HTML y CSS

Este manual va a describir el lenguaje de marcas HTML para desarrollo web de contenido estático junto con el lenguaje para formateado CSS.

Tabla de contenido

1 Introducción a los documentos web....1
1.1 Lenguajes de marcas para la web....1
1.1.1 HTML....2
1.1.2 XHTML....3
1.2 Evolución histórica de HTML....3
1.3 Diferencias entre HTML y XHTML....4
1.4 La definición de tipo de documento en XHTML....5
1.4.1 XHTML 1.0 Estricto....6
1.4.2 XHTML 1.0 Transitorio....6
1.4.3 XHTML 1.0 Frameset....6
1.5 ¿Qué se necesita para crear documentos web?....6
1.5.1 Editores web....7
1.5.2 Navegadores web....7
1.5.3 Servidor web....8
1.6 Instalación de las herramientas necesarias....8
1.6.1 Internet Information Services....8
1.6.2 Komodo Edit....9
1.6.3 Navegadores....11
1.7 Navegadores que soportan HTML 5....11
1.8 Mi primera página web....12

2 Sintaxis HTML....14
2.1 Componentes de un documento web....14
2.1.1 Elementos....14
2.1.2 Atributos....15
2.1.3 Atributos globales....16
2.1.4 Comentarios....17

3 Estructura de un documento HTML....17
3.1 Tipo de documento y elemento raíz....17
3.2 Cabecera del documento....18
3.2.1 Título del documento....18
3.2.2 Elemento meta....18
3.3 Cuerpo del documento....19
3.3.1 Cabecera del cuerpo....23
3.3.2 Barra de navegación....24
3.3.3 Información principal....24
3.3.4 Barra lateral....25
3.3.5 Pie del documento....26
3.3.6 Formato del diseño....28

4 Elementos textuales....28
4.1 Párrafos....29

4.2 Saltos de línea....29
4.3 Texto preformateado....30
4.4 Encabezados....31
4.5 Separadores....32
4.6 Elementos textuales en línea....32
4.6.1 Citas textuales....33
4.6.2 Texto entrecomillado....33
4.6.3 Acrónimos....33
4.6.4 Superíndices y subíndices....34
4.6.5 Texto resaltado....34
4.6.6 Elemento span....35
4.6.7 Elementos para estilos....35
4.7 Referencias de caracteres con nombre....35
5 Introducción a CSS....36
5.1 ¿Qué puedo hacer con CSS?....36
5.2 ¿Qué diferencias hay entre HTML y CSS?....37
5.3 ¿Qué beneficios me ofrece CSS?....37
5.4 Características CSS....37
6 Aplicando CSS a un documento HTML....38
6.1 Estilo en línea....39
6.2 Estilo por ámbito....39
6.3 Hoja de estilo interna....40
6.4 Hoja de estilo externa....41
7 Las reglas de estilo....42
7.1 El selector....43
7.2 Propiedades y valores....43
7.2.1 Unidades de medida....44
7.2.2 Valores de color....44
7.2.3 URL's....47
7.3 Herencia....49
7.4 Introducir comentarios....50
7.5 Las nuevas propiedades de la version 3....50
8 Colores y fondos con CSS....52
8.1 Color en primer plano....52
8.2 Color de fondo....52
8.3 Imagen de fondo....53
8.3.1 Propiedad abreviada background....56
8.4 Gradientes como fondo....57
9 Formato de texto con CSS....61
9.1 Fuentes....61
9.1.1 Especificar fuentes en los elementos....62
9.1.2 Propiedad abreviada font....64

9.1.3 Fuentes en el servidor....64
9.1.4 ¿Donde puedo conseguir nuevas fuentes?....66
9.2 Propiedades para elementos textuales....66
10 Imágenes en documentos web....69
10.1 Gráficos vectoriales....69
10.2 Mapas de bits....69
10.2.1 Resolución....70
10.2.2 Profundidad de color....70
10.2.3 Dimensión de la imagen....71
10.2.4 El tamaño de archivo de imagen....71
10.3 Formato de archivo de imagen....72
10.4 Incluir una imagen en la web....73
10.5 Formato de imagen....74
10.5.1 Tamaño de la imagen....74
10.5.2 Alineación....74
10.5.3 Elementos figure y figcaption....75
11 Enlaces....77
11.1 Incluir enlaces en un documento web....78
11.2 Destino del enlace....78
11.3 Marcadores....79
11.4 Enlaces a otros recursos....80
11.5 Formato del enlace....80
12 Tablas....83
12.1 Estructura de una tabla en HTML....83
12.2 Celdas de cabecera y celdas de datos....84
12.3 Título de tabla....85
12.4 Expansión de celdas....86
12.5 Grupos de filas....89
12.6 Grupos de columnas....90
12.7 Calcular el número de columnas de una tabla....91
13 Formato de tabla con CSS....92
13.1 Alto y ancho de tabla y celdas....92
13.2 Bordes....95
13.3 Alineación....98
13.4 Relleno....99
13.5 Propiedades específicas para elemento table....100
13.5.1 La ubicación del título....100
13.5.2 Formato de las tablas....100
13.5.3 Espacio entre celdas....101
13.5.4 Modelo de los bordes....101
13.5.5 Comportamiento de las celdas vacias....102
14 Maquetación web....103

14.1 Clases e identificadores....103
14.1.1 Clases....103
14.1.2 Identificadores....105
14.2 Agrupación de elementos....107
14.2.1 Elemento SPAN....107
14.2.2 Elemento DIV....108
14.3 Capas....109
14.4 El modelo de caja....110
14.4.1 Margen....112
14.4.2 Relleno....113
14.4.3 Bordes....113
14.4.3.1 Bordes con esquinas redondeados....113
14.4.3.2 Sombra....114
14.4.3.3 Borde con imágenes de fondo....115
14.4.3.4 Contorno....117
14.4.4 Dimesión....117
14.4.5 Desborde....118
14.5 Elementos flotantes....119
14.6 Visibilidad....121
14.7 Opacidad....122
14.8 Posicionamiento....122
14.8.1 Posicionamiento estático....123
14.8.2 Posicionamiento absoluto....123
14.8.3 Posicionamiento relativo....124
14.8.4 Posicionamiento fijo....125
14.9 Apilamiento....126
15 Listas....128
15.1 Listas sin numerar....128
15.2 Listas ordenadas....129
15.3 Listas de definiciones....130
15.4 Lista multinivel....130
15.5 Formato de listas....131
15.5.1 Estilo de la viñeta....132
15.5.2 Imágenes como viñetas....132
15.5.3 Posición de la viñeta o numeración....133
15.5.4 Listas multinivel....134
15.5.5 Contadores....135
16 Formularios....138
16.1 Tipos de controles....138
16.2 Estructura de un formulario web....140
16.2.1 Elemento fieldset....142
16.3 Controles de formulario....142
16.3.1 Etiquetas de control....144
16.3.2 Elemento input....145
16.3.3 Cuadros de texto....147

16.3.4 Texto de búsqueda....147
16.3.5 Teléfono....148
16.3.6 URL....148
16.3.7 Email....148
16.3.8 Cuadro de contraseña....148
16.3.9 Fecha y hora....149
16.3.10 Número....149
16.3.11 Rango....150
16.3.12 Color....150
16.3.13 Casilla de verificación....151
16.3.14 Grupo de botones de radio....151
16.3.15 Fichero....152
16.3.16 Imagen....152
16.3.17 Área de texto....153
16.3.18 Lista desplegable....153
16.3.18.1 Opciones de la lista....154
16.3.18.2 Agrupar opciones....155
16.3.19 Envío de formulario....156
16.3.20 Reinicio del formulario....157
16.3.21 Campo oculto....157
16.3.22 Elemento button....157
16.3.23 Lista de opciones. Elemento datalist....158
17 Formato del formulario....159
17.1 Mejoras en los campos de texto....159
17.2 Etiquetas alineadas y formateadas....159
17.3 Formulario de varias columnas....162
17.4 Resaltar un campo seleccionado....163
17.5 Botones con iconos....163
18 Elementos multimedia....165
18.1 Multimedia antes de HTML 5....165
18.1.1 Elemento embed....166
18.1.2 El elemento object....167
18.2 Vídeo en HTML5....169
18.2.1 Formatos de vídeos y soporte del navegador....169
18.2.2 Introducción de video en HTML5....170
18.3 Audio en HTML5....171
18.3.1 Formatos de audio y soporte del navegador....171
18.3.2 Introducción de audio en HTML5....172
18.3.3 Incluir un video de Youtube....172
19 Mapas....173
19.1 Mapas en modo cliente. Elementos <map> y <area>....174
19.2 Crear un mapa....175
20 Transiciones, transformaciones y animaciones....176
20.1 Transformaciones....177
20.1.1 Mover un elemento....177

20.1.2 Rotar un elemento....178
20.1.3 Escalar un elemento....178
20.1.4 Giro en dos ejes....179
20.1.5 Cambiándolo todo....179
20.2 Transiciones....180
20.3 Animaciones....184
20.3.1 La regla @keyframes....184
21 Marcos....187
21.1 Disposición de los marcos....188
21.1.1 Elemento frameset....189
21.1.2 Contenido de un marco. Elemento frame....190
21.2 Marcos anidados....193
21.3 Contenido alternativo....195
21.4 Marcos en línea....196
22 Publicación web....197
22.1 Construcción de un sitio web....198
22.1.1 Análisis del sitio web....199
22.1.1.1 Elegir un tema....199
22.1.1.2 Definición de los objetivos....199
22.1.1.3 Definición del destinatario....199
22.1.1.4 Análisis del contenido....200
22.1.1.5 Evaluación de la competencia....200
22.1.2 Diseño....200
22.1.3 Desarrollo....201
22.1.3.1 Nomenclatura de archivos....201
22.1.3.2 Organización del almacenamiento....202
22.1.4 Pruebas....202
22.1.4.1 Validar un sitio web....203
22.2 Dominio y hospedaje....204
22.2.1 Selección de un servidor....204
22.2.1.1 Cómo elegir un alojamiento gratuito....204
22.2.1.2 Cómo elegir un alojamiento de pago....206
22.2.1.3 Búsqueda de los mejores hostings....208
22.2.1.4 Alojamiento en un servidor propio....208
22.3 Normativa....209
22.3.1 Prácticas poco éticas....210
22.3.2 Creative Commons....210
22.3.2.1 Licencias Creative Commons....210
22.3.2.2 Publicar una obra bajo licencia Creative Commons....212
23 Bibliografía....215

HTML y CSS

1 Introducción a los documentos web

Un documento web es un conjunto de información electrónica compuesto por texto, sonido, vídeo, programas, enlaces, imágenes, hipervínculos, ..., almacenada en un archivo y adaptada para el servicio World Wide Web (WWW); y que puede ser accedida mediante un navegador web. Esta información se encuentra escrita mediante un lenguaje de marcas, generalmente HTML, y puede proporcionar acceso a otras páginas web mediante enlaces de hipertexto. Frecuentemente también incluyen otros recursos como pueden ser hojas de estilo en cascada, scripts, imágenes digitales, entre otros.

