se coloca desde la esquina superior izquierda de la página (posición 0,0) y desde ahí se repite (si se ha indicado repetición de la imagen con la propiedad anterior) o no.

La posición desde la que la imagen se coloca inicialmente se puede modificar. La forma de hacerlo es así:

background-position: posicionHorizontal posicionVertical

Podemos indicar ambos valores de esta forma:

posicionHorizontal. Se puede especificar las palabras: **left** (izquierda), **right** (derecha) o **middle** (centro). También se puede indicar una coordenada horizontal concreta (por ejemplo 12px) o relativa (5%).

posicionVertical. Se puede especificar las palabras: **top** (arriba), **bottom** (abajo) o **center** (centro). También se puede indicar una coordenada vertical concreta (por ejemplo 20px) o relativa (15%).

Así por ejemplo el código:

background-position: right bottom;

La imagen de fondo (si no hay repetición de la misma) se mostrará en la esquina inferior derecha del elemento en el que se ponga.

Desplazamiento del fondo. background-attachment

Cuando se desplaza el contenido de una página web, el fondo se mueve con el resto de la página. Mediante esta propiedad podremos hacer que el fondo quede fijo mientras que solo el resto de elementos se mueven. Los posibles valores para la propiedad son:

scroll. Valor por defecto. El fondo y el texto se mueven juntos cuando el usuario se desplaza por la página

fixed. El fondo es fijo y solo se mueve el texto.

Tamaño del fondo. background-size

Permite indicar el tamaño de la imagen de fondo indicando su tamaño en horizontal y en vertical. Es una propiedad de CSS3 que solo funciona en los navegadores más modernos.

Ejemplo (Hace que la imagen de fondo mida 320px por 240px):

```
background-size: 320px 240px;
```

Otra posibilidad es:

```
background-size: 50% auto;
```

La palabra auto hace que la dimensión en la que aparece esa palabra (en el ejemplo se refiere a la altura) se calcule automáticamente.

Más recientemente (y por lo tanto no reconocido por navegadores que no sean bastante modernos) a esta propiedad le podemos dar los valores:

cover. Hace que la imagen llene el contenedor en el que se encuentra de forma *responsive*, es decir, sin perder su proporción de altura y altura. Hace que la imagen sea lo más grande posible sin perder la proporción de altura por altura.

contain. Escala la imagen haciendo que cubra el 100% de la anchura o de la altura dependiendo del tamaño de la ventana de modo que quedarán huecos en el contenedor de la imagen ya que la otra proporción no se escala.

Área de dibujo del fondo. background-clip

Es una propiedad de CSS3 que define desde qué zona se dibuja el fondo. Sus posibilidades son:

border-box. Opción por defecto, el fondo se dibuja desde donde normalmente está el borde de elemento al que le estamos dibujando un fondo.

padding-box. El fondo se dibuja en el área de relleno (**padding**) que es una zona interior al borde.

content-box. El fondo se dibuja en la zona de contenido que es en la que se coloca el texto en el elemento.

Propiedad background

Fija en una sola propiedad todas las propiedades de fondo. Sintaxis:

```
background: background-color background-image  
            background-repeat background-attachment  
            background-position
```

Ejemplo:

```
background: maroon url('fondo1.gif') no-repeat fixed left bottom;
```

Coloca color marrón de fondo, la imagen *fondo1.gif*, que solo aparece una vez, en la parte inferior izquierda y que se queda fijo en el fondo cuando el usuario se desplaza por la pantalla.

Poner varias imágenes de fondo

CSS3 incorporó una importante mejora a las imágenes de fondo y es el hecho de poder colocar varias imágenes de fondo en un elemento. Para ello basta poner comas en las imágenes en sí. Ejemplo:

```
<style>
```

```
    body{  
        background-image:url(luna.png),  
                           url(saturno.png),url(lactea.jpg);  
        background-position: left top, right top, 0 0;  
        background-repeat: no-repeat,no-repeat,repeat;  
    }
```

```
</style>
```

Como se ha visto, aplicar varias imágenes de fondo se hace simplemente separando cada imagen por comas en cada propiedad.

Podemos usar directamente la propiedad **background** para producir el mismo resultado:

Una vez más, lo malo es que no todos los navegadores son capaces de aplicar esta posibilidad, por lo que muchos diseñadores optan por construir capas y en cada de una ellas manejar el fondo, para hacer este tipo de efectos.

Elementos del formato de caja

Se suele denominar **formato de caja** (o **formato de cuadro**) a la parte de CSS encargada del formato referente al rectángulo imaginario que envuelve a un elemento de una página HTML. Este rectángulo imaginario contiene completamente al elemento, así como al borde (si lo hay) y al espacio que se deja fuera del margen (**margin**) y el espacio que se deja por dentro del borde (**padding**).



En la ilustración se describen los elementos del formato de cuadro en CSS.

El espacio que va desde el contenido del elemento (en la ilustración sería el texto de color negro) hasta el borde es lo que se denomina espacio de relleno o *padding*. Ese espacio queda coloreado o rellenado con el fondo que se haya configurado.

El margen es un espacio exterior al borde. En dicho espacio no se ve el borde, es un espacio transparente veremos en él, el fondo configurado en el contenedor del elemento. Es decir si el elemento es una etiqueta **p** que cuelga directamente de **body**, en el espacio de margen veremos el fondo del elemento **body**.

Configuración del borde

Propiedades para el borde

El borde posee tres propiedades fundamentales: color, estilo y grosor. Además podremos configurar independientemente cada borde (izquierdo, derecho, superior e inferior). Por lo que disponemos de todas posibilidades:

border-width. Anchura del borde, indicada en la unidad deseada. Por ejemplo *12px* indicaría un borde de doce píxeles. También podemos indicar estos valores:

thin. Borde fino

medium. Borde de grosor medio

thick. Borde grueso

Podemos indicar a la vez diferentes grosores. Para ello podemos especificar cuatro valores a la vez en lugar de uno. El orden para indicarlos es arriba, derecha, abajo e izquierda (siguiendo las agujas del reloj desde las doce). Ejemplo:

```
border-width: thick thin thick 0px;
```

Ese código aplica bordes superior e inferior gruesos, a la derecha fino y a la izquierda no dibujará ningún borde. No es necesario indicar todos los bordes. Por ejemplo:

```
border-width: thick thin;
```

```
/* superior e inferior grueso, izquierdo y derecho fino */
```

```
border-width: thick thin 0px;
```

```
/* superior grueso, izquierdo y derecho fino e inferior sin borde */
```

border-style. Modifica el estilo del borde. Esta propiedad afecta a todos los bordes (superior, derecho, inferior e izquierdo). Posibilidades:

solid. Borde sólido

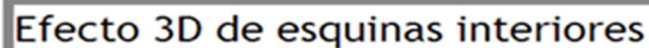
dashed. Borde rallado

dotted. Punteado

double. Doble

groove. Borde 3D con efecto de hundimiento

ridge. Borde 3D con efecto de elevación



Efecto 3D de esquinas interiores

inset. El efecto depende del color elegido



Efecto 3D de esquinas interiores

outset. El efecto depende del color elegido

none

Como ocurre con **border-width**, podemos indicar varios estilos a la vez. Ejemplos:

```
border-style: solid dashed dotted dotted;
```

```
/* superior sólido, derecho rallado, inferior e izquierdo punteado*/
```

```
border-style: solid dashed dotted;
```

```
/* superior sólido, izquierdo y derecho rallado e inferior punteado
*/
```

border-color. Color del borde. Al igual que en las dos propiedades anteriores podemos asignar colores distintos a cada borde (siguiendo lo ya comentado).

border-top-width, border-top-color, border-top-style. Permite indicar respectivamente la anchura, el color y el estilo del borde superior.

border-left-width, border-left-color, border-left-style. Permite indicar respectivamente la anchura, el color y el estilo del borde izquierdo.

border-bottom-width, border-bottom-color, border-bottom-style. Permite indicar respectivamente la anchura, el color y el estilo del borde inferior.

border-right-width, border-right-color, border-right-style. Permite indicar respectivamente la anchura, el color y el estilo del borde derecho.

border-top. Permite indicar a la vez las tres propiedades para el borde izquierdo. El orden es: grosor, estilo y color. Ejemplo:

```
border-top: 2px solid red;
```

No estamos obligados a rellenar las tres propiedades podemos rellenar solo una o dos.

border-right. Permite indicar, a la vez, las tres propiedades del borde derecho.

border-bottom. Permite indicar, a la vez, las tres propiedades del borde inferior

border-left. Permite indicar, a la vez, las tres propiedades del borde izquierdo

border. Permite indicar el grosor, estilo y color de los cuatro bordes a la vez. Se usa igual que las anteriores pero en este caso lo que configuremos implica a todos los bordes.

Bordes redondeados

En CSS3 ha aparecido la posibilidad de crear bordes redondeados así las esquinas de los bordes pueden formar un radio de un círculo cuyo tamaño podremos especificar. Es posible incluso indicar diferentes redondeados para las distintas esquinas.

La pega es que es un formato CSS3 por lo que algunos navegadores no lo reconocen. Propiedades relacionadas:

border-radius. Permite indicar bordes redondeados (también afecta al sombreado que saldrá con los bordes redondeados. Ejemplo de uso:

```
p{  
  
    background-color: lightgray;  
  
    border: thick double black;  
  
    border-radius: 10px;  
  
}
```

Permite indicar a la vez varios tamaños de redondeo en el orden de izquierda a derecha y de arriba abajo.

border-bottom-left-radius. Permite indicar un tamaño de radio para redondear el borde inferior izquierdo.

border-bottom-right-radius. Permite indicar un tamaño de radio para redondear el borde inferior derecho.

border-top-left-radius. Permite indicar un tamaño de radio para redondear el borde superior izquierdo.

border-top-right-radius. Permite indicar un tamaño de radio para redondear el borde superior derecho.

El borde admite dos radios, uno en vertical y otro en horizontal. Ejemplo:

```
p{  
    background-color: red;  
    border-radius: 80px / 20px;  
}
```

El borde redondeado queda muy extraño ya que se ha puesto un radio muy grande en horizontal y otro mucho más pequeño para la vertical.

Bordes usando imágenes (CSS3)

También en CSS3 ha aparecido la posibilidad de indicar una imagen para que cubra el borde. La idea es crear una imagen (el formato idóneo es PNG) en la que se pinten los bordes. Se estirará la imagen en la zona que no son las esquinas (el tamaño de las esquinas se puede indicar) para cubrir todo el tamaño del elemento que se está bordeando con la imagen.

Normalmente la imagen tiene solamente coloreado los bordes y el resto de la imagen son píxeles que se considerarán transparentes (por eso el formato más habitual para utilizar con este tipo de imagen es el PNG).

Un ejemplo de imagen preparada para pintarse como borde es esta:



Todo lo que aparece de blanco en la imagen, en realidad tiene color transparente y así a través de ellos se observará el fondo.

La idea es indicar la parte de la imagen dedicada a las esquinas usando su tamaño (en la imagen cada cuadrado de las esquinas mide 18 píxeles). De esa forma la imagen quedará troceada en nueve partes (las cuatro esquinas y la parte superior, inferior, izquierda, derecha y central de la imagen).

Para la zona que no es la esquina, indicaremos como haremos para rellenar completamente el elemento. Es decir, en la imagen anterior, los círculos azules no valen para rellenar el elemento completo al que estemos dando borde con la imagen. Por ello indicaremos al navegador que repita cada círculo (**round**), o bien que le estire (**stretch**).

Sintaxis de la propiedad **border-image**:

```
border-image: origenDeLaImagen tamañoEsquinas modoEstirar;
```

origenDeLaImagen. Es la ruta a la imagen

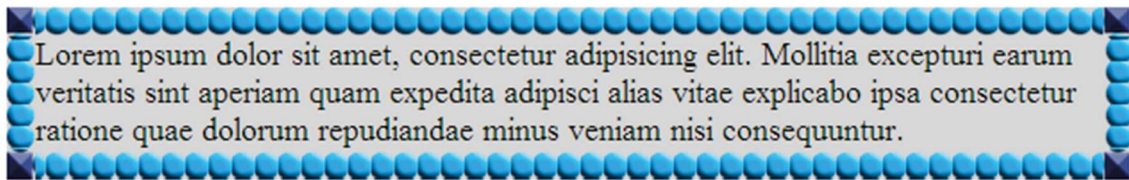
tamañoEsquinas. Es lo que mide cada esquina empezando por la superior izquierda y luego yendo hacia la derecha y hacia abajo. Si miden lo mismo, solo se indica un número (normalmente no se indica la unidad porque siempre se entiende en píxeles)

modoEstirar. Puede ser **repeat** (repetir hasta llenar), **round** (igual que al anterior pero consiguiendo que los dos extremos no queden partidos) o **stretch** (estirar la imagen). Podemos indicar dos valores, el primero se referirá a la forma de rellenar en horizontal y el segundo será en vertical.