1.1 Lenguajes de marcas para la web

Los lenguajes de marcado para crear documentos web permiten aglutinar textos, sonidos e imágenes y combinarlos a nuestro gusto. Además, y es aquí donde reside su mayor ventaja, la estructura de los documentos es hipertexto lo que nos permite la introducción de referencias a otros documentos web por medio de los enlaces hipertexto.

El lenguaje de marcado para la web fue HTML, el cual se creó en un principio con objetivos divulgativos. No se pensó que la web llegara a ser un área de ocio con carácter multimedia, de modo que, el HTML se creó sin dar respuesta a todos los posibles usos que se le iba a dar y a todos los colectivos de personas que lo utilizarían en un futuro. Sin embargo, pese a esta deficiente planificación, si que se han ido incorporando modificaciones con el tiempo, a través de diferentes versiones normalizados.

Uno de de los problemas que han acompañado al HTML es la diversidad de navegadores presentes en el mercado los cuales no son capaces de interpretar un mismo código de una manera unificada. Esto obliga al desarrollador web a, una vez creada su página, comprobar que esta puede ser leída satisfactoriamente por todos los navegadores, o al menos, los más utilizados.

A lo largo del tiempo han surgido dos lenguajes de marcado íntimamente relacionados como consecuencia del proceso de desarrollo y estandarización del W3C que ha seguido. Estos son:

- ✔ HTML ➜ (*Hipertext Markup Language*) es el lenguaje de marcado derivado de SGML con el que se escriben los documentos electrónicos popularmente conocidos como páginas web.
- ✔ XHTML ➜ (*eXtensible HyperText Markup Language*) es, básicamente, HTML expresado como XML válido, del cual deriva. Es más estricto a nivel técnico, pero esto permite que posteriormente sea más fácil al hacer cambios o buscar errores entre otros.

Vamos a ver ambos lenguajes con algo más de detalles

1.1.1 HTML

Desde la publicación de HTML 4.01, la actividad de estandarización de HTML se detuvo y el W3C se centró en el desarrollo del estándar XHTML. Por este motivo, en el año 2004, las empresas Apple, Mozilla y Opera mostraron su preocupación por la falta de interés del W3C en HTML y decidieron organizarse en una nueva asociación llamada WHATWG (*Web Hypertext Application Technology Working Group*).

La actividad actual del WHATWG se centra en el estándar HTML5, cuyo primer borrador oficial se publicó el 22 de enero de 2008. Debido a la fuerza de las empresas que forman el grupo WHATWG y a la publicación de los borradores de HTML 5, en marzo de 2007 el W3C decidió unirse al WHATWG y retomar el desarrollo y estandarización de HTML.

El HTML5 es la quinta revisión del lenguaje. Esta nueva versión pretende corregir problemas con los que los desarrolladores web se encuentran, así como rediseñar el código actualizandolo a nuevas necesidades que demanda la web de hoy en día.

Cuando se publicó el borrador, la mayoría de los navegadores no contemplaban esta nueva versión y la fueron incorporando poco a poco, lo que permitió a desarrolladores ir haciendo sitios web con esta versión del lenguaje. Actualmente va por la versión 5.2 cuyo último borrador se publicó en diciembre de 2017

HTML5 introduce además nuevas APIs (*Application Programming Interface*) y se puede definir como un conjunto de subrutinas, especificaciones y métodos que ofrece cierta biblioteca para ser utilizado por otro programa. Las APIs permiten hacer uso de funciones o infraestructura ya existente en otro programa, ahorrando tiempo y recursos al evitar programar todo desde el inicio.

Las APIs para HTML5 están escritas en JavaScript y permiten realizar una serie de tareas que a menudo son más utilizadas y útiles en dispositivos móviles. Estas son:

- ✔ Drag and Drop ➜ Esta es una característica común que muchas interfaces de servicios y plataformas poseen. Consiste en arrastrar un objeto o elemento y moverlo a una nueva ubicación. Puede realizarse en HTML mediante eventos.

- ✔ Geolocalización ➜ Función para averiguar la posición geográfica del usuario, una alternativa a otros métodos más exactos GPS o inexactos como captura de dirección IP. La API sólo devuelve las coordenadas del usuario. Aunque la geolocalización puede ser implementada por los desarrolladores, su utilización debe ser aprobado por los usuarios ya que viola normas de privacidad.
- ✔ Almacenamiento local ➜ Al implementarlo, las aplicaciones web son capaces de almacenar datos de forma local dentro del navegador del usuario. Es una opción más segura, con mayor capacidad de almacenamiento que las usuales cookies en donde la información no se transfiere al servidor.
- ✔ App Cache ➜ Función para acceder a una aplicación web sin necesidad de conectarse a internet. Como su nombre indica se hace uso de la copia caché de dicha aplicación para poder acceder a ella la próxima vez. Esta característica no sólo permite acceder a aplicaciones web sin conexión a internet, sino que mejora la velocidad al reunir recursos almacenados en caché.
- ✔ WebWorkers ➜ Hace referencia a un código JavaScript que corre en segundo plano, sin afectar el rendimiento de la página. De forma usual, cuando un script se ejecuta en una página HTML, la página no responde hasta que el script ha sido procesado por completo. Con el uso de los WebWorkers, el script se sigue ejecutando, pero no se pierde la capacidad de interacción con otros elementos de la página.

1.1.2 XHTML

De forma paralela a su actividad con HTML, W3C ha continuado con la estandarización de XHTML, una versión avanzada de HTML y basada en XML. La primera versión de XHTML se denomina XHTML 1.0 y se publicó el 26 de Enero de 2000 (y posteriormente se revisó el 1 de Agosto de 2002).

XHTML 1.0 es una adaptación de HTML 4.01 al lenguaje XML, por lo que mantiene casi todas sus etiquetas y características, pero añade algunas restricciones y elementos propios de XML. La versión XHTML 1.1 ya ha sido publicada en forma de borrador y pretende modularizar XHTML. También ha sido publicado el borrador de XHTML 2.0, que iba a suponer un cambio muy importante respecto de las anteriores versiones de XHTML. Sin embargo, esta versión no llegó a publicarse formalmente ya que se abandonó el proyecto en favor de HTML5.

1.2 Evolución histórica de HTML

El origen de HTML se remonta a 1980, cuando el físico Tim Berners-Lee, trabajador del CERN (Centro Europeo para la Investigación Nuclear) propuso un nuevo sistema de "hipertexto" para compartir documentos a partir del lenguaje SGML.

Los sistemas de "hipertexto" habían sido desarrollados años antes. En el ámbito de la informática, el "hipertexto" permitía que los usuarios accedieran a la información relacionada con los documentos electrónicos que estaban visualizando. De cierta manera, los primitivos sistemas de "hipertexto" podrían asimilarse a los enlaces de las páginas web actuales.

Tras finalizar el desarrollo de su sistema de "hipertexto", Tim Berners-Lee lo presentó a una convocatoria organizada para desarrollar un sistema de "hipertexto" para Internet. Después de unir sus fuerzas con el ingeniero de sistemas Robert Cailliau, presentaron la propuesta ganadora llamada World Wide Web (W3).

El primer documento formal con la descripción de HTML se publicó en 1991 bajo el nombre "HTML Tags" (Etiquetas HTML) y todavía hoy puede ser consultado online a modo de reliquia informática.

La primera propuesta oficial para convertir HTML en un estándar se realizó en 1993 por parte del organismo IETF (*Internet Engineering Task Force*). Aunque se consiguieron avances significativos (en esta época se definieron las etiquetas para imágenes, tablas y formularios) ninguna de las dos propuestas de estándar, llamadas HTML y HTML+ consiguieron convertirse en estándar oficial.

En 1995, el organismo IETF organiza un grupo de trabajo de HTML y consigue publicar, el 22 de septiembre de ese mismo año, el estándar HTML 2.0. A pesar de su nombre, HTML 2.0 es el primer estándar oficial de HTML.

A partir de 1996, los estándares de HTML los publica otro organismo de estandarización: el W3C (*World Wide Web Consortium*). La versión HTML 3.2 se publicó el 14 de Enero de 1997 y es la primera recomendación de HTML publicada por el W3C. Esta revisión incorpora los últimos avances de las páginas web desarrolladas hasta 1996, como applets de Java y texto que fluye alrededor de las imágenes.

HTML 4.0 se publicó el 24 de Abril de 1998 (siendo una versión corregida de la publicación original del 18 de Diciembre de 1997) y supone un gran salto desde las versiones anteriores. Entre sus novedades más destacadas se encuentran las hojas de estilos CSS, la posibilidad de incluir pequeños programas o scripts en las páginas web, mejora de la accesibilidad de las páginas diseñadas, tablas complejas y mejoras en los formularios.

La siguiente especificación oficial de HTML se publicó el 24 de diciembre de 1999 y se denomina HTML 4.01. Se trata de una revisión y actualización de la versión HTML 4.0, por lo que no incluye novedades significativas.

El gran cambio vendría con la versión 5, cuyo estándar se publico en octubre de 2014. HTML 5 especifica dos variantes de sintaxis para HTML: una «clásica», HTML (`text/html`), conocida como HTML5, y una variante XHTML conocida como sintaxis XHTML 5 que deberá servirse con sintaxis XML (`application/xhtml+xml`).