Ejemplos (suponiendo que la imagen anterior es *borde2.png* y que los cuadraditos de la imagen miden *18 píxeles x 19 píxeles*):

```
p{
    background-color: lightgray;
    border:15px solid transparent;
    -webkit-border-image: url(borde2.png) 18 19 round;
    /*para Safari*/
    -o-border-image: url(borde2.png) 18 19 round;
    /* para Opera*/
    border-image: url(borde2.png) 18 19 round;
}
```

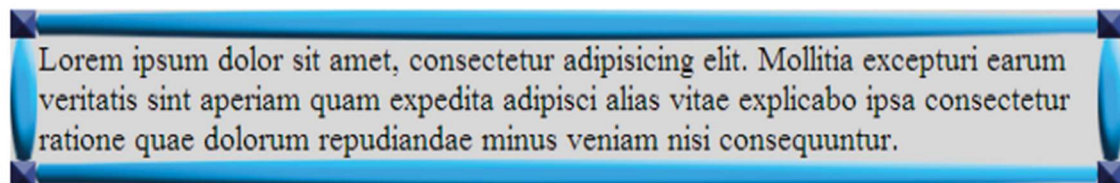
La propiedad **border** debe indicar como tipo de borde el valor **transparent**. con el fin de usar imágenes Resultado:



Otro:

```
p{
    background-color:lightgray;
    border-width:15px;
    -webkit-border-image: url(borde2.png) 18 19 stretch;
    /*para Safari*/
    -o-border-image: url(borde2.png) 18 19 stretch;
    /* para Opera*/
    border-image: url(borde2.png) 18 19 stretch;
}
```

Resultado:



Sombreado de la caja. `box-shadow`

Se puede aplicar una sombra a un elemento mediante la propiedad `box-shadow`. La sintaxis es:

```
box-shadow:distanciaH distanciaV desenfocado propagación color inset;
```

Donde:

distanciaH. Es la distancia horizontal del borde del elemento al borde de la sombra aplicada.

distanciaV. Es la distancia vertical del borde del elemento al borde de la sombra aplicada.

desenfocado. Es la cantidad de desenfocado de la imagen. Si no se indica, no se aplica desenfoque alguno a la sombra.

propagación. Fuerza de la sombra, a mayor fuerza, más contraste tendrá y sus bordes invadirán más al resto de elementos. Por defecto vale 0.

color. Color que aplicamos a la sombra (por defecto es el negro)

inset. Si aparece hace que la sombra se muestre hacia el interior de la caja, haciendo un efecto de “hundimiento”. De otro modo, la sombra se dibuja hacia el exterior, haciendo un efecto de “elevación”

Ejemplo:

```
p{  
  
    background-color:yellow;  
  
    box-shadow: 20px 20px 15px black;  
  
    -webkit-box-shadow:20px 20px 15px black;  
  
    /* para Chrome anterior a la versión 10 */  
  
    -moz-box-shadow:20px 20px 15px black;  
  
    /* Para mozilla anterior a la versión 15*/  
  
}
```

Esta propiedad ha sido recogida desde hace poco tiempo en todos los navegadores, Internet Explorer a partir de la versión 9 y Chrome y Firefox desde las versiones 10 y 15 (antes permitían su uso mediante prefijo -webkit- y -moz- respectivamente)

Perfiles

Los perfiles son bordes que se ponen en el exterior de los elementos de tipo caja.

A diferencia de los bordes, propiamente dichos, explicados en los apartados anteriores, los perfiles son líneas que no comen espacio ni al relleno ni al margen. Los perfiles son bordes que se colocan en la posición en la que normalmente ocupa un borde pero no ocupan un espacio extra, como si la línea que dibujan estuviera por *debajo* del texto. Las propiedades de los perfiles son:

outline-color. Color del perfil.

outline-width. Anchura del perfil

outline-style. Estilo del perfil (admite las mismas posibilidades que border-style).

outline. Resume las tres anteriores en una sola propiedad, admite poner primero el color, luego el estilo y luego el ancho.

El estándar CSS añadió estas propiedades no para sustituir a los bordes, sino para ser aplicados a elementos de formulario con el fin de que quedará marcado el elemento que actualmente posee el foco; es decir, aquel elemento que recibe los datos procedentes del dispositivo de entrada del usuario, normalmente el teclado. Se usan también para depuración y para dar formato CSS a eventos de tipo *hover*, ya que el borde se añade sin que se muevan los elementos a su alrededor, lo que provoca un efecto más elegante.

Listas

Propiedades CSS para listas

list-style-type

La propiedad *list-style-type* permite especificar el tipo de elemento de numeración de la lista. Normalmente los navegadores muestran un círculo relleno en las listas no numeradas (las que se realizan mediante la etiqueta **ul**) y números arábigos en las listas numeradas (etiqueta **ol**).

Esta propiedad permite cambiar esos símbolos para elegir el que deseemos. La cuestión es que (aunque se podría) no es recomendable hacer que la etiqueta **ol** muestre símbolos no numéricos y tampoco que la etiqueta **ul** muestre símbolos numéricos, para mantener la coherencia semántica en el lenguaje HTML.

En todo caso los posibles valores de esta propiedad son:

armenian. La lista quedará encabezada por números del alfabeto armenio.

circle. La lista quedará encabezada por círculos.

cjk-ideographic. Se usan símbolos numéricos ideográficos chinos.

decimal. Números decimales.

decimal-leading-zero. Números decimales empezando por cero.

disc. La lista quedará encabezada por círculos sin rellenar.

georgian. Números del alfabeto georgiano.

hebrew. Números del alfabeto hebreo.

hiragana. Números del alfabeto Hiragana japonés en uso actual (*gojūon*).

hiragana-iroha. Números del alfabeto japonés Hiragana clásico (*iroha*)

katakana. Números del alfabeto Katakana japonés en uso (*gojūon*).

katakana-iroha. Números del alfabeto Katakana japonés clásico (*gojūon*).

lower-alpha. Letras minúsculas del alfabeto actual.

lower-greek. Letras griegas minúsculas

lower-latin. Letras minúsculas latinas.

lower-roman. Números romanos en minúsculas.

none. Sin números.

square. Cuadrados rellenos.

upper-alpha. Letras mayúsculas del alfabeto actual.

upper-latin. Letras mayúsculas latinas.

upper-roman. Números romanos en mayúsculas.

list-style-image

En lugar de utilizar uno de los símbolos anteriores, se puede indicar una imagen con la que se rellenará la lista. La imagen puede tener cualquiera de los formatos habituales en las páginas web (*gif*, *jpg*, *png*). Lógicamente el tamaño debe de ser apropiado; si es muy grande la lista quedará totalmente descuadrada.

Ejemplo de uso:

```
list-style-image:url('cuadrado.gif');
```

list-style-position

Solo tiene dos posibles valores referidos a la posición del texto respecto de la imagen.

inside. El símbolo de numeración es interior a los márgenes del elemento en el que se coloca. Es la opción por defecto.

outside. El símbolo de numeración es exterior.

list-style

La propiedad **list-style** permite en un solo golpe configurar las propiedades anteriores. Su sintaxis es:

```
list-style: list-Style-Type, list-style-position,  
list-style-image;
```

Posicionamiento mediante CSS

Una de las grandes e históricas peticiones de los diseñadores web ha sido la posibilidad de poder colocar contenido en una posición concreta de la página, haciendo que los contenidos no se vayan mostrando en el orden en el que aparecen en el código, sino en el que decidamos libremente.

Fue la principal aportación de CSS2 y ha provocado que, en lugar de usar tablas para definir la maquetación de la página, ahora sean las capas las encargadas de hacerlo. Las **capas** son elementos que permiten una distribución de contenidos más libre. La etiqueta **div** suele ser la encargada de crear capas, aunque con HTML5 son otros elementos los que también se usan para esta labor como **section** o **article**, ya que **div** no es una etiqueta semántica (no dice qué tipo de contenido posee, si es un artículo, un título, un menú,...).

Hay que tener en cuenta que la posición de un elemento también estará influido por su valor de la propiedad `display`, como se explicó en el capítulo anterior.

Flotación

La propiedad `float` es muy útil para contenidos multimedia como imágenes o tablas. Permite indicar cómo se comporta el contenido de los elementos adyacentes sobre el actual.

Normalmente todos los elementos *no flotan*. Es decir, cuando colocamos, por ejemplo, una imagen, el contenido que va a continuación aparece en su línea base, dejando un hueco poco estético.

Podemos hacer que ese contenido contornee la imagen a su derecha (`float:left`) o a su izquierda (`float:right`). En definitiva, los posibles valores de la propiedad `float` son:

`left`. El elemento *flota* a la izquierda. El contenido inmediatamente siguiente le contornea a su derecha, siempre y cuando quepa

`right`. El elemento *flota* a la derecha. El contenido inmediatamente siguiente le contornea a su izquierda.

`none`. El elemento no flota (valor por defecto). En el caso del texto, la primera línea iría seguida al elemento, el resto caería por debajo.

Hay que tener en cuenta que esta propiedad está pensada para elementos de bloque, que son aquellos que tengan un `display` con valor `block` o `inline-block`.



Lorem ipsum dolor sit amet,
consectetur adipisicing elit. Ad, aliquam, aliquid aspernatur ducimus error harum
illo in libero modi mollitia nostrum odit omnis rem, reprehenderit sed soluta
tempore tenetur voluptatum.

El texto que sigue a la imagen pega solo la primera línea. Este es el comportamiento normal de flotación de elementos de tipo `inline-block` (al que pertenece la imagen).

En el caso de que la imagen estuviera dentro de un contenedor de bloque (un elemento `div` por ejemplo), ni la primera línea aparecería seguida.

Sin embargo el mismo código HTML usando `float:left`, queda



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ad, aliquam, aliquid aspernatur ducimus error harum illo in libero modi mollitia nostrum odit omnis rem, reprehenderit sed soluta tempore tenetur voluptatum.

En el caso de flotación derecha ([float-right](#)):

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ad, aliquam, aliquid aspernatur ducimus error harum illo in libero modi mollitia nostrum odit omnis rem, reprehenderit sed soluta tempore tenetur voluptatum.



Propiedad shape-outside

Muchos diseñadores se han quejado durante años de la imposibilidad de que el texto contornee a las imágenes al estilo de los programas de autoedición (como por ejemplo [InDesign](#) o [QuarkXPress](#)).

El caso es que CSS aporta una nueva propiedad llamada *shape-outside* que, desgraciadamente, no la reconoce del todo casi ningún navegador, aunque sí las últimas versiones del popular navegador Google Chrome.

Esta propiedad permite diseñar una forma a través del cual el contenido adyacente se contornea. Por supuesto, el elemento debe de poseer valores *left* o *right* para la propiedad *float* para que *shape-outside* haga su labor.

Las posibles valores de esta propiedad son:

[none](#). La forma de la imagen flotada será rectangular.

[margin-box](#). El contenido se contornea a través del margen de la imagen. Es la distancia habitual

[content-box](#). El contenido se acerca al contenido del elemento, más allá del margen y del relleno.

[border-box](#). El contenido se contornea alrededor del borde.

[padding-box](#). El contenido se contornea alrededor del relleno del elemento.

[circle\(\)](#). Función que permite indicar un círculo a través del cual el texto se contornea.

[ellipse\(\)](#). En este caso se usa una forma elíptica.

[url\(ruta\)](#). Se usa la ruta a una imagen a cuya forma (suponiendo que sea PNG o GIF con píxeles transparentes) se adaptará el contenido.

[polygon\(\)](#). Permite indicar coordenadas para establecer un polígono alrededor del cual se contorneará el texto. Existe un plugin para el navegador muy útil llamado [Shape Editor](#) que

permite indicar fácilmente estas coordenadas de forma visual. Simplemente se instala el plugin y se siguen las instrucciones.

Tamaño del elemento

Dos propiedades permiten establecer el tamaño de un elemento:

width. Anchura del elemento. Su valor por defecto es **auto**, lo que implica que el elemento se hará lo suficientemente ancho para que quepa su contenido. De otro modo indicaremos su tamaño utilizando la unidad de medida deseada (**px**, **em**, **%**, etc.).

En el caso de las unidades relativas, se toma como referencia lo que mide el contenedor; si indicamos un 50% a un elemento que está dentro de un elemento **div**, tomará la mitad de la anchura de ese **div**, si nuestro elemento cuelga directamente de **body**, tomará la mitad de la anchura de la página.

height. Altura del elemento. Tiene las mismas posibilidades que el elemento anterior.

max-width. Se utiliza cuando la medida de la anchura de un elemento se indica de forma relativa, por ejemplo con porcentajes. Esta propiedad indica un tamaño máximo para el elemento.

Es decir, si hemos indicado **width:50%**; y además **max-width:750px**, entonces el elemento medirá de ancho la mitad de su contenedor. Si por ejemplo el contenedor mide 2000 píxeles, nuestro elemento medirá 1000 píxeles aplicando el 50%; pero como hemos indicado la propiedad **max-width:750px**, medirá exactamente 750 píxeles.