1.3 Diferencias entre HTML y XHTML

En el momento de escribir este documento, podemos decir que XHTML es un proyecto abandonado por el W3C. Solamente publicó las versiones 1.0 y 1.1 para centrarse en el desarrollo de HTML5. Es recomendable para los desarrolladores web emplear este último cuyo último borrador (versión 5.2) se publico en diciembre del 2017. Las principales diferencias de XHTML y HTML son las siguientes:

- ✔ HTML deriva de SGML y XHTML deriva de XML.

- ✔ HTML es menos estricto, si hay un error de sintaxis el navegador todavía puede mostrar la página web, mientras que en XHTML hay que seguir estrictamente las reglas de sintaxis para que el navegador la muestre.
- ✔ HTML está diseñado básicamente para desplegar datos y está centrado en la presentación de esos datos mientras que XHTML está básicamente enfocado a la distribución de datos.

Respecto a la sintaxis del lenguaje empleado en un documento web, cuando un documento se transmite con un tipo mime XML (`application/xhtml+xml`) es tratado como un documento XML por los navegadores para ser traducido por el procesador XML. Por tanto el más pequeño error de sintaxis podría provocar que el documento no se visualizara correctamente. Sin embargo, si se transmite con tipo mime HTML (`text/html`) entonces pequeños errores de sintaxis pueden pasar por alto y el navegador mostrar el documento correctamente.

El W3C permite escribir los documentos con la sintaxis HTML o XHTML, sin embargo no se especifican requisitos nivel de sintaxis más allá de los propios de XML. Por tanto no se define una especificación formal en un DTD. Por tanto no se requiere la inclusión de una declaración de documento `<!DOCTYPE>` con un DTD. La sintaxis XML consiste en:

- ✔ Cierre de elementos ➜ Cada elemento debe cerrarse con su correspondiente etiqueta. Si es un elemento vacío, entonces se indicará una barra antes del símbolo mayor indicando que el elemento se abre y cierra en el mismo punto. Por ejemplo, `<br/>`.
- ✔ Anidamiento de elementos ➜ Los elementos anidados se cierran en orden inverso a su apertura.
- ✔ Solo puede haber un elemento raíz.
- ✔ Sensible a mayúsculas y minúsculas ➜ Las etiquetas de los elementos y sus atributos deben escribirse en minúsculas.
- ✔ Los valores de los atributos siempre van entrecomillados.
- ✔ Los atributos siempre tienen valor ➜ Si el atributo no tiene valor se pondrá como valor el propio atributo.
- ✔ No se permiten caracteres especiales, tienen que sustituirse por su correspondiente entidad.
- ✔ Se prescinde del atributo `name` para identificar los elementos de las páginas. Se emplea en exclusiva el atributo `id`.

1.4 La definición de tipo de documento en XHTML

El estándar XHTML deriva de XML, por lo que comparte con el muchas de sus normas y sintaxis. Uno de los conceptos fundamentales de XML es la utilización de la definición del tipo de documento o DTD (*Document Type Definition*).

Un DTD es un documento que recoge el conjunto de normas, obligaciones y restricciones que deben cumplir los documentos de un determinado tipo. Si por ejemplo se define un DTD para los documentos relacionados con libros, se puede fijar como norma que cada libro tenga un título y sólo uno, que tenga uno o más autores, que la información sobre el número de páginas pueda ser opcional, etc.

El estándar XHTML define el DTD que deben seguir las páginas y documentos XHTML. En este documento se definen las etiquetas que se pueden utilizar, los atributos de cada etiqueta y el tipo de valores que puede tener cada atributo.

En realidad, la versión 1.0 del estándar de XHTML define tres DTD diferentes. Para indicar el DTD utilizado al crear una determinada página, se emplea la declaración de documento con `<!DOCTYPE>`

Para que una página XHTML sea correcta y válida es imprescindible que incluya el correspondiente declaración de documento que indica el DTD utilizado. A continuación se muestran los tres DTD que se pueden utilizar al crear páginas XHTML:

1.4.1 XHTML 1.0 Estricto

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Se trata de la variante con las normas más estrictas y las restricciones más severas. Las páginas web que incluyan esta declaración, no pueden utilizar atributos relacionados con el aspecto de los contenidos, por lo que requiere una separación total de código HTML y estilos CSS.

1.4.2 XHTML 1.0 Transitorio

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Se trata de una variante menos estricta que la anterior, ya que permite el uso de algunos atributos HTML relacionados con el aspecto de los elementos.

1.4.3 XHTML 1.0 Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Esta última variante la utilizan las páginas que están formadas por marcos, una práctica completamente desaconsejada y que hoy en día sólo utilizan los sitios web obsoletos.

1.5 ¿Qué se necesita para crear documentos web?

Desarrollar un sitio web no es una tarea trivial, pero afortunadamente se simplifica con el uso de una serie de herramientas, la mayoría de las cuales son de uso libre. Recordemos que las páginas web son archivos de texto escritas en un lenguaje de etiquetado web. En principio, para crear los documentos electrónicos que formaran un sitio web se necesita lo

siguiente:

1. Editor web para escribir las páginas web.
2. Navegador web para probar y visualizar el resultado de la página que hemos hecho.
3. Servidor web para almacenar y gestionar la página web.

1.5.1 Editores web

Los editores de texto para desarrollo web HTML pueden ser de tres tipos tipos: Editor simple de texto, Editor HTML y Editor WYSIWYG. Un editor simple de texto, como el *Bloc de notas* de Windows o *Edit* en Linux, son editores para crear archivos de texto de propósito general. Por tanto, también se pueden emplear para crear páginas web, aunque su simplicidad es tal que no incorpora ningún tipo de ayuda o funcionalidad para el desarrollo web. Así que es mejor optar por alguno de los siguientes.

El editor HTML es un editor de texto plano específicamente diseñado para la edición web, los cuales incluyen una serie de características como la revisión de sintaxis en tiempo real, asistente para escritura correcta del código de acuerdo a la sintaxis del lenguaje, inserción automática de etiquetas o la coloración inteligente, que facilitan tanto la escritura como la depuración de programas. Ejemplo de este tipo de editores tenemos *Apstana Studio* y *Komodo Edit*.

Un editor WYSIWYG es el acrónimo de *What You See Is What You Get*, que quiere decir: "*lo que ves es lo que obtienes*", en los que de manera visual se pueden colocar distintos elementos sobre una vista previa de la página, encargándose el programa de generar el documento web en lenguaje HTML. Este tipo de editores nos permiten crear la página como si estuviéramos escribiendo un documento con un editor del tipo de Word o Writer. La aplicación es la encargada posteriormente de traducir lo que el desarrollador escribe y programar internamente la página con el código HTML. Ejemplo de editor WYSIWYG tenemos *FrontPage* y *Dreamweaver*.

1.5.2 Navegadores web

En cuanto a los navegadores web tenemos varias opciones donde elegir, aunque se recomienda elegirlos todos. Desgraciadamente y a pesar de la estandarización de muchas de las tecnologías en la red, el desarrollo de los navegadores ha estado condicionado por la competencia entre las compañías de software, las cuales se han ido más allá de los estándares impuestos por los organismos de estandarización. Como resultado de ello, los navegadores han implementado funcionalidad propia de las compañías que lo han desarrollado y que es incompatible con los navegadores de otras compañías.

La *guerra de las navegadores* ha supuesto una pesadilla para los desarrolladores web, los cuales han sido los daños colaterales. Un desarrollador escribe una página web y al probarla ve como se visualiza perfectamente en un navegador, sin embargo, si visualiza esa misma página en otro aparece como un churro. Por tanto, el desarrollo web obliga a probar nuestras páginas web en todos los navegadores posibles para garantizar que se visualizarán tal y como esperamos de la misma forma en todos ellos. Así, nos aseguramos que un usuario

pueda visualizar nuestra página web independientemente del navegador que utilice.

Los navegadores más populares del mercado son: Google Chrome, Microsoft Edge, Mozilla Firefox, Safari y Opera.

1.5.3 Servidor web

En principio, no es absolutamente necesario emplear un servidor web para el desarrollo web. Los navegadores pueden visualizar las páginas web almacenadas localmente en nuestro disco duro. Sin embargo, si tenemos pensado publicar nuestro sitio web, es recomendable instalar un servidor web en nuestro ordenador y colocar las páginas web en él para simular su funcionamiento en un entorno de servidor web en el que posteriormente se alojará. Cuando desarrollamos un sitio web de forma local, podemos encontrarnos posteriormente algún problema cuando lo pasemos a un servidor web en Internet. Para minimizarlos, deberíamos crear un entorno lo más parecido posible al que tendrá nuestro sitio web cuando esté publicado en Internet.

En plataformas Windows podemos emplear *Internet Information Services* y para plataformas Windows/Linux tenemos *Apache Web Server*[1].

1.6 Instalación de las herramientas necesarias

En principio, el desarrollo web es independiente de la plataforma utilizada. Una página web se crea de la misma forma en un ordenador con Windows que en otro con Linux. En ambas plataformas hay herramientas de libre distribución a disposición gratuita de los usuarios para tal fin. Sin embargo, vamos a utilizar Windows ya que los navegadores que vamos a utilizar para probar nuestras páginas web tienen versiones para Windows, pero no todos ellos disponen de versiones para Linux.

Vamos a necesitar las siguientes herramientas:

1. Servidor web Internet Information Services
2. Editor web Komodo Edit
3. Navegadores Microsoft Edge, Mozila Firefox, Google Chrome, Safari y Opera

1.6.1 Internet Information Services

Windows incorpora un servidor web: Internet Information Services. Este no se instala durante la instalación de Windows, sino que hay que añadirlo como característica. Para ello seguir los siguientes pasos:

1. Abrir el panel de control de Windows.
2. Hacer clic en *Programas y características*.
3. En *Programas y características* hacer clic en *Activar o desactivar las características*

1 La instalación y configuración de un servidor web es tarea para un administrador de sistemas. No es objetivo de este manual ilustrar este proceso, por lo que solamente se indicará donde almacenar los documentos web y como acceder a ellos utilizando un navegador que contacte con el servidor web.

de Windows.

4. Marcar la casilla *Internet Information Services*.
5. Hacer clic en el botón *Aceptar*.

Figura 1.- Web por defecto de IIS

Cuando se reinicie, podemos comprobar si el servidor web está en ejecución abriendo el navegador y escribiendo en la barra de dirección `http://localhost`. Debe de aparecer la anterior pantalla.

Ya podemos colocar nuestras páginas web en el servidor web para acceder a ellas. Los servidores web disponen de lo que se conoce como directorio raíz o carpeta donde se almacenan las páginas web que gestionan y sirven. Naturalmente dentro del directorio raíz podemos crear más carpetas y subcarpetas para organizar mejor todos los documentos web y archivos que formarán nuestro sitio web. En IIS, el directorio raíz está en la carpeta `C:\Inetpub\wwwroot`.

1.6.2 Komodo Edit

Komodo Edit es un editor de código fuente bastante avanzado y extensible. No se limita sólo a proveernos de las diversas herramientas habituales de los editores populares para programadores, sino que va más allá, proporcionando algunas de las utilidades típicas de los entornos de desarrollo profesionales o IDEs.

La verdad es que si queremos editar código fuente en lenguajes de programación orientados para la web, Komodo Edit es una de las opciones más interesantes. Primero

porque es multiplataforma, lo que da soporte tanto a usuarios de Windows como los de otras plataformas como puede ser Mac o Linux. Otra de las principales ventajas de Komodo IDE es que se ofrece con la misma licencia que el navegador de la fundación Mozilla, Firefox, por lo que sobra decir que es un producto de código libre y por tanto, gratuito para cualquier uso. Entre sus características están:

- ✔ Resaltado y coloreado de código fuente, con soporte para casi todos los lenguajes que podemos utilizar al construir una página web.
- ✔ Multi-documento, que permite abrir y editar varios archivos al mismo tiempo.
- ✔ Auto-completado de código con menús contextuales, que aparecen a medida que vamos escribiendo los programas.
- ✔ Revisión de sintaxis en vivo, que nos proporciona información sobre los errores que podemos realizar en la sintaxis de los lenguajes de programación soportados, a medida que vamos realizando los scripts.
- ✔ Vista previa de HTML, para poder ver las páginas web que estamos realizando dentro del propio editor.
- ✔ Gestión de proyectos, con la posibilidad de navegar por las carpetas de nuestro sistema.
- ✔ Posibilidad de conexión con servidores remotos para editar código directamente sobre el archivo publicado en un servidor, por medio de FTP, SFTP, SSH.
- ✔ Posibilidad de extender el programa, con diversos complementos o add-ons creados por terceras personas, como ocurre con el navegador Firefox y sus extensiones.

En la dirección web http://www.activestate.com/komodo-edit/downloads podemos descargar el paquete de instalación de la última versión de Komodo Edit. Una vez descargado solo hay que hacer doble clic sobre él y seguir los pasos del asistente de instalación. Cuando finalice, haremos doble clic sobre el icono del escritorio para comenzar su ejecución.

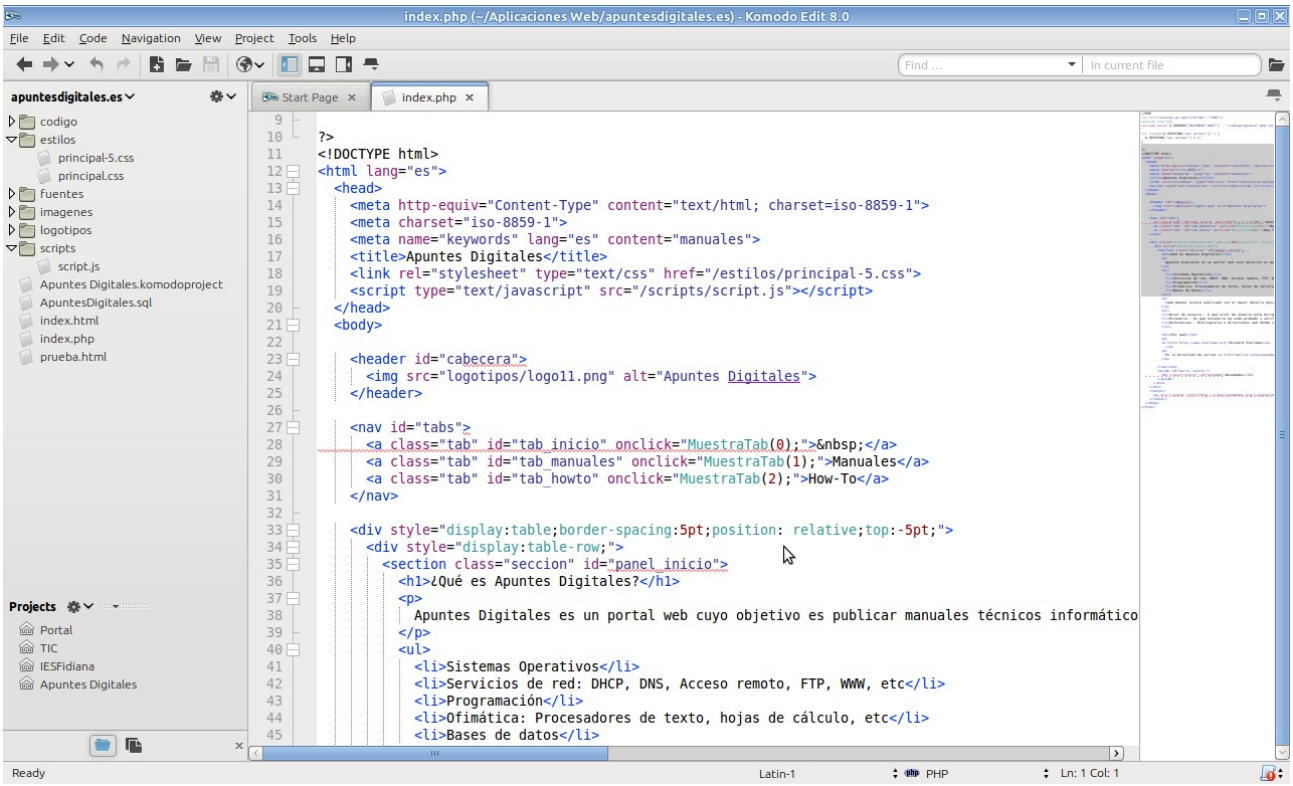


Figura 2.- Komodo Edit

1.6.3 Navegadores

Debemos utilizar cinco navegadores para probar nuestras páginas web. Si nuestro sistema es Windows, disponemos de Microsoft Edge, aunque puede que no en su versión más reciente. Si no es así, se recomienda una actualización.

El resto de navegadores se pueden instalar descargando los paquetes de instalación en las siguientes direcciones web:

1. Mozilla Firefox.- http://www.mozilla.org
2. Google Chrome.- http://www.google.com/intl/es_es/chrome/
3. Safari.- http://support.apple.com/kb/DL1531?viewlocale=es_ES
4. Opera.- http://www.opera.com/computer/linux

Las instalaciones en sistemas Windows sólo consiste en ejecutar el paquete de instalación y seguir las instrucciones del asistente de instalación. Generalmente, las opciones por defecto suelen ser las adecuadas para la instalación correcta de la aplicación. Sin embargo, en el caso de los navegadores es posible que solicite la importación de historial de navegación y favoritos/marcadores junto con la asignación de navegador por defecto del sistema. Queda a elección del usuario la selección de estas opciones.

1.7 Navegadores que soportan HTML 5

Cuando se desarrollan páginas web utilizando la versión 5 de HTML nos vamos a

encontrar que unos navegadores reconocerán parte del etiquetado mientras que otros no. Ya hemos mencionado anteriormente que las compañías que han desarrollado y mantienen los navegadores están implementando poco a poco las nuevas características y funcionalidad de HTML5, pero hasta el momento de escribir este documento, ninguno de ellos lo ha hecho por completo. ¿Cómo podemos saber que navegador viene mejor para nuestro sitio web?

Existe un portal en Internet que nos permite examinar nuestro navegador y asignarle una nota en función de su capacidad para reconocer las características de HTML5. En http://html5test.com podemos someter a nuestro navegador a una prueba que permitira conocer el grado de compatibilidad con HTML5.