Esta propiedad es muy interesante para aplicar diseños de tipo adaptable.

min-width. La idea es la misma que en la propiedad anterior, pero ahora se indica un tamaño mínimo que tendrá prioridad sobre el tamaño relativo de la propiedad **width**.

max-height. La misma idea aplicada a la altura del elemento. Indica la altura máxima que podrá tomar el elemento.

min-height. Altura mínima del elemento.

Posicionamiento

La propiedad **position** es la fundamental para establecer el posicionamiento. Sirve para indicar de qué forma un elemento fluye por la página. Gracias esta propiedad podremos indicar si el elemento al que se aplica realmente se comportará como una capa de contenidos que se colocará libremente en la página. Se detallan los posibles valores que puede tomar **position**.

Posicionamiento estático

Se consigue usando el valor **static** en la propiedad **position**.

El posicionamiento estático es el que se aplica por defecto. Si se establece así, el elemento al que se aplique aparece en la página en la posición que dicte el flujo natural (de izquierda a derecha y de arriba abajo) del código. Es decir, ignorará las propiedades de posicionamiento (**left**, **top**, **right** y **bottom**).

Posicionamiento relativo

Se logra con el valor **relative**. Funciona como **static** solo que, en este caso, sí permite modificar su posición inicial usando las propiedades **left**, **top**, **right** y **bottom**.

En modo **relative**, el elemento marcado con este posicionamiento sí se mueve con las propiedades de posicionamiento, pero deja el hueco que le correspondería en modo **static**. En definitiva, **relative** provoca un desplazamiento en el elemento.

Ejemplo de posicionamiento relative:

```
<!doctype html>
```

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

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>Estático</title>
```

```
<style>
```

```
div{
```

```
    width:400px;
```

```
    border:2px solid gray;
```

```
}
```

```
#capa1{
```

```
    position:relative;
```

```
    left:80px;
```

```
    top:40px;
```

```
    background-color: lightgrey;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div id="capa1">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium  
beatae blanditiis consequatur debitis
```

```
    deleniti ducimus ea in maiores, molestias mollitia, nisi, quaerat. Consequatur eveniet  
    fugit impedit incidunt odit
```

```
    quas ullam?
```

```
</div>
```

```
<div>Ab amet deserunt eaque, eveniet hic illo itaque magni reprehenderit soluta  
temporibus ut vel. Architecto at, facere
```

id iste laboriosam praesentium provident sint. Aliquam asperiores et molestias, optio sit ullam.

</div>

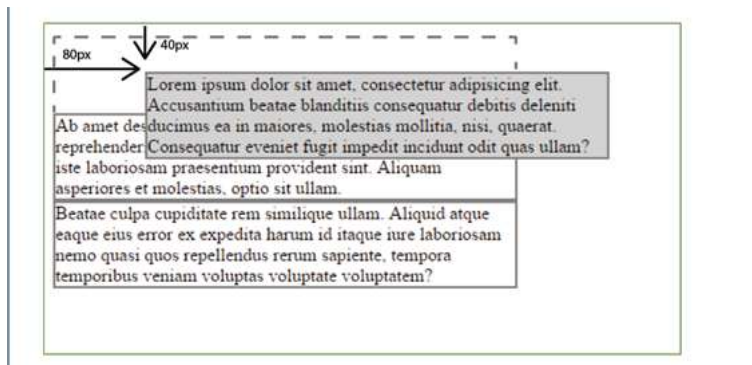
<div>Beatae culpa cupiditate rem similique ullam. Aliquid atque eaque eius error ex expedita harum id itaque iure

laboriosam nemo quasi quos repellendus rerum sapiente, tempora temporibus veniam voluptas voluptate voluptatem?

</div>

</body></html>

El resultado es el que se observa en la ilustración, el primer párrafo se desplaza pero deja el hueco que ocuparía la capa en el posicionamiento habitual, *static*.

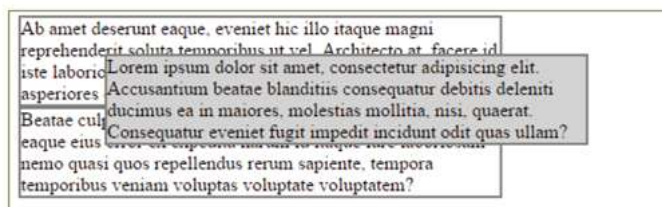


Posicionamiento fijo

El valor para este posicionamiento es *fixed*. Con este posicionamiento, el elemento flota libremente por la ventana y abandona el hueco que, por su orden natural, le correspondería. De hecho, ese hueco se elimina y le ocupa el siguiente elemento disponible. La posición del elemento se fija en base a las coordenadas de la ventana, independientemente de dónde esté situado o contenido el elemento. Es decir toma como referencia la ventana del documento de modo que la esquina superior izquierda será la coordenada 0,0.

La posición de los elementos *fixed*, además, no varía cuando el usuario desplace (*scroll*) la ventana. La posición que ocupan los elementos *fixed* no se modifica con el scroll.

Usando el mismo código del apartado anterior, si el posicionamiento fuera *fixed*, ocurre lo siguiente:



Aunque la posición de la capa es la misma que en el caso del posicionamiento relativo, se observa, que el hueco que ocupaba la misma, esta totalmente ocupada por la capa siguiente. Con el posicionamiento fijo, la capa realmente flota por encima del documento.

Posicionamiento absoluto

El valor de la propiedad `position` relacionado con este posicionamiento es `absolute`.

Como ocurría con el valor `fixed`, permite cambiar la posición del elemento a voluntad, pero toma como referencia la de su contenedor (en lugar de la ventana); siempre que el contenedor no tenga posicionamiento `static`. Si el contenedor es `static`, se usa al contenedor de dicho contenedor (si no es `static`) y así sucesivamente hasta encontrar un ancestro que no sea `static` o bien llegar al elemento `body`, en cuyo caso se tomará (como ocurre con `fixed`) a la ventana como origen de coordenadas.

Es decir, si tenemos un elemento (por ejemplo de tipo `div`) identificado como `interior` dentro de otro `div` llamado `padre`, si utilizamos la **propiedad `position: absolute`**; dentro del CSS de la capa interior, entonces las coordenadas de la capa `interior` tomarán los límites de la capa `padre` como referencia. Poniendo en la capa interior: `left: 30px`; separamos esta capa treinta píxeles del margen izquierdo de la capa padre (en lugar de separarnos del borde de la ventana como hace `fixed`).

Ahora bien si el padre es `static`, entonces tendríamos que tomar las coordenadas del padre de ese padre (`abuelo`) suponiendo que no sea `static`.

Hay otra importante diferencia. Con `fixed`, los elementos no varían la posición dentro de la ventana, pero con `absolute` sí. Por lo que los elementos de tipo `absolute` sí son desplazables.

Establecer coordenadas de posicionamiento

Cuatro propiedades se encargan de mostrar un elemento en una posición exacta en la página:

`left`. Permite indicar la coordenada izquierda del elemento. Tomada desde el margen izquierdo

`top`. Coordenada superior. Tomada desde el borde superior.

`bottom`. Coordenada inferior. Distancia al borde inferior

`right`. Coordenada derecha. Distancia al borde derecho.

Las coordenadas se calculan desde la referencia de coordenadas establecidas por la propiedad `position`. Si es `fixed` se toma los bordes de la ventana. Es incompatible usar `bottom` y `right` y, a la vez, los tamaños con `width` y `height`.

Coordenada z

La pantalla es un medio en dos dimensiones y por tanto la posición de los elementos se indica con las coordenadas **X** e **Y**. Sin embargo se puede simular la tercera dimensión (la profundidad) mediante una coordenada ficticia, **Z**.

Realmente, lo que aporta es la posibilidad de que los elementos se solapen y podamos decidir qué elemento está por encima, de modo que el que tenga la coordenada z más alta diríamos que está más cerca del espectador y por lo tanto tiene prioridad; es decir, quedará *encima* de los otros.

Es el mismo efecto que logran los sistemas operativos gráficos como Windows u OS X cuando tenemos dos ventanas solapadas; una de ellas tapa a la otra entendiéndolo que está *por delante*.

La propiedad que permite controlar la coordenada z es **z-index** a la que se le otorga un número entero de modo que cuanto mayor sea, más quedará por delante de los otros elementos. El texto normal (de elementos que no tengan marcado un **z-index**) está en el nivel cero; con lo cual, si indicamos un nivel negativo en la propiedad **z-index** de un elemento, entenderemos que por detrás del contenido normal (no se verá el elemento, quedará tapado).

Hay que tener en cuenta que incluso a las capas se les puedan dar valores negativos, entendiendo que por debajo del cero estamos por debajo del texto normal de la página.

Tema 5 Tablas

Las tablas son uno de los elementos fundamentales de HTML para mejorar la puesta visual de las páginas web. Antes de que aparecieran las capacidades de CSS para realizar maquetaciones avanzadas en las páginas web, las tablas eran la única forma de crear documentos HTML con disposiciones complejas.

Una tabla típica podría ser esta:

Equipo	Ciudad	Títulos NBA
Lakers	Los Ángeles	16
Celtics	Boston	17
Spurs	San Antonio	5
Bulls	Chicago	6

Una tabla normal se entiende que es un conjunto de datos presentados en forma de filas y columnas. En HTML las tablas tienen muchas posibilidades, lo que ha permitido, tradicionalmente, ser utilizadas como elemento de maquetación de páginas complejas.

En especial permiten unir varias celdas para formar tablas con una disposición irregular, lo que permite realizar acabados muy complejos en las páginas. Aún más: se permite incluso colocar una tabla dentro de una celda lo que nos proporciona aun más libertad en la maquetación de páginas.

Sin embargo, hoy en día se recomienda no usar las tablas de esta forma ya que las mejoras que proporciona CSS en el uso de capas, permiten olvidar a las tablas como recurso de maquetación.

El uso actual de las tablas es más semántico: indicar una disposición en celdas las cuales se marcan también de forma semántica (celdas de cabecera, celdas normales, filas, columnas, pie de tabla, título de tabla, etc.).

Creación de tablas simples

En HTML cada tabla está asociado a un elemento `table`, dentro de este elemento se indican las filas mediante el elemento `tr` y dentro de cada fila se indican las celdas mediante elementos `td`.

En HTML 5 las etiquetas `table` y `tr` no tienen atributos, aunque hay atributos, como por ejemplo el atributo `border`, que reconocen los navegadores por estar presentes en versiones anteriores de HTML. Es, como siempre, el lenguaje CSS el que determina la apariencia y formato de la tabla.

Ejemplo de tabla:

```
<table>
  <tr>
    <td></td>
    <td>Lunes</td>
```

```

        <td>Martes</td>
        <td>Miércoles</td>
        <td>Jueves</td>
        <td>Viernes</td>
    </tr>
    <tr>
        <td>10:30</td>
        <td>Matemáticas</td>
        <td>Geografía</td>
        <td>Física</td>
        <td>Dibujo</td>
        <td>Matemáticas</td>
    </tr>
    <tr>
        <td>11:30</td>
        <td>Inglés</td>
        <td>Lenguaje</td>
        <td>Geografía</td>
        <td>Química</td>
        <td>Física</td>
    </tr>
</table>

```

Obtiene el resultado:

	Lunes	Martes	Miércoles	Jueves	Viernes
10:30	Matemáticas	Geografía	Física	Dibujo	Matemáticas
11:30	Inglés	Lenguaje	Geografía	Química	Física

Celdas de cabecera

Es muy habitual que las tablas muestren datos y que estos posean celdas que sirvan para describirles. Esas celdas se consideran de cabecera y se marcan con **th**.

Ejemplo:

```

<table>
  <tr>
    <th>&nbsp;</th>
    <th>Lunes</th>
    <th>Martes</th>
    <th>Miércoles</th>
    <th>Jueves</th>
    <th>Viernes</th>
  </tr>
  <tr>
    <th>10:30</th>
    <td>Matemáticas</td>
    <td>Geografía</td>
    <td>Física</td>

```

```

        <td>Dibujo</td>
        <td>Matemáticas</td>
    </tr>
    <tr>
        <th>11:30</th>
        <td>Inglés</td>
        <td>Lenguaje</td>
        <td>Geografía</td>
        <td>Química</td>
        <td>Física</td>
    </tr>
</table>

```

En el resultado sólo se aprecia que el navegador colorea en negrita las celdas de cabecera. Pero con ayuda de CSS podríamos diferenciarlas más. Resultado:

	Lunes	Martes	Miércoles	Jueves	Viernes
10:30	Matemáticas	Geografía	Física	Dibujo	Matemáticas
11:30	Inglés	Lenguaje	Geografía	Química	Física

Títulos

A las tablas se les puede poner un título con ayuda de la etiqueta `caption`. Ejemplo:

```

<table style="border:1px solid black">
    <caption>Ventas</caption>
    <tr>
        <th>Hardware</th>
        <td>12.190 €</td>
    </tr>

    <tr>
        <th>Software</th>
        <td>9.870 €</td>
    </tr>
</table>

```

En el código se ha utilizado al atributo prohibido `border` para que quede más claro la forma de considerar el título por parte de los navegadores. El resultado del código es:

Ventas	
Hardware	12.190 €
Software	9.870 €

Agrupación de filas

Hay tres elementos HTML que sirven para diferenciar las tres partes principales de una tabla, son:

thead. Sirve para indicar las filas que forman la cabecera de la tabla

tfoot. Indica el pie de la tabla

`tbody`. Indica el cuerpo de la tabla

De esa forma se podrá más adelante dar formato diferencial a cada parte.

Ejemplo de uso:

```
<table border="1" rules="groups">
  <caption>Ventas por secciones</caption>
  <thead>
    <tr>
      <td>&nbsp;</td>
      <td>Hardware</td>
      <td>Software</td>
    </tr>
  </thead>

  <tfoot>
    <tr>
      <th>Total</th>
      <th>25000</th>
      <th>22000</th>
    </tr>
  </tfoot>

  <tbody>
    <tr>
      <th>Enero</th>
      <td>12000</td>
      <td>15000</td>
    </tr>
    <tr>
      <th>Febrero</th>
      <td>13000</td>
      <td>9000</td>
    </tr>
  </tbody>
</table>
```

Resultado (se ha utilizado `border` como atributo para visualizar el borde, y un atributo prohibido llamado `rules` con el valor `groups`, que nos permite distinguir mejor la agrupación realizada):

Agrupación de columnas

Elemento `colgroup`

Permite realizar agrupaciones de columnas de una tabla con la finalidad de dar un formato homogéneo a cada grupo de columnas indicado. El único atributo permitido en HTML 5 es `span` que sirve para indicar el número de columnas que agrupamos.

Otra posibilidad, más recomendable, es usar dentro de `colgroup` elementos `col` los cuales dan un mayor sentido y son más versátiles.

Esta etiqueta debe de ser la primera de una tabla excepto si se usa título en la tabla (etiqueta `caption`) en cuyo caso será la segunda.

Elemento `col`

Se usa dentro de la anterior para dar propiedades a cada grupo de columnas realizado. Permite el uso del atributo `col` para indicar cuántas columnas se incluyen en el grupo.

Ejemplo (usa CSS, para ver mejor la forma de agrupar las columnas y para diferenciar, coloreando en negro, la primera fila):

```
<table>
```

```
  <colgroup>
```

```
    <col style="background-color: gray">
```

```
    <col span="2" style="background-color: yellow;">
```

```
  </colgroup>  <thead>
```

```
    <tr style="background-color: black; color: white">
```

```
      <th>País</th>
```

```
      <th>Capital</th>
```

```
      <th>Otras ciudades</th>
```

```
    </tr>
```

```
  </thead>
```

```
  <tbody>
```

```
    <tr>
```

```
      <th>España</th>
```

```
      <td>Madrid</td>
```

```
      <td>Barcelona,  
        Bilbao</td>
```

```
        Zaragoza,
```

```
        Sevilla,
```

```
    </tr>
```

```
    <tr>
```

```
      <th>Francia</th>
```

```
      <td>París</td>
```

```
      <td>Lyon, Marsella, Nantes, Toulouse</td>
```

```
    </tr>
```

```
    <tr>
```

```
      <th>Portugal</th>
```

```

        <td>Lisboa</td>

        <td>Oporto, Coimbra</td>

    </tr>

</tbody>

</table>

```

Resultado:

País	Capital	Otras ciudades
España	Madrid	Barcelona, Zaragoza, Sevilla, Valencia, Bilbao
Francia	París	Lyon, Marsella, Nantes, Toulouse
Portugal	Lisboa	Oporto, Coimbra

Combinar celdas

Es posible unir celdas y de esta forma conseguir tablas de formas caprichosas que permiten una maquetación más poderosa.

Las etiquetas de columna (**td** y **th**) son las que poseen los atributos que permiten esta operación. En concreto son los atributos:

atributo	significado
colspan	Combina la celda actual con el número de celdas a la derecha que se indique. Por ejemplo colspan="3" une esta celda con las dos que tiene a su derecha, formando una combinación de tres celdas en horizontal.
rowspan	Combina la celda actual con el número de celdas hacia abajo que se indique. Por ejemplo rowspan="3" une esta celda con las dos que tiene hacia abajo, formando una combinación de tres celdas en vertical.

Hay que tener en cuenta que se pueden usar ambos atributos a la vez, lo que permite obtener celdas combinadas de otras columnas y filas a la vez.

Ejemplo de uso:

```

<!doctype html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <title>Tabla combinada</title>

    <style>

        table{

            width:50%;

        }

```

```
td{  
    border:2px solid black;  
}  
</style>  
</head>  
<body>  
<table>  
  <tr>  
    <td>&nbsp;</td>  
    <td>&nbsp;</td>  
    <td>  
  </tr>  
  <tr>  
    <td>&nbsp;</td>  
    <td>  
    <td>  
  </tr>  
  <tr>  
    <td>  
    <td>&nbsp;</td>  
    <td>&nbsp;</td>  
  </tr>  
  <tr>  
    <td>  
  </tr>  
  <tr>  
    <td>  
  </tr>  
</table>  
</body>
```

Formato CSS para tablas

La mayoría de propiedades de uso habitual en CSS (bordes, colores, tipos de letra, alineaciones,...) son perfectamente aplicables a las tablas. Sin embargo hay propiedades CSS que, explícitamente, sirven para modificar la forma de visualizar los elementos de una tabla.

No son muchas propiedades, pero sí son extremadamente útiles e importantes

Espaciado y borde en las tablas

Las propiedades CSS específicas para tablas son:

border-collapse. Permite indicar cómo manejar los bordes adyacentes entre elementos. Posibles valores:

collapse. Se unen los bordes adyacentes para formar un único borde. Es decir no hay espacio entre bordes adyacentes.

separate. Los bordes adyacentes se mantienen separados. Es el valor por defecto.

Ejemplo:

```
table{  
    width:100%;  
}  
td{  
    border:2px solid black;  
}
```

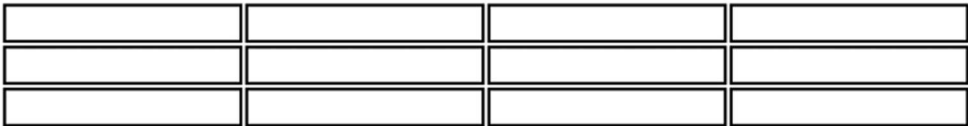


Ilustración 85. Tabla con bordes en modo **separate** (modo por defecto)

La tabla se muestra en modo **separate**. Sin embargo indicando modo **collapse** en la etiqueta **table**:

```
table{  
    width:100%;  
    border-collapse:collapse;  
}  
td{  
    border:2px solid black;  
}
```

Resultado:

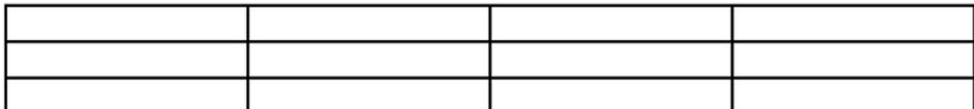


Ilustración 86. Tabla con bordes en modo **collapse**

Hay que fijarse en que es en el elemento **table** (el contenedor general de la tabla) en el que se indica esta propiedad.

border-spacing. Indica el espacio entre bordes. Admite dos medidas, la primera se utiliza para la distancia horizontal y la segunda para la vertical. Si se utiliza una medida se referirá a ambas distancias.

Ejemplo:

```
border-spacing:10px 5px;
```

caption-side. Posición del título de la tabla (elemento **caption**). Posibilidades: **top** (arriba, valor por defecto), **bottom** (abajo).

empty-cells. Indica si deseamos mostrar las celdas vacías de la tabla. Valores: **show** (mostrar, valor por defecto) y **hide** (ocultar). No funciona (*hide*) con los bordes colapsados.

table-layout. Propiedad presente en los últimos navegadores (no en **Internet Explorer**) permite indicar la forma en la que se calcula la anchura de las celdas. Por defecto toma el valor **auto**, en el que, aunque hayamos indicado una anchura fija para una celda, el tamaño de la misma depende del contenido. **fixed**, sin embargo deja la tabla con el tamaño fijado por las propiedades de anchura de columna (**width**).

El problema es cuando un contenido sobrepasa el tamaño de una celda (una imagen por ejemplo). En el primer caso la celda se amplía para dar cabida al contenido. Eso significa que el navegador no puede renderizar la tabla hasta no haber cargado todo su contenido.

En el segundo caso la tabla se dibuja al instante aunque los contenidos no hayan llegado. Aunque en ese caso esos contenidos sobresalgan de la tabla. Solo es interesante **fixed** si sabemos seguro el tamaño de la tabla por adelantado.

Ejemplo:

```
<!doctype
<html
  <head>
    <meta
    <title>Document</title>
    <style>
      table{
        width:100%;
        border-spacing:
        border:2px
      }
      td{
        border:2px
      }
    </style>
  </head>
  <body>
    <table>
      <tr>
        <td>&nbsp;</td>
```

```

        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
    </tr>
    <tr>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
    </tr>
    <tr>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
    </tr>
</table>
</html>

```


caption-side. Propiedad utilizada para la etiqueta **caption** que contiene el título de una tabla. Permite indicar si queremos que el título esté por encima de la tabla (valor **top**, valor por defecto) o por debajo (valor **bottom**)

empty-cells. Permite especificar si los bordes y sombreados de la tabla se aplican a las celdas vacías. Valores:

show. Se aplican bordes y sombreados a esas celdas (valor por defecto)

hide. No se aplican los bordes y sombreados a las celdas vacías.

table-layout. Indica la forma en la que se determina la anchura de la tabla y las celdas. Posibles valores:

auto. Valor por defecto. Aunque hayamos indicado anchura para tabla y/o celdas, su tamaño se determina por el contenido de las mismas (si el contenido requiere mayor anchura, así se hará).

fixed. Predomina la anchura establecida por las propiedades y no por el contenido

Tema 6 Formularios

Introducción

Las aplicaciones web están basadas en dos tipos de acciones:

- Dar información al usuario
- Que el usuario proporcione información.

La herramienta para ejecutar la última acción son los formularios web.

Para implementar un formulario se utiliza la etiqueta **<form>** que designa al **elemento formulario de HTML**. Este elemento es un contenedor para diferentes tipos de elementos de entrada, tales como: campos de texto, casillas de verificación, botones de opción, botones de envío, etc.

El elemento formulario tiene atributos comunes a todos los elementos HTML y propios que van a identificar en qué modo se realiza el envío de información, qué acción va a recibir esta información, si se va autocompletar o no esta información y si va a validarse.

El elemento formulario debe seguir una determinada estructura, llamados elementos de formulario, siendo uno de los más importante y usados los tipos de entrada

Atributos de formulario HTML

El elemento formulario está parametrizado por los siguientes atributos:

Atributo	Descripción
<code>accept-charset</code>	Especifica las codificaciones de caracteres utilizadas para el envío de formularios
<code>action</code>	Especifica dónde enviar los datos del formulario cuando se envía un formulario, si se omite este atributo el valor que toma es la página actual
<code>autocomplete</code>	Especifica si un formulario debe tener autocompletar activado o desactivado, valores on/off. Si esta activado el navegador

	completará el formulario con valores anteriormente introducidos en otros.						
<code>enctype</code>	Especifica cómo se deben codificar los datos de formulario al enviarlos al servidor (solo para el método "post")						
<code>method</code>	Especifica el método HTTP que se utilizará al enviar datos de formulario. Los métodos usuales para mandar información son GET, POST y PUT. El método por defecto es GET. Ver notas al final de la tabla.						
<code>name</code>	Especifica el nombre del formulario.						
<code>novalidate</code>	Especifica que el formulario no debe validarse cuando se envía, si se indica este atributo el formulario no se validará en HTML.						
<code>rel</code>	Especifica la relación entre un recurso vinculado y el documento actual.						
<code>target</code>	<p>Especifica dónde mostrar la respuesta que se recibe después de enviar el formulario:</p> <table> <tr> <td><code>_blank</code></td><td>La respuesta se mostrará en otra ventana o pestaña nueva.</td></tr> <tr> <td><code>_self</code></td><td>La respuesta se mostrará en la misma ventana.</td></tr> <tr> <td><code>_parent</code></td><td>La respuesta se mostrará en el frame padre.</td></tr> </table>	<code>_blank</code>	La respuesta se mostrará en otra ventana o pestaña nueva.	<code>_self</code>	La respuesta se mostrará en la misma ventana.	<code>_parent</code>	La respuesta se mostrará en el frame padre.
<code>_blank</code>	La respuesta se mostrará en otra ventana o pestaña nueva.						
<code>_self</code>	La respuesta se mostrará en la misma ventana.						
<code>_parent</code>	La respuesta se mostrará en el frame padre.						

	<code>_top</code>	La respuesta se mostrará en el cuerpo entero de la ventana.
	<code>frameName</code>	La respuesta se mostrará dentro de un frame.

Notas sobre GET:

- Anexa los datos del formulario a la dirección URL, en pares nombre/valor
- Los datos son visibles en la URL
- La longitud de una dirección URL es limitada (2048 caracteres)
- Es útil para los envíos de formularios en los que un usuario desea marcar el resultado
- GET es bueno para datos no seguros, como cadenas de consulta en Google

Notas sobre POST:

- Anexa los datos del formulario dentro del cuerpo de la solicitud HTTP (los datos del formulario enviados no se muestran en la dirección URL)
- POST no tiene limitaciones de tamaño y se puede utilizar para enviar grandes cantidades de datos.
- Los envíos de formularios con POST no se pueden marcar

HTML <form> Elementos

El elemento HTML puede contener uno o varios de los siguientes elementos de formulario:

Elemento <input>

Es uno de los elementos principales del formulario, el elemento <input> es el llamado tipo de entrada, ya que según sea el valor de su atributo type así será el formato y/o valor de la entrada de datos del usuario, además de cómo se mostrará en los navegadores.

Cada elemento <input> también designa un campo del formulario. Después se verán más campos de formulario que no serán elementos <input>.

Se verán todos los tipos de entrada en el siguiente punto del temario.

Ejemplo