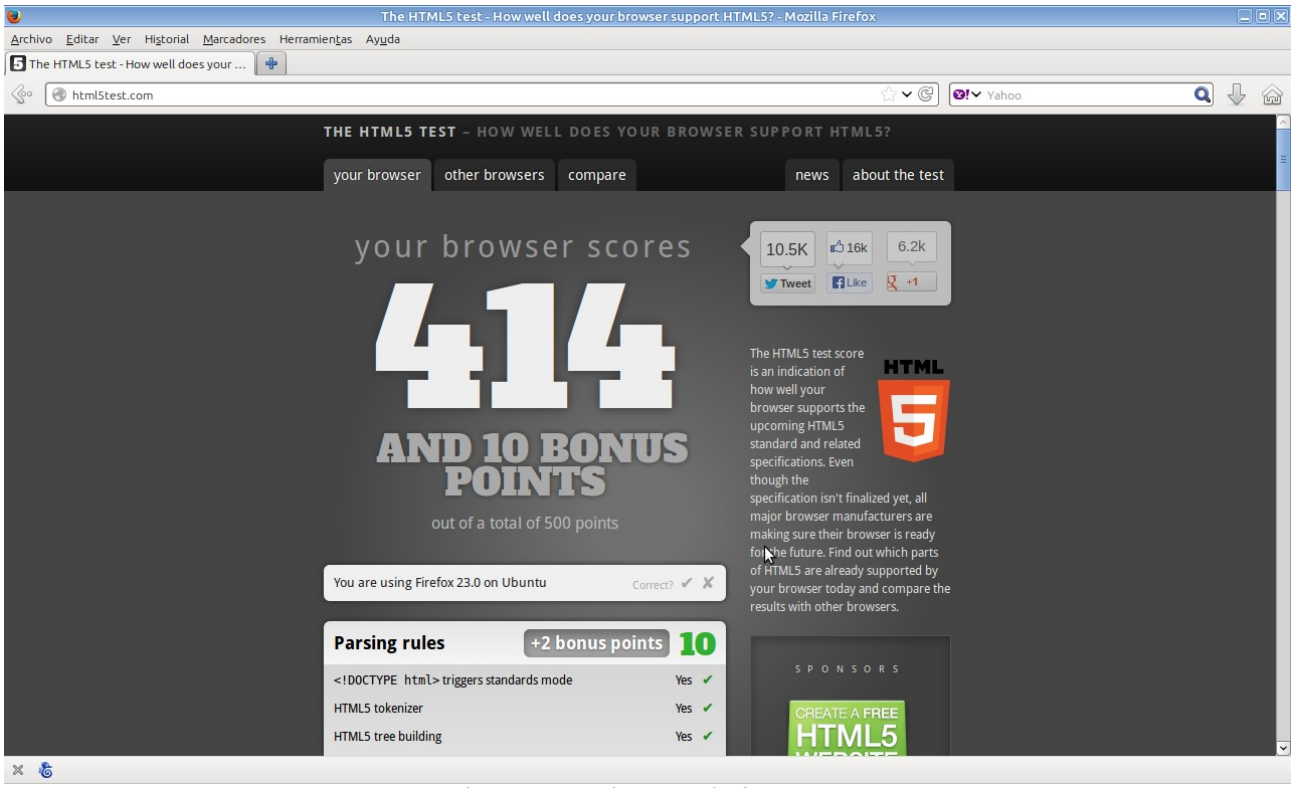


Figura 3.- Prueba para Firefox v23.0

En la imagen anterior se aprecia el resultado de una prueba. Firefox v23.0 ha obtenido una puntuación de 414 sobre 500. Si nos desplazamos hacia abajo veremos la puntuación obtenida sobre diferentes secciones en las que se ha dividido HTML5. En cada una de ellas establece la puntuación obtenido y la puntuación máxima que puede obtener. Debajo, se pueden apreciar cada una de las características y si son reconocidas o no.

1.8 Mi primera página web

Vamos a crear una página web inicial sencilla como ejemplo práctico de HTML. Para ello, abre tu editor de textos y copia y pega el siguiente texto en un nuevo documento.

```
<!DOCTYPE html>
<html lang="es">
    <head>
```

```
        <title>Desarrollo web estático</title>
    </head>
    <body>
        <h1>Bienvenido</h1>
        <p>Esta es mi primera página web.</p>
    </body>
</html>
```

Ahora guarda ese archivo con extensión `.html` en tu disco duro. Para ello accedemos al menú *Archivo* y seleccionamos la opción *Guardar como*.

En la ventana elegimos la carpeta donde deseamos guardarlo, en nuestro caso si queremos que esta página esté disponible a través del servidor web elegiremos `C:\Inetpub\wwwroot`. Después, escribimos su nombre, por ejemplo `mi_pagina.html`. Si ahora abrimos el navegador y en la barra de dirección escribimos `http://localhost/mi_pagina.html` deberemos de ver el siguiente resultado.

Figura 4.- Mi primera página web

En la URL que hemos utilizado te llamará la atención el nombre de servidor `localhost`. Este nombre es un nombre asignado automáticamente por el sistema operativo al PC.

También, podemos ver el resultado sin utilizar el servidor web. Para ello debemos abrir la carpeta donde se encuentra el archivo que contiene la página web creada y hacemos doble clic sobre ella. A continuación se abrirá el navegador por defecto del sistema y mostrará su contenido.

Si ahora hacéis click con el botón derecho sobre la página y elegís la opción de menú *Ver código fuente* veréis como en una ventana aparece el código de nuestra página. Este recurso es de extremada importancia ya que nos permite ver el tipo de técnicas empleadas por otros desarrolladores para la confección de sus páginas.

Respecto a la nomenclatura de archivos de las páginas web o cualquier otro archivo que vaya a estar disponible en el servidor web, se recomienda seguir unas pautas simples y sencillas para evitar problemas. Más adelante, en el capítulo dedicado a la publicación web, veremos con más detalle esta cuestión nada trivial. En principio y mientras tanto, un consejo es que no utilices acentos ni espacios ni otros caracteres raros. También te ayudará escribir siempre las letras en minúsculas. Esto no quiere decir que debes hacer nombres de archivos cortos, es mejor hacerlos descriptivos para que te aclaren lo que hay dentro. Algún caracter como el guión "-" o el guión bajo "_" te puede ayudar a separar las palabras. Por ejemplo `quienes_somos.html`.

2 Sintaxis HTML

El HTML es un lenguaje de marcas que basa su sintaxis en un elemento de base al que llamamos etiqueta. A través de las etiquetas vamos definiendo los elementos del documento, como enlaces, párrafos, imágenes, etc. Así pues, un documento HTML estará constituido por texto y un conjunto de etiquetas para definir la forma con la que se tendrá que estructurar el texto y otros elementos en la página.

HTML suministra una gran variedad de elementos semánticos que pueden usarse para marcar varias estructuras tipográficas. Hay elementos de cabecera para marcar varios niveles de cabecera, un elemento de párrafo para el párrafo, varios elementos de listas para representar diferentes tipos de listas y elementos de tabla para formar tablas.

Es importante distinguir estructura y semántica de contenido, el cual debería ser descrito usando HTML, y su presentación. En un documento, una cabecera puede ser presentada visualmente en una fuente en negrita con un amplio margen superior e inferior para separarla del contenido subyacente. En otro documento, una cabecera puede ser presentada en un color claro, en cursiva y una fuente más informal. Pero sin tener en cuenta la presentación, sigue siendo una cabecera y el marcado puede todavía usar los mismos elementos básicos para identificar estructuras comunes.

El W3C recomienda seguir la sintaxis HTML, ya que la sugieren la mayoría de los autores y es compatible con la mayoría de los navegadores.

2.1 Componentes de un documento web

Un documento web se forma mediante elementos, los cuales pueden tener atributos. Los elementos se utilizan tanto para formar la estructura del documento como para mostrar su contenido. Vamos a ver en detalle estos componentes.

2.1.1 Elementos

Un elemento representa a cualquier parte de una página web. Un párrafo, una tabla, un elemento de lista, incluso la página de web entera se representan mediante elementos. Existen varios tipos de elementos que son:

- ✔ Elementos vacíos, como `br` e `img`.
- ✔ Elementos `<template>`.
- ✔ Elementos de texto sin procesar, como `script` y `style`.
- ✔ Elementos de texto escapado, como `textarea title`.
- ✔ Elementos externos que proceden del espacio de nombres MathGL o SVG.
- ✔ Elementos normales, como `p` o `table`.

Para delimitar el principio y el fin de un elemento se emplean las etiquetas. Los elementos de texto sin procesar, de texto escapado y normales tienen una etiqueta inicio para indicar donde comienzan y una etiqueta final para indicar donde acaban. En algunos

elementos normales la etiqueta final se puede omitir. Los elementos vacíos solamente tienen una etiqueta de inicio. El resto de elementos deben tener la etiqueta inicio y final.

El contenido del elemento se coloca entre la etiqueta inicio y la etiqueta final. El contenido admisible para el elemento dependerá de la definición del elemento. Los elementos vacíos no tienen contenido, por tanto no tienen etiqueta final.

Las etiquetas tienen un nombre de etiqueta, dado por el nombre del elemento. Las etiquetas de inicio deben tener el siguiente formato:

- ✔ Comienzan por el signo menor <
- ✔ El nombre del elemento.
- ✔ Los atributos del elemento, si los hubiera.
- ✔ El signo mayor >

Una etiqueta final tiene el siguiente formato:

- ✔ Comienzan por el signo menor <
- ✔ El nombre del elemento.
- ✔ El signo mayor >

El siguiente ejemplo ilustra el uso de las etiquetas de apertura y cierre en un elemento normal, en este caso un párrafo de texto.

```
<p>El rápido zorro pardo salta encima del perro vago.</p>
```

En ambas etiquetas se permite un espacio en blanco entre el nombre de la etiqueta y el signo mayor, sin embargo es habitual omitirlo. Los nombres de las etiquetas no son sensibles a mayúsculas y minúsculas, aunque suelen escribirse en minúsculas. Evitar en lo posible utilizar diferente capitalización para la etiqueta de apertura y la de cierre para aumentar la claridad del código.

Algunos elementos, sin embargo, no tienen contenido. Estos son conocidos como elementos vacíos. En HTML, la sintaxis anterior no puede usarse para elementos vacíos. Tales elementos no tienen etiqueta de cierre. El siguiente ejemplo incluye un salto de línea en el documento web

```
<br>
```

2.1.2 Atributos

Los atributos se emplean para definir propiedades de los elementos. Los atributos de un elemento se ponen dentro de la etiqueta de inicio después del nombre del elemento y antes de la barra de dividir. Si hay varios se separan por un espacio.

Los atributos modifican el comportamiento básico de los elementos al presentar el contenido. Algunos son globales y se pueden utilizar en cualquier elemento, otros son solo

específicos de un elemento.

Todos los atributos tienen un nombre y pueden tener un valor, separados por el signo =, como en el siguiente ejemplo.

```
<img src=”logotipo.jpg”>
```

En el ejemplo anterior, el atributo `src` tiene el valor `“logotipo.jpg”`. El elemento `img` se emplea para introducir una imagen dentro del documento. Los atributos siempre se ponen en la etiqueta de apertura y nunca en la de cierre.

Hay cuatro sintaxis diferentes que pueden usarse para los atributos y puede usarse cualquiera de ellas, dependiendo de las necesidades específicas del atributo específico. Estas son:

- ✔ Atributo vacío.
- ✔ Valor de atributo sin comillas.
- ✔ Valor de atributo encerrado entre comillas dobles.
- ✔ Valor de atributo encerrado entre comillas simples.

Un atributo vacío es aquél que no tiene valor. Esto es una forma corta de aquellos atributos que no tienen valores, como por ejemplo atributos lógicos, con valor verdadero, si el atributo está presente, o falso si no lo está. Por ejemplo

```
<input disabled>
```

El atributo `disabled` no tiene valor, por tanto no va seguido del signo igual y un texto.