```
<label for="fnombre">Nombre:</label>
<input type="text" id="fnombre" name="fnombre">
```

Elemento <label>

Se define una texto que está asociado a un campo determinado del formulario.

Estas etiquetas se utilizan además en los lectores de pantalla, ya que el lector de pantalla leerá en voz alta la etiqueta cuando el usuario se centra en el elemento de entrada.

El elemento también ayuda a los usuarios que tienen dificultades para hacer clic en regiones muy pequeñas (como botones de opción o casillas de verificación) - porque cuando el usuario hace clic en el texto dentro del elemento, alterna el botón de opción / casilla de verificación.

El atributo for de la etiqueta debe ser igual al atributo id del elemento para enlazarlos.

Ejemplo

```
<label for="fnombre">Nombre:</label>
<input type="text" id="fnombre" name="fnombre">
```

Elemento <select>

El elemento define un campo que va a recoger su valor de la selección una lista desplegable de opciones:

De forma predeterminada, se selecciona el primer elemento de la lista desplegable.

Ejemplo

```
<label for="colores">Elegir un color:</label>
<select id="colores" name="colores">
  <option value="Azul">Azul</option>
  <option value="Rojo">Rojo</option>
  <option value="Verde">Verde</option>
  <option value="Amarillo">Amarillo</option>
</select>
```

Para definir una opción preseleccionado, agregue el atributo a la opción

Ejemplo

```
<option value="Verde" selected>Verde</option>
```

Se utiliza el atributo size para especificar el número de valores visibles

Ejemplo

```
<label for="colores">Elegir un color:</label>
<select id="colores" name="colores" size="3">
  <option value="Azul">Azul</option>
  <option value="Rojo">Rojo</option>
  <option value="Verde">Verde</option>
  <option value="Amarillo">Amarillo</option>
</select>
```

Utilice el atributo `múltiple` para permitir que el usuario seleccione más de un valor, automáticamente el campo identificará una lista de valores.

Ejemplo

```
<label for="colores">Elegir un color:</label>
<select id="colores" name="colores" size="4" multiple>
  <option value="Azul">Azul</option>
  <option value="Rojo">Rojo</option>
  <option value="Verde">Verde</option>
  <option value="Amarillo">Amarillo</option>
</select>
```

El `<textarea>` Elemento

El elemento define un campo de entrada de varias líneas (un área de texto).

El atributo `rows` especifica el número visible de líneas en un área de texto.

El atributo `cols` especifica el ancho visible de un área de texto.

Ejemplo

```
<textarea name="comentario" rows="10" cols="30">Algún comentario
sobre los datos
</textarea>
```

También puede definir el tamaño del área de texto mediante CSS:

Ejemplo

```
<textarea name="comentario" style="width:200px; height:600px;">
Algún comentario sobre los datos
</textarea>
```


Elemento <button>

El elemento define un botón en el que se puede hacer clic y normalmente se procederá al envío del formulario (`submit`), también se suele utilizar para limpiar el formulario (`reset`).

Diferentes navegadores pueden utilizar diferentes tipos predeterminados para el elemento de botón, siempre hay que especificar el atributo `type`.

Ejemplo

```
<button type="button" onclick="alert('Hola Mundo!')">
  ¡Púlsame!
</button>
```

Elementos <fieldset> y <legend>

El elemento `fieldset` se utiliza para agrupar datos relacionados en un formulario

El elemento `legend` define un título para el elemento

Ejemplo

```
<form action="/accion.php">
  <fieldset>
    <legend>Ficha:</legend>
    <label for="fnombre">Nombre:</label><br>
    <input type="text" id="fnombre" name="fnombre" value="John"><br>
    <label for="fapellidos">Apellidos:</label><br>
    <input type="text" id="fapellidos" name="fapellidos" value="Doe">
    <br><input type="submit" value="Submit">
  </fieldset>
</form>
```

Elemento <datalist>

El elemento `datalist` especifica una lista de opciones predefinidas para un elemento `input`

Los usuarios verán una lista desplegable de las opciones predefinidas a medida que introducen datos.

El atributo `list` del elemento `input`, debe hacer referencia al atributo `id` del elemento `datalist`.

Ejemplo

```
<form action="/accion.php">
  <input list="colores">
  <datalist id="colores">
    <option value="Azul">
    <option value="Rojo">
    <option value="Verde">
    <option value="Amarillo">
    <option value="Morado">
  </datalist>
</form>
```

El elemento <output>

El elemento output representa el resultado de un cálculo (como uno realizado por un script).

Ejemplo

Realice un cálculo y muestre el resultado en un elemento output:

```
<form action="/accion.php"
  oninput="x.value=parseInt(a.value)+parseInt(b.value)">
  0
  <input type="range" id="a" name="a" value="50">
  100 +
  <input type="number" id="b" name="b" value="50">
  =
  <output name="x" for="a b"></output>
  <br><br>
  <input type="submit">
</form>
```

Tipos de entrada HTML. Atributo type del elemento input.

En este apartado enumeraremos todos los tipos de entrada que puede tomar el elemento input. Estos se van a designar por el atributo type.

El valor por defecto de type es "text".

Estos son los diferentes tipos de entrada que puede utilizar en HTML:

- <input type="button">
- <input type="checkbox">

- `<input type="color">`
- `<input type="date">`
- `<input type="datetime-local">`
- `<input type="email">`
- `<input type="file">`
- `<input type="hidden">`
- `<input type="image">`
- `<input type="month">`
- `<input type="number">`
- `<input type="password">`
- `<input type="radio">`
- `<input type="range">`
- `<input type="reset">`
- `<input type="search">`
- `<input type="submit">`
- `<input type="tel">`
- `<input type="text">`
- `<input type="time">`
- `<input type="url">`
- `<input type="week">`

Text

Define un campo de entrada de texto de una sola línea:

Ejemplo

```
<form>
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre"><br>
  <label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos">
</form>
```

Password

Los caracteres de un campo de contraseña se enmascaran (se muestran como asteriscos o círculos).

Ejemplo

```
<form>
  <label for="username">Username:</label><br>
  <input type="text" id="username" name="username"><br>
  <label for="pwd">Password:</label><br>
```

```
<input type="password" id="pwd" name="pwd">
</form>
```

Submit

Define un botón para enviar datos de formulario a un controlador de formularios.

El controlador de formularios suele ser una página de servidor con un script para procesar los datos de entrada.

Ejemplo

```
<form action="/action_page.php">
<label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre"><br>
  <label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos"><br>
  <input type="submit" value="Submit">
</form>
```

Si se omite el atributo value del botón de envío, el botón obtendrá un texto predeterminado:

Ejemplo

```
<label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre"><br>
  <label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos"><br>
  <input type="submit" >
</form>
```

Reset

Define un **botón de restablecimiento** que restablecerá todos los valores de formulario a sus valores predeterminados:

Ejemplo

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
  <input type="reset">
</form>
```

Radio

Los botones de opción permiten al usuario seleccionar SOLAMENTE UNA de un número limitado de opciones:

Ejemplo

```
<form>
  <input type="radio" id="masculino" name="genero" value="masculino"
">
  <label for="masculino">Masculino</label><br>
  <input type="radio" id="femenino" name="generico" value="femenino"
">
  <label for="femenino">Femenino</label><br>
  <input type="radio" id="otros" name="genero" value="otros">
  <label for="otros">Otros</label>
</form>
```

Checkbox

Las casillas de verificación permiten a un usuario seleccionar cero o más opciones de un número limitado de opciones.

Ejemplo

```
<form>
  <input type="checkbox" id="vehiculo1" name="vehiculo1" value="Bicicleta">
  <label for="vehiculo1"> Bicicleta</label><br>
  <input type="checkbox" id="vehiculo2" name="vehiculo2" value="Coche">
  <label for="vehiculo2"> Coche</label><br>
  <input type="checkbox" id="vehiculo3" name="vehiculo3" value="Barco">
  <label for="vehiculo"> Barco</form>
```

Button

Un botón se utiliza para ejecutar una función javascript a través de un evento

Ejemplo

```
<input type="button" onclick="alert('Hello World!')" value="Click Me!">
```

Color

Se utiliza para los campos de entrada que deben contener un color.

Dependiendo de la compatibilidad con el navegador, un selector de color puede aparecer en el campo de entrada.

Ejemplo

```
<form>
  <label for="favcolor">Seleccionar un color:</label>
  <input type="color" id="favcolor" name="favcolor">
</form>
```

Date

Se utiliza para los campos de entrada que deben contener una fecha

Dependiendo de la compatibilidad con el navegador, un selector de fecha puede aparecer en el campo de entrada. Aparecerá un DatePicker

Ejemplo

```
<form>
  <label for="fechaNacimiento">FechaNacimiento:</label>
  <input type="date" id=" fechaNacimiento " name=" fechaNacimiento
">
</form>
```

También puede utilizar los atributos `min` y `max` para agregar restricciones a las fechas:minmax

Ejemplo

```
<form>
  <label for="fechamax">Introducir una fecha anterior a 1980-01-
01:</label>
  <input type="date" id="fechamax" name="fechamax" max="1979-12-
31"><br><br>
  <label for="fechamin">Introducir una fecha después 2000-01-
01:</label>
  <input type="date" id="fechamin" name="fechamin" min="2000-01-
02">
</form>
```

Datetime-local

Especifica un campo de entrada de fecha y hora, sin zona horaria

Dependiendo de la compatibilidad con el navegador, un selector de fecha puede aparecer en el campo de entrada.

Ejemplo

```
<form>
  <label for="fechaNotificacion">Fecha notificacion:</label>
  <input type="datetime-
local" id="fechaNotificacion" name="fechaNotificacion ">
</form>
```

Email

El se utiliza para los campos de entrada que deben contener una dirección de correo electrónico.

Dependiendo del soporte del navegador, la dirección de correo electrónico se puede validar automáticamente cuando se envía.

Algunos smartphones reconocen el tipo de correo electrónico y añaden ".com" al teclado para que coincida con la entrada de correo electrónico.

Ejemplo

```
<form>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email">
</form>
```

File

Define un campo de selección de archivos y un botón "Examinar" para las cargas de archivos

Ejemplo

```
<form>
  <label for="fichero">Seleccionar un fichero:</label>
  <input type="file" id="fichero" name="fichero">
</form>
```

Month

Permite al usuario seleccionar un mes y un año.

Dependiendo de la compatibilidad con el navegador, un selector de fecha puede aparecer en el campo de entrada.

Ejemplo

```
<form>
  <label for="mesNacimiento">Mes Nacimiento:</label>
```

```
<input type="month" id="mesNacimiento" name="mesNacimiento">
</form>
```

Number

Define un campo de entrada **numérico**.

También puede establecer restricciones sobre qué números se aceptan.

En el ejemplo siguiente se muestra un campo de entrada numérico, donde puede introducir un valor de 1 a 5:

Ejemplo

```
<form>
  <label for="cantidad">Cantidad (entre 1 y 5):</label>
  <input type="number" id="cantidad" name="cantidad" min="1" max="5"
">
</form>
```

En el ejemplo siguiente se muestra un campo de entrada numérico, donde puede especificar un valor de 0 a 100, en pasos de 10. El valor predeterminado es 30:

Ejemplo

```
<form>
  <label for="cantidad">Quantity:</label>
  <input type="number" id="cantidad" name="cantidad" min="0" max="1
00" step="10" value="30">
</form>
```

Range

Define un control para introducir un número cuyo valor exacto no es importante (como un control deslizante). El rango predeterminado es de 0 a 100. Sin embargo, puede establecer restricciones sobre qué números se aceptan con los atributos `min`, `max` y `step`

Ejemplo

```
<form>
  <label for="vol">Volume (between 0 and 50):</label>
  <input type="range" id="vol" name="vol" min="0" max="50">
</form>
```


Search

El se utiliza para los campos de búsqueda (un campo de búsqueda se comporta como un campo de texto normal).

Ejemplo

```
<form>
  <label for="gsearch"> Búsqueda Google:</label>
  <input type="search" id="gsearch" name="gsearch">
</form>
```

Tel

El se utiliza para los campos de entrada que deben contener un número de teléfono

Ejemplo

```
<form>
  <label for="phone">Introduzca su teléfono:</label>
  <input type="tel" id="phone" name="phone" pattern="[0-9]{9}">
</form>
```

Time

Permite al usuario seleccionar una hora (sin zona horaria).

Dependiendo de la compatibilidad con el navegador, puede aparecer un selector de tiempo en el campo de entrada.

Ejemplo

```
<form>
  <label for="appt">Seleccionar una hora:</label>
  <input type="time" id="appt" name="appt">
</form>
```

Url

El se utiliza para los campos de entrada que deben contener una dirección URL

Dependiendo de la compatibilidad con el navegador, el campo url se puede validar automáticamente cuando se envía.

Algunos teléfonos inteligentes reconocen el tipo de URL y añaden ".com" al teclado para que coincida con la entrada url.

Ejemplo

```
<form>
  <label for="homepage">Introduzca la url de inicio:</label>
  <input type="url" id="homepage" name="homepage">
</form>
```

Week

Permite al usuario seleccionar una semana y un año.

Dependiendo de la compatibilidad con el navegador, un selector de fecha puede aparecer en el campo de entrada.

Ejemplo

```
<form>
  <label for="week">Seleccionar una semana:</label>
  <input type="week" id="week" name="week">
</form>
```

Atributos de entrada

En este capítulo se describen los diferentes atributos para el elemento `input`

Value

Especifica un valor inicial para un campo de entrada

Ejemplo

Campos de entrada con valores iniciales (predeterminados):

```
<form>
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre" value="John"><br>
  <label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos" value="Doe">
</form>
```

Readonly

El atributo de entrada especifica que un campo de entrada es de solo lectura.

Un campo de entrada de solo lectura no se puede modificar (sin embargo, un usuario puede tabularlo, resaltarlo y copiar el texto de él).

¡El valor de un campo de entrada de solo lectura se enviará al enviar el formulario!

Ejemplo

Un campo de entrada de solo lectura:

```
<form>
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre" value="John"
readonly ><br>
  <label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos" value="Doe">
</form>
```

Disabled

El atributo de entrada especifica que se debe deshabilitar un campo de entrada.

Un campo de entrada deshabilitado es inutilizable y no se puede hacer clic.

¡El **valor** de un campo de entrada deshabilitado no se enviará al enviar el formulario!

Ejemplo

Un campo de entrada deshabilitado:

```
<form>
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre" value="John"
disabled ><br>
  <label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos" value="Doe">
</form>
```

Size

El atributo de entrada especifica el ancho visible, en caracteres, de un campo de entrada.

El valor predeterminado para es 20.size

Nota: El atributo funciona con los siguientes tipos de entrada: text, search, tel, url, email y password.

Ejemplo

Establezca un ancho para un campo de entrada:

```
<form>
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre" size="50"><br>
  <label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos">
</form>
```

Maxlength

El atributo de entrada especifica el número máximo de caracteres permitidos en un campo de entrada.

Nota: Cuando se establece a, el campo de entrada no aceptará más que el número especificado de caracteres. Sin embargo, este atributo no proporciona ningún comentario. Por lo tanto, si desea alertar al usuario, debe escribir código JavaScript.

Ejemplo

Establezca una longitud máxima para un campo de entrada:

```
<form>
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre" size="50"><br>
  <label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos"
maxlength="4" size="4">>
</form>
```

Min Max

La entrada y los atributos especifican los valores mínimo y máximo para un campo de entrada.

Los atributos y los atributos funcionan con los siguientes tipos de entrada: number, range, date, fecha y date-local, month, time y week.

Se utilizan los atributos max y min juntos para crear un rango de valores legales.

Ejemplo

Establezca una fecha máxima, una fecha mínima y un rango de valores legales:

```
<form>
  <label for="datemax">Enter a date before 1980-01-01:</label>
  <input type="date" id="datemax" name="datemax" max="1979-12-31"><br><br>

  <label for="datemin">Enter a date after 2000-01-01:</label>
  <input type="date" id="datemin" name="datemin" min="2000-01-02"><br><br>

  <label for="quantity">Quantity (between 1 and 5):</label>
  <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>
```

Múltiple

El atributo de entrada especifica que el usuario puede introducir más de un valor en un campo de entrada.multiple

El atributo funciona con los siguientes tipos de entrada: email y file

Ejemplo

Un campo de carga de archivos que acepta varios valores:

```
<form>
  <label for="fichero">Seleccionar un fichero:</label>
  <input type="file" id="fichero" name="fichero" multiple >
</form>
```

Pattern

El atributo input especifica una expresión regular con la que se comprueba el valor del campo de entrada, cuando se envía el formulario.

El atributo funciona con los siguientes tipos de entrada: text, date, search, url, tel, email y password.

.

Ejemplo

Un campo de entrada que solo puede contener tres letras (sin números ni caracteres especiales):

```
<form>
  <label for="codigoPostal">Código postal:</label>
  <input type="text" id="codigoPostal " name="codigoPostal"
    pattern="[0-9]{5}" title="Cinco dígitos">
</form>
```

Placeholder

El atributo de entrada especifica una sugerencia corta que describe el valor esperado de un campo de entrada (un valor de muestra o una breve descripción del formato esperado).

La sugerencia corta se muestra en el campo de entrada antes de que el usuario introduzca un valor.

El atributo funciona con los siguientes tipos de entrada: text, search, url, tel, email y password.

Ejemplo

Un campo de entrada con un texto de marcador de posición:

```
<form>
  <label for="codigoPostal">Código postal:</label>
  <input type="text" id="codigoPostal " name="codigoPostal"
    pattern="[0-9]{5}" title="Cinco dígitos" placeholder="12345">
</form>
```

Required

El atributo de entrada especifica que se debe rellenar un campo de entrada antes de enviar el formulario.

El atributo funciona con los siguientes tipos de entrada: tex, search, url, tel, email, password, date,date-local,time, number, checkbox, radio y file.

Ejemplo

Un campo de entrada requerido:

```
<form>
  <label for="codigoPostal">Código postal:</label>
  <input type="text" id="codigoPostal" name="codigoPostal"
    pattern="[0-9]{5}" title="Cinco dígitos" placeholder="12345"
    required>
</form>
```

Step

El atributo de entrada especifica los intervalos de números legales para un campo de entrada.

Ejemplo: si el paso "3", los números legales podrían ser -3, 0, 3, 6, etc.

Este atributo se puede utilizar junto con los atributos max y min para crear un rango de valores legales.

El atributo funciona con los siguientes tipos de entrada: number, range, date, date-time y date-local, month, time y week.

Ejemplo

Un campo de entrada con un intervalo de números legales especificado:

```
<form>
  <label for="puntos">Puntos:</label>
  <input type="number" id="puntos" name="puntos" step="3">
</form>
```

Autofocus

El atributo de entrada especifica que un campo de entrada debe obtener automáticamente el foco cuando se carga la página.

Ejemplo

Deje que el campo de entrada "Nombre" se enfoque automáticamente cuando se cargue la página:

```
<form>
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre" autofocus ><br>
  <label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos" >
</form>
```

Height Width

La entrada y los atributos especifican la altura y la anchura de un elemento.

Se debería especificar siempre los atributos de alto y ancho para las imágenes. Si se establecen altura y anchura, el espacio necesario para la imagen se reserva cuando se carga la página. Sin estos atributos, el navegador no conoce el tamaño de la imagen y no puede reservar el espacio adecuado para ella. El efecto será que el diseño de página cambiará durante la carga (mientras se cargan las imágenes).

Ejemplo

Defina una imagen como el botón Enviar, con atributos de alto y ancho:

```
<form>
  <input type="image" src="img_submit.gif" alt="Submit" width="48"
height="48">
</form>
```

List

El atributo de entrada hace referencia a un elemento que contiene opciones predefinidas para un elemento `<input>.list<datalist>`

Ejemplo

Un elemento `<input>` con valores predefinidos en una <lista de datos>:

```
<form action="/accion.php">
  <input list="colores">
  <datalist id="colores">
    <option value="Azul">
    <option value="Rojo">
    <option value="Verde">
    <option value="Amarillo">
    <option value="Morado">
  </datalist>
</form>
```

Autocomplete

El atributo de entrada especifica si un formulario o un campo de entrada deben tener autocompletar activado o desactivado.

Autocompletar permite al navegador predecir el valor. Cuando un usuario comienza a escribir un campo, el explorador debe mostrar opciones para rellenar el campo, en función de los valores escritos anteriormente.

El atributo funciona con y los siguientes tipos: text, search, url, tel, email, paswr, selectores de fecha, range y color.

Ejemplo

Un formulario HTML con autocompletar activado y desactivado para un campo de entrada:

```
<form action="/action_page.php" autocomplete="on">
  label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre"><br>
  <label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos">
  <input type="submit" value="Submit">
</form>
```

En algunos navegadores es posible que deba activar una función de autocompletar para que esto funcione (consulte "Preferencias" en el menú del navegador).

Formulario de entrada HTML* Atributos

En este capítulo se describen los diferentes atributos para el elemento HTML. form*<input>

Form

El atributo de entrada especifica el formulario al que pertenece el elemento.

El valor de este atributo debe ser igual al atributo id del elemento form al que pertenece.

Ejemplo

Un campo de entrada situado fuera del formulario HTML (pero sigue siendo parte del formulario):

```
<form action="/action_page.php" id="form1">
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre"><br><br>
  <input type="submit" value="Submit">
</form>

<label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos" form="form1">
```

Formaction

El atributo de entrada especifica la dirección URL del archivo que procesará la entrada cuando se envíe el formulario.

Este atributo reemplaza el atributo del elemento. action

El atributo funciona con los siguientes tipos de entrada: submit e imagen.

Ejemplo

Un formulario HTML con dos botones de envío, con diferentes acciones:

```
<form action="/action_page.php">
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre"><br><br>
  <label for="apellidos">apellidos:</label><br>
  <input type="text" id="apellidos" name="apellidos" form="form1">

  <input type="submit" value="Submit">
  <input type="submit" formaction="/action_page2.php" value="Submit
as Admin">
</form>
```

Formenctype

El atributo de entrada especifica cómo se deben codificar los datos de formulario cuando se envían (solo para los formularios con method="post").

Este atributo reemplaza el atributo `enctype` del elemento `<form>`

El atributo funciona con los siguientes tipos de entrada: submit e imagen.

Ejemplo

Un formulario con dos botones de envío. El primero envía los datos de formulario con codificación predeterminada, el segundo envía los datos de formulario codificados como "multipart/form-data":

```
<form action="/action_page_binary.asp" method="post">
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
  value="Submit as Multipart/form-data">
</form>
```

Formmethod

El atributo input define el método HTTP para enviar datos de formulario a la dirección URL de la acción.

Este atributo reemplaza el atributo `method` del elemento.

El atributo funciona con los siguientes tipos de entrada: submit e imagen.

Los datos de formulario se pueden enviar como variables de dirección URL (método "get") o como una transacción de publicación HTTP (método "post").

Notas sobre el método "get":

- Este método anexa los datos de formulario a la dirección URL en pares nombre/valor
- Este método es útil para los envíos de formularios donde un usuario desea marcar el resultado
- Hay un límite en la cantidad de datos que puede colocar en una URL (varía entre navegadores), por lo tanto, no puede estar seguro de que todos los datos del formulario se transferirán correctamente

¡Nunca se utiliza el método "get" para pasar información confidencial! (la contraseña u otra información confidencial será visible en la barra de direcciones del navegador)

Notas sobre el método "post":

- Este método envía los datos del formulario como una transacción de publicación HTTP
- Los envíos de formularios con el método "post" no se pueden marcar
- El método "post" es más sólido y seguro que "get", y "post" no tiene limitaciones de tamaño.

Ejemplo

Un formulario con dos botones de envío. El primero envía los datos de formulario con method-"get". El segundo envía los datos de formulario con method-"post":

```
<form action="/action_page.php" method="get">
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre"><br><br>
  <input type="submit" value="Submit using GET">
  <input type="submit" formmethod="post" value="Submit using POST">
</form>
```

Formtarget

La entrada de un atributo especifica un nombre o una palabra clave que indica dónde mostrar la respuesta que se recibe después de enviar el formulario.

Este atributo reemplaza el atributo `target` del elemento `<form>`

El atributo funciona con los siguientes tipos de entrada: enviar e imagen.

Ejemplo

Un formulario con dos botones de envío, con diferentes ventanas de destino:

```
<form action="/action_page.php">
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formtarget="_blank" value="Submit to a new
window/tab">
</form>
```

Formnovalidate

El atributo de entrada especifica que un elemento <input> no se debe validar cuando se envía.

Este atributo reemplaza el atributo `novalidate` del elemento.

El atributo funciona con los siguientes tipos de entrada: submit.