Un atributo con un valor sin comillas es aquel cuyo valor incluye cualquier carácter excepto el espacio en blanco, comillas dobles, comilla simple, signo igual y signo mayor. Por ejemplo

```
<div class=example>
```

Los atributos con valores encerrados entre comillas dobles pueden contener cualquier carácter excepto las comillas dobles. Por ejemplo

```
<img alt=”Imagen del logotipo”/>
```

El caso anterior es análogo a los valores de atributos encerrados entre comillas simples, pueden contener cualquier valor excepto la comilla simple.

```
<img alt='Imagen del logotipo'/>
```

2.1.3 Atributos globales

Un atributo global es aquél que puede asignarse a cualquier elemento HTML. En el borrador de la versión 5 de HTML hay 21 atributos globales, de los cuales los más habituales

son:

Atributo	Descripción
`id="identificador"`	Asigna un identificador único al elemento HTML.
`class="clase"`	Asigna una clase de elemento. Este atributo se emplea para dar formato a los elementos.
`style="propiedades_CSS"`	Atributo para asignar código CSS a un elemento y darle formato.
`title="Texto"`	Fragmento de texto que se asigna al elemento y que aparece en forma de tooltip cuando se sitúa el ratón encima.

2.1.4 Comentarios

Podemos incluir texto dentro de un documento web que no se verá en la pantalla del navegador. Este texto suelen ser notas aclaratorias para el propio desarrollador de la página web. Cualquier texto encerrado entre `<!--` y `-->` será ignorado por el navegador y no se presentará en pantalla. Por ejemplo

```
<!-- Esto es un comentario y no se verá en el navegador -->
```

3 Estructura de un documento HTML

Los documentos HTML están altamente estructurados con partes bien diferenciadas por elementos específicos para definirlas. Un documento HTML está dividido en dos secciones. La cabecera, la cual se utiliza para contener metadatos del documento, tales como el título, hojas de estilo, scripts, etc. y el cuerpo, el cual contiene todo el contenido de la página y que se visualiza en el navegador.

Reproduzcamos el primer ejemplo de página web que vimos en el capítulo anterior.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8"/>
        <title>Desarrollo web estático</title>
    </head>
    <body>
        <h1>Bienvenido</h1>
        <p>Esta es mi primera página web.</p>
    </body>
</html>
```

3.1 Tipo de documento y elemento raíz

Primero, hay que especificar el tipo de documento. En HTML5 consiste en poner al principio del documento un preámbulo con la siguiente línea

```
<!DOCTYPE html>
```

A continuación ponemos la estructura del documento que consiste en una cabecera y un cuerpo, ambos dentro del elemento raíz.

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

El elemento `html` representa la raíz de un documento web. Todos los documentos web contienen solamente una raíz donde se incluyen la cabecera y el cuerpo. Es una buena práctica especificar el idioma del contenido del documento utilizando el atributo `lang` .

3.2 Cabecera del documento

El elemento `head` se emplea para indicar la cabecera del documento y es un contenedor de metadatos. Metadatos es información adicional sobre el documento, como su autor, el título, etc. Los scripts y hojas de estilo también deben incluirse dentro de este elemento. Cada documento solamente tiene un elemento `head`. Para representar esta información adicional se emplean otros elementos, que se incluyen dentro de `head`.

3.2.1 Título del documento

El elemento `title` se emplea para introducir un título al documento. Este título será un texto breve descriptivo del contenido del documento y que aparecerá en la barra de título o en las pestañas del navegador.

Aunque en principio no se imponen límites a la longitud de un título, conviene ser razonable y no usar ni nombre largos nombre muy pequeños. Aunque está directiva no es obligatoria, es conveniente poner título a nuestra página ya que podremos identificarla en todo momento sin tener que verla por completo, para conocer el contenido de la misma.

3.2.2 Elemento meta

El elemento `meta` se emplea para incluir información adicional del documento como el código de caracteres que empleará, autor, palabras clave, etc. Los autores pueden incluir información sobre el documento HTML y otros recursos no accesibles en la red.

Primero, es recomendable incluir un elemento `meta` para especificar el juego de caracteres del contenido del archivo donde se guarda la página web. En Europa Occidental se emplea habitualmente el juego de caracteres utf-8. Sin embargo este juego de caracteres tiene limitados el número de caracteres que puede representar. Es por ello que se recomienda el uso de utf8 como juego de caracteres para garantizar la página web se va a poder visualizar perfectamente independientemente del navegador y lugar del mundo en el que se visualice. Por tanto debemos incluir un elemento `meta` como el siguiente en la cabecera del documento para aseguramos que nuestras páginas se verán correctamente en el navegador.

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

El elemento `meta` no es obligatorio pero se recomienda incluir al menos el conjunto de

caracteres. Si es necesario pueden incluirse más de un elemento `meta`. Por ejemplo, también podemos indicar una descripción breve sobre el contenido del documento

```
<meta name="description" content="Mi primera página web"/>
```

Podemos además incluir un elemento `meta` para indicar información útil para los buscadores.

```
<meta name="keywords" content="HTML5, CSS3"/>
```

Para indicar el autor del documento se añadiría el siguiente elemento `meta`.

```
<meta name="Author" content="Rafael Lozano">
```

3.3 Cuerpo del documento

En el cuerpo del documento se incluirá el contenido de la página web: texto, las imágenes, sonidos, tablas, listas, etc... Como descripción gráfica, podremos decir que todo lo que aparezca entre las directivas `<body>...</body>` será visualizado de una u otra manera en la pantalla del navegador.

Esta parte visible del documento debe estar estructurada. Hay diferentes formas de estructurar y organizar el contenido del cuerpo. Hasta la versión 4, HTML daba liberta el desarrollador web para organizar el contenido mediante diferentes elementos, siendo los más usados `div` y `table`.

Este es el código que producirá nuestra página web. HTML siempre ofreció diferentes formas de construir y organizar la información dentro del cuerpo de un documento. Uno de los primeros elementos provistos para este propósito fue `table`. Las tablas permitían a los diseñadores acomodar datos, texto, imágenes y herramientas dentro de filas y columnas de celdas, incluso sin que hayan sido concebidas para este propósito.

En los primeros días de la web, las tablas fueron una revolución, un gran paso hacia adelante con respecto a la visualización de los documentos y la experiencia ofrecida a los usuarios. Más adelante, gradualmente, otros elementos reemplazaron su función, permitiendo lograr lo mismo con menos código, facilitando de este modo la creación, permitiendo portabilidad y ayudando al mantenimiento de los sitios web.

El elemento `div` comenzó a dominar la escena. Con el surgimiento de webs más interactivas y la integración de HTML y CSS, el uso de `div` se volvió una práctica común. Pero este elemento, así como `table`, no provee demasiada información acerca de las parte del cuerpo que está representando. Desde imágenes a menús, textos, enlaces, códigos, formularios, cualquier cosa puede ir entre las etiquetas de apertura y cierre de un elemento `div`. En otras palabras, la palabra clave `div` solo especifica una división en el cuerpo, como la celda de una tabla, pero no ofrece indicio alguno sobre qué clase de división es, cuál es su propósito o qué contiene.

Para los usuarios estas claves o indicios no son importantes, pero para los navegadores la correcta interpretación de qué hay dentro del documento que se está

procesando puede ser crucial en muchos casos. Después de la revolución de los dispositivos móviles y el surgimiento de diferentes formas en que la gente accede a la web, la identificación de cada parte del documento es una tarea que se ha vuelto más relevante que nunca.

Considerando todo lo expuesto, HTML5 incorpora nuevos elementos que ayudan a identificar cada sección del documento y organizar el cuerpo del mismo. En HTML5 las secciones más importantes son diferenciadas y la estructura principal ya no depende más de los elementos `div` o `table`.

Cómo usamos estos nuevos elementos depende de nosotros, pero las palabras clave otorgadas a cada uno de ellos nos dan ayudan a entender sus funciones. Normalmente una página o aplicación web está dividida entre varias áreas visuales para mejorar la experiencia del usuario y facilitar la interactividad. Las palabras claves que representan cada nuevo elemento de HTML5 están íntimamente relacionadas con estas áreas, como veremos pronto.

La siguiente figura representa un diseño común encontrado en la mayoría de los sitios webs estos días. A pesar del hecho de que cada diseñador crea sus propios diseños, en general podremos identificar las siguientes secciones en cada sitio web estudiado:

Figura 5.- Estructura de un documento web según HTML5

En la parte superior, descripto como Cabecera, se encuentra el espacio donde

usualmente se ubica el logo, título, subtítulos y una corta descripción del sitio web o la página.

Inmediatamente debajo, podemos ver la Barra de Navegación en la cual casi todos los desarrolladores ofrecen un menú o lista de enlaces con el propósito de facilitar la navegación a través del sitio. Los usuarios son guiados desde esta barra hacia las diferentes páginas o documentos, normalmente pertenecientes al mismo sitio web.

El contenido más relevante de una página web se encuentra, en casi todo diseño, ubicado en su centro. Esta sección presenta información y enlaces valiosos. La mayoría de las veces es dividida en varias filas y columnas. En el ejemplo de la figura se utilizaron solo dos columnas: Información Principal y Barra Lateral, pero esta sección es extremadamente flexible y normalmente diseñadores la adaptan acorde a sus necesidades insertando más columnas, dividiendo cada columna entre bloques más pequeños o generando diferentes distribuciones y combinaciones. El contenido presentado en esta parte del diseño es usualmente de alta prioridad. En el diseño de ejemplo, Información Principal podría contener una lista de artículos, descripción de productos, entradas de un blog o cualquier otra información importante, y la Barra Lateral podría mostrar una lista de enlaces apuntando hacia cada uno se esos ítems. En un blog, por ejemplo, esta última columna ofrecerá una lista de enlaces apuntando a cada entrada del blog, información acerca del autor, etc...

En la base de un diseño web clásico siempre nos encontramos con una barra más que aquí llamamos Institucional. La nombramos de esta manera porque esta es el área en donde normalmente se muestra información acerca del sitio web, el autor o la empresa, además de algunos enlaces con respecto a reglas, términos y condiciones y toda información adicional que el desarrollador considere importante compartir. La barra Institucional es un complemento de la Cabecera y es parte de lo que se considera estos días la estructura esencial de una página web, como podemos apreciar en el siguiente ejemplo.

Figura 6.- Ejemplo de estructura de cuerpo según HTML5

La figura anterior es una representación de un blog normal. En este ejemplo se puede claramente identificar cada parte del diseño considerado anteriormente.

1. Cabecera
2. Barra de Navegación
3. Sección de Información Principal
4. Barra Lateral
5. El pie o la barra Institucional

Esta simple representación de un blog nos puede ayudar a entender que cada sección definida en un sitio web tiene un propósito. A veces este propósito no es claro pero en esencia se encuentra siempre allí, ayudándonos a reconocer cualquiera de las secciones descriptas anteriormente en todo diseño.

HTML5 considera esta estructura básica y provee nuevos elementos para diferenciar y declarar cada una de sus partes. A partir de ahora podemos decir al navegador para qué es cada sección

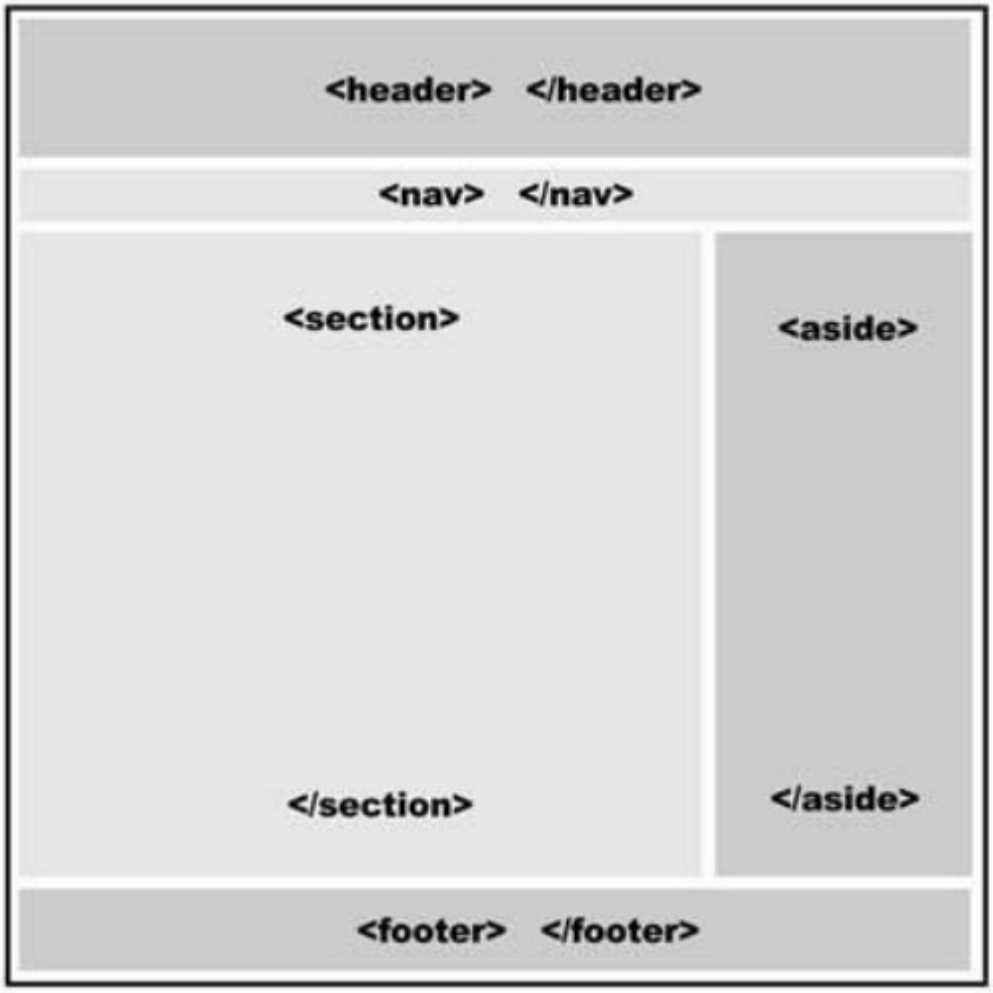


Figura 7.- Elementos de estructura HTML5

HTML5 ha incluido una serie de elementos para representar zonas concretas de una página web y que responde a un diseño muy habitual hoy en día en la mayoría de los sitios web. Estos elementos son `header`, `section`, `aside`, `nav`, `footer`, `article`.

3.3.1 Cabecera del cuerpo

Uno de los nuevos elementos incorporados en HTML5 es `header`. El elemento `header` no debe ser confundido con `head` usado antes para construir la cabecera del documento. Del mismo modo que `head`, la intención de `header` es proveer información introductoria (títulos, subtítulos, logos), pero difiere con respecto a `head` en su alcance. Mientras que el elemento `head` tiene el propósito de proveer información acerca de todo el documento, `header` es usado solo para el cuerpo o secciones específicas dentro del cuerpo.

Veamos el siguiente ejemplo

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