Ejemplo

Un formulario con dos botones de envío (con y sin validación):

```
<form action="/action_page.php">
  <label for="email">Introducir email:</label>
  <input type="email" id="email" name="email"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formnovalidate="formnovalidate"
value="Submit without validation">
</form>
```

Tema 7 Elementos multimedia

Indudablemente no se puede concebir una página web hoy en día sin incluir en ella elementos visuales atractivos. En este sentido se conoce como multimedia el hecho de incorporar medios procedentes de diversos formatos.

Esencialmente la base es la inclusión de imágenes, pero también es cada vez más común la inclusión de vídeos, sonido. También hay que tener en cuenta que hay otros elementos que proceden de tecnologías más complejas, como es el caso de los elementos de tipo Flash. Todo aquello que no sea texto, está metido respecto a este capítulo, en el mismo carro.

El hecho de que podamos incluir una imagen en la página requiere que tengamos en cuenta cómo funcionan los navegadores.

Como hemos explicado en los primeros capítulos, una página web es un documento de texto que utiliza el lenguaje etiquetado HTML, el navegador interpreta las etiquetas HTML y así da formato de forma apropiada a los distintos elementos de la página. Evidentemente cualquier contenido que no sea texto queda fuera de este planteamiento.

Por lo tanto la idea para ese contenido parte de estas premisas:

1. El contenido no textual de una página web se suele almacenar en un archivo aparte. Es decir, si deseamos incorporar una imagen, ésta deberá estar almacenada en un archivo; no se puede copiar y pegar la misma en la página al estilo de los programas como Word esa idea debe quedar desterrada de nuestra mente.

Sí es posible colocar algunos elementos multimedia directamente en el código HTML de la página, pero siempre y cuando se definan con lenguajes de etiquetas compatibles con los navegadores. Tal es el caso, por ejemplo, de SVG (formato XML de imagen vectorial) o MathML (lenguaje XML para representar ecuaciones matemáticas).

2. El archivo formará parte de nuestro sitio web, cuando publiquemos en Internet nuestro código, todos los archivos multimedia que se muestran en nuestras páginas se deben subir también, haciendo que la ruta local hacia ellos no cambie al subirlos.
3. Para que el contenido del archivo forme parte de la página, en el código de la misma se usará una etiqueta (por ejemplo `img` para las imágenes) que haga referencia a ese archivo, de modo que el contenido será un elemento más dentro de la página.
4. La etiqueta se coloca en la posición de la página en la que queremos que aparezca el contenido del archivo. Mediante la propia etiqueta, normalmente, tendremos atributos que nos permiten indicar el tamaño con el que queremos ver el contenido y otros detalles que pueden ser muy importantes para su correcta visualización.
5. El contenido del archivo será visible en el navegador si el navegador web tiene capacidad para mostrarle. Esto es fundamental, podemos intentar colocar por ejemplo una hoja de cálculo hecha en el programa Excel de Microsoft, pero no será posible porque los navegadores no tienen capacidad de mostrar ese contenido.
6. Este último punto implica una idea fundamental, sólo se puede mostrar contenido compatible con el navegador. Más exactamente, con el navegador del usuario. De nada sirve que instalemos en nuestro navegador plugins (extensiones en los navegadores)

especiales para mostrar elementos concretos. Si los usuarios no tienen ese plugin, el contenido no le podrán ver. Por lo que solo deberemos utilizar elementos multimedia que sepamos que son muy compatibles, teniendo en cuenta qué navegadores se usan habitualmente por parte de los usuarios.

Un ejemplo muy paradigmático es lo que ha ocurrido con los contenidos de tipo **Flash**. Cuando ese tipo de contenido se hizo muy popular, los navegadores incluían de base el plugin para poder mostrar elementos Flash. Incluso se exigía al usuario que su navegador incluyera dicho plugin para poder ver algunas páginas. Pero Flash ha acarreado cierta mala fama en estos últimos años por el uso extensivo de recursos y por, según algunas compañías, causar problemas en sus navegadores.

En la actualidad el plugin de Flash está en claro desuso. Ahora ya no se incluye este plugin de base en la mayoría de navegadores. Por lo tanto, es muy arriesgado para un creador de aplicaciones web crear elementos de este tipo y se ha optado por utilizar otras técnicas para dotar a las páginas de la interactividad y potencia gráfica que ofrecían los plugins de Flash.

7. Hay que tener en cuenta que el contenido multimedia suele ocupar mucho, por lo que es fundamental optimizarlo y/o aplicar técnicas de tipo **streaming** (especialmente en el caso del vídeo) que permitan ir visualizando el contenido poco a poco a medida que se descarga. Es fácil entender esta idea observando como funciona la visualización de vídeos, por ejemplo, en YouTube.

El problema de la incompatibilidad de formatos

Dejando de lado los contenidos especiales (como Flash, Silverlight u otros formatos complejos), cualquier aplicación web actual muestra imágenes y, cada vez más, vídeo y audio.

El problema es el formato de estos elementos, ya vimos en el tema anterior la cantidad de formatos diferentes que poseen las imágenes. En el caso del vídeo y el audio el problema es mayor.

El vídeo es un tipo de contenido que ocupa mucho espacio por lo que la lucha ha sido siempre comprimir al máximo su tamaño. Para ello se utiliza software especial capaz de decodificar y codificar vídeo haciendo que ocupe muy poco espacio. La potencia de los actuales procesadores permite realizar esa ardua tarea incluso a tiempo real (mientras el vídeo se reproduce).

El problema es que el navegador tiene que tener incorporado ese software, llamado popularmente **codec**. No solo eso: tiene que ser el mismo **codec**.

En el vídeo (también en el audio), los archivos se graban eligiendo un tipo concreto de vídeo, como por ejemplo puede ser **mp4**. Sin embargo el hecho es que un archivo de vídeo es un contenedor de elementos multimedia que incluye: las imágenes de vídeo, el audio, metadatos y otras informaciones. Lo normal es que se usen al menos dos codecs: uno para el audio y otro para el vídeo. Incluso el mismo formato puede usar diferentes codecs para cada cosa.

En definitiva, las posibilidades son enormes porque hay decenas de codecs (aunque algunos más populares que otros). Por lo tanto poder reproducir un elemento concreto dependerá de si el navegador incorpora los codecs concretos.

De esta forma si incorporamos vídeo a nuestra página a través de un enlace normal:

```
<a href="video1.mpeg">Ver vídeo</a>
```

El navegador buscará si disponemos del plugin apropiado para ver el vídeo. Si no es así, simplemente lo descargará en nuestro ordenador al no poder mostrar el contenido directamente.

En otros tipos de elementos multimedia pasará parecido. Como ya se ha comentado los navegadores deben ser capaces de traducir el tipo de contenido concreto, bien porque de base incorporan esta posibilidad o porque se añaden plugins para ello.

Vídeos en las páginas web

Formatos de vídeo

Como ya hemos comentado, el vídeo posee numerosos formatos relacionados con los diversos codecs y tipos de contenedores del mismo. Como cada navegador ha apostado por unos codecs concretos de vídeo, lo normal es que los creadores de contenido graben los vídeos usando varios codecs y formatos.

En la siguiente dirección disponemos de una lista de compatibilidad en audio y vídeo:

https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats#Browser_compatibility

En HTML 5 se ha simplificado, en parte, el problema ya que se habla solo de tres formatos de vídeo:

- **MP4**. Probablemente el más popular ya que es muy compatible con los navegadores de los dispositivos móviles. Se basa en el formato **MPEG-4 parte 14** de vídeo (creado por el grupo de trabajo especificación del MPEG (*Moving Pictures Experts Group*) pensada para la reproducción de vídeo en línea.

Normalmente usa el codec **H264** para el vídeo, pero también se puede usar **FLAC** y otros codecs. Para el audio se usan, normalmente, los codecs **MP3** o el **AAC**. Este contenido tiene asociado el tipo MIME (indicación oficial de contenido) **video/mp4**.

- **WebM**. Popularizado por Google, pero ya muy compatible con casi todos los navegadores se ha creado para optimizar la visualización de vídeo a través de los navegadores. Suele usar como códec **VP8**, pero ya se empieza a usar **VP9** (supera al anterior). Para el audio lo normal es que use el códec **Vorbis**. Su tipo MIME es **video/webm**.
- **OGG Vídeo**. Formato creado por la fundación **Xiph.org** bajo la premisa de ser un formato abierto y sin patentes. Suele usar el codec de vídeo **Theora** y para el audio **Vorbis** u **Opus**. La extensión de sus archivos suele ser **ogv**, pero su tipo MIME es **video/ogg**.

Inserción de vídeo

Se hace a través del elemento video. Ejemplo:

```
<video src="ejemplo.mp4"></video>
```

Atributos del elemento video

atributo uso	
src	URL al vídeo que se desea mostrar

atributo	uso
<code>width</code>	Anchura del vídeo en nuestra página
<code>height</code>	Altura del vídeo en nuestra página
<code>autoplay</code>	Usa el valor fijo <i>autoplay</i> (o no se indica valor alguno) para indicar que el vídeo se inicia automáticamente en cuanto se descargue
<code>loop</code>	Usa el valor fijo <i>loop</i> (o no se indica valor alguno) para indicar que el vídeo se ejecuta automáticamente una y otra vez
<code>controls</code>	Con valor <i>controls</i> (o no indicando valor alguno) indica que el navegador mostrará controles para que el usuario pueda controlar la reproducción del vídeo de reproducción (pausa, play,...)
<code>preload</code>	Indica al navegador una recomendación sobre cómo debemos realizar la descarga. Posibilidades: <code>auto</code> . El vídeo se descarga en cuanto se carga la página <code>none</code> . El vídeo no se descarga. Cuando el usuario pulse play, se descargará. <code>metadata</code> . Descarga los metadatos del vídeo, no el vídeo en sí
<code>poster</code>	Permite indicar la dirección URL a una imagen que se mostrará mientras el vídeo no se está reproduciendo. Si no se usa este atributo, se usa el primer fotograma del vídeo como póster.
<code>muted</code>	Atributo sin valor que marca que el vídeo se debe reproducir sin audio.

Ejemplo:

```
<video src="carrera.mp4" controls autoplay
      poster="f1.jpg">
</video>
```

Se mostrará el vídeo *carrera.mp4* de modo que se empezará a reproducir en cuanto se cargue. Antes de la carga del vídeo se mostrará la imagen *f1.jpg*, el vídeo se reproducirá inmediatamente (si que haga falta que el usuario pulse play).

Uso de diferentes formatos de vídeo

Dentro del elemento `video`, podemos incorporar elementos `source` que hagan referencia a distintas fuentes de vídeo. `source` dispone de estos atributos:

atributo	uso
<code>src</code>	URL a la fuente de vídeo
<code>type</code>	Tipo de vídeo que queremos mostrar. Se usa un tipo MIME para especificar el tipo

Ejemplo:


```

<video autoplay="autoplay" controls="controls"
poster="foto1.jpg" >
  <source src="video.mp4"
  type="video/mp4;codecs='avc1.42E01E, mp40a.40.2' " >
  <source src="video.ogv"
  type="video/ogg;codecs='theora, vorbis' " >
No se puede mostrar el video en este navegador
</video>

```

Dentro de `type`, el uso de codecs es opcional, ya que si el navegador no reconoce el formato no suele hacer caso a los `codecs` que se indiquen (aunque a veces les descarga). La idea es que si el primer formato no se reconoce (primer elemento `source`), se intenta el segundo y así sucesivamente. Si ninguno es reproducible por el navegador actual, éste mostrará la frase final tras el último `source`.

Subtítulos

Se realiza mediante elementos de tipo `track`, los cuales se colocan dentro de la etiqueta `video`. Desde hace un par de años, casi todos los navegadores manejan los subtítulos, aunque todavía hay algunas carencias al respecto.

Los subtítulos son textos que permiten traducir el contenido del vídeo. Por ejemplo un vídeo podría estar grabado en inglés y los subtítulos nos enseñarían el texto traducido en español a tiempo real. Hay varios formatos de subtítulos pero el estándar en la web es `WebVTT` (*Web Video Text Tracks*). Normalmente se utilizan archivos de texto cuya extensión es `vtt` basados en este estándar.

Ejemplo de archivo `vtt`:

```

WEBVTT

1
00:00:00.150 --> 00:00:01.300
¿Habéis visto a Sara?

2
00:00:01.540 --> 00:00:03.800
-No
-Yo tampoco la he visto

```

En este archivo la primera frase se muestra en el primer segundo del vídeo. En los dos segundos siguientes se mostrará el otro texto. En el navegador Chrome, y en algunos otros, se permite dar formato CSS al texto de los subtítulos.