```
        <title>Desarrollo web estático</title>
    </head>
    <body>
        <header>
            <img src="logo.jpeg"/>
            <h1>Desarrollo web estático con HTML y CSS</h1>
        </header>
    </body>
</html>
```

En el ejemplo anterior, definimos el título de la página web utilizando el elemento `header`. La inserción del elemento `header` representa el comienzo del cuerpo y por lo tanto de la parte visible del documento. En este caso hemos introducido una imagen con el logo y un encabezado de nivel 1.

3.3.2 Barra de navegación

El siguiente elemento de la estructura del documento web es la barra de navegación. Esta barra es generada en HTML5 con el elemento `nav`.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8"/>
        <title>Desarrollo web estático</title>
    </head>
    <body>
        <header>
            <img src="logo.jpeg"/>
            <h1>Desarrollo web estático con HTML y CSS</h1>
        </header>
        <nav>
            <a href="inicio.html">Inicio</a>
            <a href="lenguaje_html.html">HTML</a>
            <a href="lenguaje_css.html">CSS</a>
        </nav>
    </body>
</html>
```

Como se puede apreciar en el ejemplo anterior, el elemento `nav` se encuentra dentro del cuerpo del documento después de la cabecera (`header`), pero no dentro de esta. Esto es porque la barra de navegación no es parte de la cabecera sino una nueva sección.

El elemento `nav` fue creado para ofrecer ayuda para la navegación, como en menús principales o grandes bloques de enlaces, y debería ser utilizado de esa manera.

3.3.3 Información principal

Siguiendo nuestro diseño estándar nos encontramos con las columnas que hemos llamado Información Principal . Como explicamos anteriormente, la columna Información Principal contiene la información más relevante del documento y puede ser encontrada en diferentes formas (por ejemplo, dividida en varios bloques o columnas). Debido a que el

propósito de estas columnas es más general, el elemento en HTML5 que especifica estas secciones se llama simplemente `section`.

Sigamos ampliando nuestro ejemplo con una sección.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8"/>
        <title>Desarrollo web estático</title>
    </head>
    <body>
        <header>
            <img src="logo.jpeg"/>
            <h1>Desarrollo web estático con HTML y CSS</h1>
        </header>
        <nav>
            <a href="inicio.html">Inicio</a>
            <a href="lenguaje_html.html">HTML</a>
            <a href="lenguaje_css.html">CSS</a>
        </nav>
        <section>
        <h2>Desarrollo web estático</h2>
        <p>
        Bienvenidos al portal de ayuda al desarrollo web
estático...
        </p>
        </section>
    </body>
</html>
```

En el ejemplo anterior hemos incluido una sección a la sección principal del documento web.

3.3.4 Barra lateral

En un típico diseño web la columna llamada Barra Lateral se ubica al lado de la columna Información Principal. Esta es una columna o sección que normalmente contiene datos relacionados con la información principal pero que no son relevantes o igual de importantes. Por ejemplo, en nuestro caso podríamos introducir una lista con las novedades que se producen en el portal.

La información dentro de esta barra está relacionada con la información principal pero no es relevante por sí misma. Siguiendo el mismo ejemplo podemos decir que las novedades del portal son relevantes pero los enlaces y las pequeñas reseñas sobre esas novedades son solo una ayuda para la navegación.

En HTML5 podemos diferenciar esta clase secundaria de información utilizando el elemento `aside`.

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

```
        <meta charset="utf-8"/>
        <title>Desarrollo web estático</title>
    </head>
    <body>
        <header>
            <img src="logo.jpeg"/>
            <h1>Desarrollo web estático con HTML y CSS</h1>
        </header>
        <nav>
            <a href="inicio.html">Inicio</a>
            <a href="lenguaje_html.html">HTML</a>
            <a href="lenguaje_css.html">CSS</a>
        </nav>
        <section>
        <h2>Desarrollo web estático</h2>
        <p>
        Bienvenidos al portal de ayuda al desarrollo web
estático...
        </p>
        </section>
        <aside>
        <h2>Novedades</h2>
        <p>
        10 de Agosto - Se ha incluido un artículo sobre los
gradientes en las imágenes de fondo
        </p>
        <p>
        12 de Agosto - Se incluye un zip con miles de iconos en
formato png.
        </p>
        ...
        </aside>
    </body>
</html>
```

En el ejemplo anterior, la barra lateral contendrá pequeñas reñeñas de la sección novedades para informar sobre los últimos cambios del portal.

El elemento `aside` podría estar ubicado del lado derecho o izquierdo de nuestra página de ejemplo, la etiqueta no tiene una posición predefinida. El elemento `aside` solo describe la información que contiene, no el lugar dentro de la estructura.

Este elemento puede estar ubicado en cualquier parte del diseño y ser usado siempre y cuando su contenido no sea considerado como el contenido principal del documento. Por ejemplo, podemos usar `aside` dentro del elemento `section` o incluso insertado entre la información relevante, como en el caso de una cita.

3.3.5 Pie del documento

Para finalizar la construcción de la plantilla o estructura elemental de nuestro documento HTML5, solo necesitamos un elemento más. Ya contamos con la cabecera del cuerpo, secciones con ayuda para la navegación, información importante y hasta una barra lateral con datos adicionales, por lo tanto lo único que nos queda por hacer es cerrar

nuestro diseño para otorgarle un final al cuerpo del documento. HTML5 provee un elemento específico para este propósito llamado `footer`.