El elemento `track` que permite asignar subtítulos a un vídeo posee los siguientes atributos:

atributo	uso
<code>src</code>	URL al archivo VTT o SRT que contiene los subtítulos
<code>kind</code>	Tipo de subtítulos. Posibilidades:

atributo	uso
	<p><code>subtitles</code>. Subtítulos normales</p> <p><code>captions</code>. Igual que el anterior pero con posibilidad de añadir sonidos, música y otros elementos más allá de la traducción del diálogo. El caso más utilizado este tipo de “tracks” son los audios especiales para personas con problemas de visión o textos para personas con problemas de audición.</p> <p><code>descriptions</code>. Texto alternativo e independiente de la película. También muy utilizado para ayudar a interpretar el audio del vídeo a las personas sordas.</p> <p><code>chapters</code>. Contiene los capítulos que permiten navegar de forma ágil por la película</p> <p><code>metadata</code>. Metadatos de la película. No visibles para los usuarios, se usan con JavaScript.</p>
<code>label</code>	Nombre de los subtítulos para reconocerlos en caso de haber indicado varios. Es el texto con el que aparece el nombre del idioma en el navegador al elegir un subtítulo u otro.
<code>srclang</code>	Código de idioma (oficial según la norma ISO BCP 47) de los subtítulos: por ejemplo <code>es</code> para español, <code>en</code> para inglés, <code>fr</code> francés, etc.
<code>default</code>	Indica los subtítulos que se deben de cargar por defecto si no es posible saber el idioma de preferencia del usuario.

Ejemplo:

```
<video autoplay="autoplay" controls="controls" poster="foto1.jpg"
>
  <source src="video.mp4" type="video/mp4" >
  <source src="video.ogv" type="video/ogg" >
  <track src="spa.vtt" kind="subtitles" srclang="es"
    label="Español" >
  <track src="deu.vtt" kind="subtitles" srclang="de"
    label="Alemán" >
  El navegador no puede mostrar el vídeo
</video>
```

Audio en las páginas web

Insertar audio en una páginas web es muy similar a insertar vídeo. El elemento que lo hace posible, se llama precisamente audio. Sus atributos (que funcionan igual que con el elemento `video`) son:

atributo	valor
<code>src</code>	URL a la fuente de audio que estamos insertando
<code>autoplay</code>	El audio se reproducirá inmediatamente no se esperará a que el usuario pulse el botón Play

atributo	valor
<code>loop</code>	El audio se repetirá continuamente
<code>controls</code>	Se muestran los controles de ese audio (<code>play</code> , <code>rewind</code> , <code>pause</code> , etc.)
<code>preload</code>	Recomendación sobre cómo debemos realizar la descarga. Posibilidades: <code>auto</code> . El audio se descarga en cuanto se carga la página <code>none</code> . El audio no se descarga. Cuando el usuario pulse play, se descargará. <code>metadata</code> . Descarga los metadatos del audio, no el audio en sí

Con el audio hay el mismo problema con la cuestión de los codecs y los formatos, por lo que también es habitual convertir el audio a distintos formatos y dar opciones dentro de la etiqueta audio gracias a la etiqueta `source`. Ejemplo:

```
<audio controls="controls" autoplay>
  <source src="audio1.ogg" type="audio/ogg">
  <source src="audio2.mp3" type="audio/mpeg">
  No se puede reproducir el archivo de audio
</audio>
```

Elemento canvas

En inglés `canvas` significa *lienzo*, y define muy bien para que sirve este elemento creado en la norma HTML5. Es uno de los componentes de HTML5 más famosos por el gran aporte que ha supuesto al dinamismo de las páginas web.

Mediante este elemento dispondremos de un área que podremos utilizar donde queramos para dibujar elementos gráficos mediante las instrucciones gráficas del lenguaje JavaScript. Eso ha permitido (gracias a la potencia de JavaScript) crear juegos, animaciones y elementos visuales atractivos en las páginas web.

Atributos de canvas

`id`. Es el atributo que utilizan todos los elementos HTML para identificarlos. En el caso de canvas es casi obligatorio su uso para poder hacer referencia al mismo desde el lenguaje JavaScript.

`width`. Anchura del lienzo (funciona igual que en el caso de `img`)

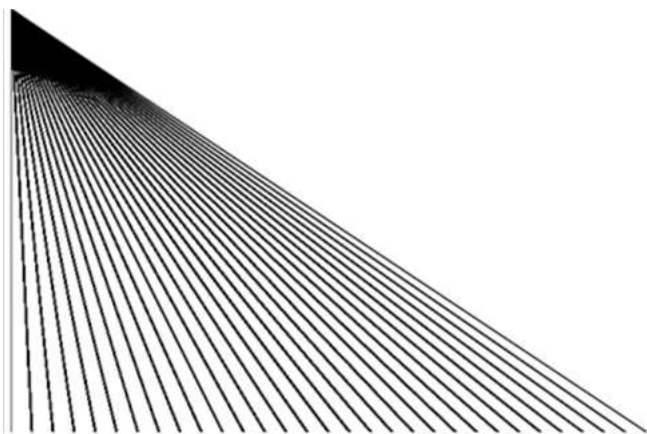
`height`. Altura del lienzo.

Ejemplo de uso de `canvas` (posee código JavaScript):

```
<canvas id="lienzo1" width="600" height="400">
<script type="text/javascript">
  var canvas=document.getElementById("lienzo1");
  var contexto=canvas.getContext("2d");
  contexto.lineWidth=2;
  for (i=0;i<=600;i+=20) {
    contexto.moveTo(0,0);
    contexto.lineTo(i,400)
    contexto.stroke();
  }
</script>
```

Mediante la etiqueta `script` podemos colocar código en lenguaje JavaScript, desde ese código podemos utilizar el lienzo para dibujar.

El resultado del código anterior es la imagen:



En cualquier caso es un elemento que no está reconocido en muchos navegadores (aunque sí en todos los modernos).

Inserción directa de imágenes SVG

En el tema anterior hemos visto que es posible insertar imágenes SVG (imágenes de tipo vectorial) a través del elemento `img` al igual que las imágenes JPEG o PNG por ejemplo. Sin embargo las imágenes SVG están construidas en un dialecto de XML que cualquier navegador de hoy en día reconoce.

Es decir, podemos insertar el código SVG directamente en una página web. Ejemplo:

```
<svg xmlns="http://www.w3.org/2000/svg">
  ..etiquetas svg
</svg>
```

Dentro del elemento `svg` se colocan las etiquetas que permiten dibujar en el área del `svg`. En principio si no indicamos tamaño alguno, todo el área de la página web se utiliza para dibujar en SVG, pero podemos indicar atributos `width` y `height` para indicar tamaños concretos.

Las etiquetas que se pueden utilizar dentro de `svg` pertenecen al lenguaje estándar SVG y rebasan el propósito de este manual, pero un ejemplo de ellas sería:

```
<h1>Dibujo SVG</h1>

<svg xmlns="http://www.w3c.org/2000/svg" height="400">
  <rect id="rec1" x="50" y="50" width="300"
    height="100" fill="red" >
  <ellipse id="elips1" cx="200" cy="100" rx="50"
    ry="75" fill="blue" >
</svg>
```

Ecuaciones matemáticas

El lenguaje `MathML` es un clásico de los lenguajes derivados de XML. Su aplicación es mostrar ecuaciones matemáticas al estilo del lenguaje `LaTeX`, consiguiendo la tipografía y forma visual

apropiada para las fórmulas. Aunque se considera parte de HTML 5, solo Safari y Firefox admiten este lenguaje.

Ejemplo de código MathML:

```
<math>
  <msub><mi>x</mi><mtext>0</mtext></msub>
  <mo>=</mo>
  <mn>2</mn>
  <msup>
    <mfenced open="(" close=")">
      <mfrac>
        <msup>
          <mi>y</mi><mn>2</mn>
        </msup>
        <msup>
          <mi>z</mi><mn>3</mn>
        </msup>
      </mfrac>
    </mfenced>
    <mn>2</mn>
  </msup>
</math>
```

$$x_0 = 2 \left(\frac{y^2}{z^3} \right)^2$$

Obtiene el resultado:

Otros elementos multimedia

Nuevamente en HTML 5 ha habido una preocupación sobre cómo utilizar elementos multimedia en una página web. Hasta la aceptación de HTML 5 ha sido problemática la forma de añadir elementos multimedia.

El uso de [Flash](#) simplificó esta posibilidad dada la potencia de esta tecnología, pero dificultaba el aprendizaje para realizar páginas y exigía utilizar una forma de trabajar ajena a HTML y que además dependía de una empresa en concreto (**Adobe**). Actualmente la idea de muchos creadores es ir retirando Flash y aprovechar las nuevas capacidades de HTML 5 para la multimedia.

En especial HTML 5 avanza enormemente en cuestiones relacionadas con el vídeo y gracias al elemento [canvas](#), gran parte de lo que sólo parecía poder hacerse mediante Flash, es posible hacerlo ahora sin salirse del estándar HTML.

El problema del vídeo y el audio

El problema es el mismo que con las imágenes, es que hay numerosos formatos de vídeo y audio y que cada navegador tiene capacidades distintas al respecto. Es decir, cada navegador es capaz de reproducir directamente sólo un conjunto pequeño de tipos de vídeo.

Por ello muchos usuarios conocedores de estos problemas saben que en los navegadores hay que instalar plugins (extensiones) para poder ver vídeos u oír música. No es un problema sólo de los navegadores, casi es el problema fundamental del mundo de la imagen, el vídeo y el audio.

De esta forma si incorporamos vídeo a nuestra página a través de un enlace normal:

```
<a href="video1.mpeg">Ver vídeo</a>
```

El navegador buscará si disponemos del plugin apropiado para ver el vídeo, si no es así simplemente lo descargará en nuestro ordenador al no poder mostrar él el contenido.

Etiqueta embed

Se trata de una etiqueta ya veterana que se utiliza para colocar en una página web elementos no pertenecientes al lenguaje HTML de cualquier tipo: como animaciones Flash, vídeo, audio, etc.

Usa los siguientes atributos:

atributos	uso
src	URL al recurso que se desea mostrar
type	Tipo MIME (según la norma IANA de tipos de medios) que indica el tipo de recurso al que se refiere el elemento embed.
height	Altura de la ventana que mostrará el recurso
width	Anchura de la ventana que mostrará el recurso

Etiqueta object

Esta etiqueta está orientada a sustituir a la anterior y permite incorporar cualquier tipo de contenido a una página web. Tiene más posibilidades que la anterior. Los atributos posibles son:

atributo	uso
data	URL al recurso que se desea mostrar
type	Tipo MIME que indica el contenido del recurso que se incorpora con la etiqueta.
height	Altura de la ventana que mostrará el recurso
width	Anchura de la ventana que mostrará el recurso
usemap	Permite indicar el nombre de un mapa de imágenes (usando #nombre) que actuará sobre el objeto. El mapa de imágenes funcionará igual que con la etiqueta img (según lo comentado en el apartado dedicado a los mapas de imágenes)
name	Permite indicar un nombre para el objeto
form	Indica el nombre del formulario al que pertenece este objeto

Elemento param

Elemento interior a **object** (no tiene sentido sin él), que permite especificar parámetros al objeto incrustado. A través de **param** pasamos instrucciones al plugin que reproduce el objeto a fin de que lo haga según nuestras necesidades.

La forma de indicar los parámetros es mediante los atributos **name** (en el que se indica el nombre del atributo) y **value** (valor del atributo)

Ejemplo:

```
<object data="Wildlife.wmv" type="video/x-ms-wmv" width="500"
height="300">
  <param name="autoplay" value="true" />
</object>
```

En el ejemplo el vídeo (en caso de que el navegador pueda reproducirle), se ejecutará automáticamente al cargar la página gracias a que se ha indicado el parámetro `autoplay` con valor `true`.

Elemento iframe

Es un elemento que había desaparecido en el estándar pero que HTML 5 ha recuperado. La idea original es colocar un documento dentro de otro documento, es decir sirve para incrustar contenido de una dirección dentro de la página que le hace referencia. Su popularidad actual se debe a [youtube](#) y otras páginas que nos sugieren el uso de esta etiqueta para incrustar contenido procedente de ellas; por lo que es una etiqueta de uso muy habitual para incrustar contenido multimedia (aunque `iframe` significa marco interior e inicialmente estaba pensado para los obsoletos marcos).

Atributos:

atributo	uso
<code>src</code>	URL al recurso que se desea mostrar
<code>width</code>	Anchura del objeto en nuestra página
<code>height</code>	Altura del objeto en nuestra página
<code>sandbox</code>	<p>Permite indicar restricciones al contenido incorporado por <code>iframe</code>. Los posibles valores son:</p> <p><code>allow-same-origin</code>. Hace que el contenido sea tratado como si tuviera el mismo origen que la página que contiene al elemento <code>iframe</code></p> <p><code>allow-top-navigation</code>. Permite que el marco pueda navegar por la página web.</p> <p><code>allow-forms</code>. Permite formularios en el contenido</p> <p><code>allow-scripts</code>. Permite la ejecución de scripts dentro del elemento <code>iframe</code>.</p> <p>Sin indicar nada (""), se aplican todas las restricciones anteriores.</p>
<code>seamless</code>	No se utiliza porque ningún navegador le reconoce todavía. Hace que el <code>iframe</code> sea tratado como si fuera parte normal de la página web (no se mostrarán bordes, indicaciones del destino, barras de desplazamiento,...)