```
<!DOCTYPE html>
<html lang=”es”>
    <head>
        <meta charset=”utf-8”/>
        <title>Desarrollo web estático</title>
    </head>
    <body>
        <header>
            <img src=”logo.jpeg”/>
            <h1>Desarrollo web estático con HTML y CSS</h1>
        </header>
        <nav>
            <a href=”inicio.html”>Inicio</a>
            <a href=”lenguaje_html.html”>HTML</a>
            <a href=”lenguaje_css.html”>CSS</a>
        </nav>
        <section>
        <h2>Desarrollo web estático</h2>
        <p>
        Bienvenidos al portal de ayuda al desarrollo web
estático...
        </p>
        </section>
        <aside>
        <h2>Novedades</h2>
        <p>
        10 de Agosto - Se ha incluido un artículo sobre los
gradientes en las imágenes de fondo
        </p>
        <p>
        12 de Agosto - Se incluye un zip con miles de iconos en
formato png.
        </p>
        ...
        </aside>
        <footer>
        <p>
        Desarrollo web estático se distribuye bajo licencia
Creative Commons Atribución-NoComercial-CompartirIgual 3.0
Unported
        </p>
        </footer>
    </body>
</html>
```

En el típico diseño de una página web la sección llamada Institucional será definida por etiquetas `footer`. Esto es debido a que la barra representa el pie del documento y esta parte de la página web es normalmente usada para compartir información general sobre el autor o la organización detrás del proyecto.

Generalmente, el elemento `footer` representará el final del cuerpo de nuestro

documento y tendrá el propósito descripto anteriormente. Sin embargo, `footer` puede ser usado múltiples veces dentro del cuerpo para representar también el final de diferentes secciones.

3.3.6 Formato del diseño

En la siguiente imagen vemos el resultado del ejemplo final donde tenemos definida toda la estructura del documento.

Figura 8.- Ejemplo estructura del documento según HTML5

Como podemos ver, los nuevos elementos de diseño HTML5 no tienen un formato predefinido. El resultado anterior no se parece a la distribución de las diferentes partes. Vemos que la barra lateral está debajo de la información principal. En realidad los elementos han seguido el flujo normal HTML, donde ya sabemos que tienen un posicionamiento por defecto estático y, por tanto, aparecen en la pantalla del navegador uno detrás de otro en el mismo orden en que aparecen en el código HTML.

Nos corresponde entonces a nosotros dar formato a nuestro diseño para colocar cada elemento de la estructura en su lugar.

4 Elementos textuales

Tras haber visto la estructura básica de un documento HTML, pasamos ahora a ver la parte que hace referencia a lo que es el cuerpo del documento, donde irá incluido todo el contenido a visualizar en la pantalla del navegador. Todos los elementos que vamos a ver a continuación deben introducirse dentro del elemento `body` o de lo contrario no se verán.

En los documentos HTML, hay una peculiaridad con respecto a los editores de texto, Debemos de indicar con elementos cualquier componente de la página: párrafos, saltos de línea, títulos, etc. Podemos escribir texto dentro del cuerpo del documento, sin elementos,

pero el texto rellenaría la ventana del navegador y al no estar en ningún elemento, resultaría muy difícil darle formato. Así que se recomienda que todo el texto del documento esté dentro de un elemento.

4.1 Párrafos

En HTML, los párrafos se representan por el elemento `p`. Un párrafo es un bloque de texto que está separado de otros bloques de texto adyacentes por medio de una línea en blanco. Por ejemplo

```
<p>
En este ejemplo se muestra como pese a que no nos importe el
poder ver los retornos de carro en distintas posiciones, el
elemento de párrafo, lo que hace es que cuando finaliza el ancho
de pantalla de que dispone, realiza un retorno de carro y
continúa con el texto en la siguiente línea.
</p>
<p>
Si reducimos el ancho de la pantalla del visualizador veremos
como el elemento de párrafo se adapta a este ancho, dando un
formato distinto al texto.
</p>
```

Si visualizamos en el navegador los párrafos anteriores veríamos lo siguiente

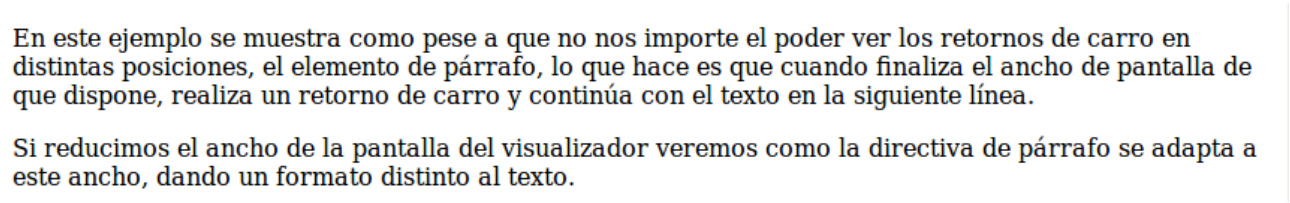


Figura 9.- Párrafos

El elemento de párrafo no debería ser usado para texto que dispone de elementos más adecuados acorde su naturaleza.

4.2 Saltos de línea

Los saltos de línea puestos en el archivo HTML no son respetados por el navegador. Por ejemplo, si tuviéramos el siguiente código HTML.

```
<p>
Esto es un texto
que muestra como los
retornos de carro que pongo
no se respetan en el
navegador
</p>
```

Veremos que todo el texto sale seguido y solamente hará salto de línea cuando llegue al extremo derecho de la pantalla.

Esto es un texto que muestra como los retornos de carro que pongo no se respetan en el navegador

Figura 10.- El texto aparece contiguo a pesar de los saltos de línea

Esto es porque HTML solamente reconoce un espacio en blanco para separar una palabra de la siguiente. Ignora en el archivo los saltos del línea, espacios en blanco adicionales, tabuladores, etc

Sin embargo, disponemos del elemento vacío `br` para introducir un salto de línea donde más nos convenga. Si modificamos el párrafo anterior y añadimos elementos `br` tendremos lo siguiente.

```
<p>
Esto es un texto<br>
que muestra como los<br>
retornos de carro que pongo<br>
no se respetan en el<br>
navegador
</p>
```

Ahora el resultado es diferente

Esto es un texto
que muestra como los
retornos de carro que pongo
no se respetan en el
navegador

Figura 11.- Saltos de línea

Allí donde haya un elemento `br`, se introducirá un salto de línea. No conviene utilizar elementos `br` para separar bloques de texto o para añadir más espacio entre párrafos.

4.3 Texto preformateado

Cuando hablé de los retornos de carro, dije que para poder ser visualizados, deberíamos colocar un elemento `br` al final de la línea para poder visualizarlos correctamente en el navegador. Sin embargo, contamos con el elemento `pre` que permite visualizar el texto tal y esté en el archivo HTML .

Todo el texto incluido en el elemento `pre` se mostrará tal y como lo escribamos, incluyendo las tabulaciones y los retornos de carro. Al finalizar un bloque de texto en un elemento `pre` se incorpora automáticamente un salto de línea, pero en este caso no se deja una línea en blanco con respecto al siguiente elemento. Ejemplo:

```
<pre>
LUNES | MARTES | MIERCOLES | JUEVES | VIERNES
-----------------------------------------
100   | 200    | 200       | 150    | 500
-----------------------------------------
</pre>
```

Este cuadro aparecerá tal cual lo podemos visualizar aquí en el navegador.

```
LUNES | MARTES | MIERCOLES | JUEVES | VIERNES
-----------------------------------------
100   | 200    | 200       | 150    | 500
-----------------------------------------
```

Figura 12.- Texto preformateado

4.4 Encabezados

Para poner un título se emplea un elemento de encabezado. Disponemos de seis elementos de encabezados `h1`, `h2`, `h3`, `h4`, `h5` y `h6` en orden de tamaño., es decir `h1` es el encabezado de nivel 1, `h2` de nivel 2, etc.

El bloque de texto dentro de un elemento de encabezado aparecerá por defecto en negrita y con un tamaño mayor que el texto normal. Además, habrá saltos de línea implícitos antes y después del elemento.

Si introducimos el siguiente ejemplo en un archivo HTML.

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

Veremos el siguiente resultado

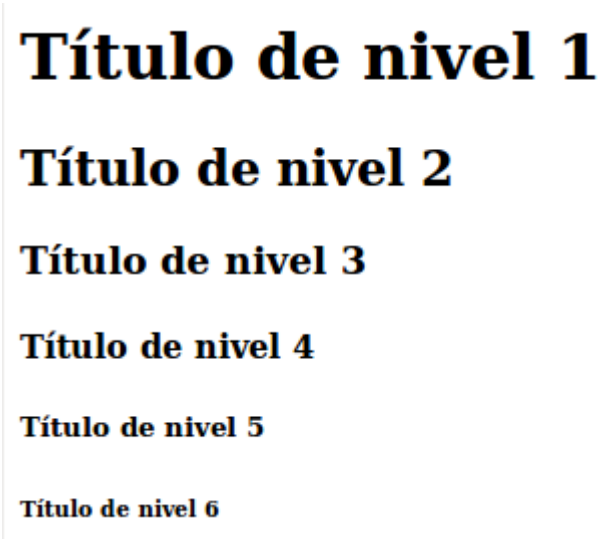


Figura 13.- Encabezados

Si vamos a poner un título y un subtítulo a una sección disponemos del elemento `hgroup` para agrupar encabezados. Generalmente incorporamos elementos `h1` para declarar un título y para la cabecera de una sección del documento puede ser suficiente. Sin embargo, también es posible que necesitemos agregar un subtítulo con un elemento `h2` y un subtítulo de nivel 3 con un elemento `h3`.

Para construir este tipo de cabeceras, donde una sección tiene más de un elemento de encabezado, estos elementos deben ser agrupados utilizando un elemento `hgroup`. Así, podemos aprovechar el resto de los elementos de encabezado, pero siempre considerando que por propósitos de procesamiento interno, y para evitar generar múltiples secciones durante la interpretación del documento por parte del navegador. Por esta razón, HTML5 provee el elemento `hgroup`. Veamos el siguiente ejemplo

```
<hgroup>
    <h1>Introducción a la Informática</h1>
    <h2>Concepto de información</h2>
</hgroup>
```

Vemos que hay dos elementos de encabezado juntos. Por tanto se agrupan con un elemento `hgroup`.

4.5 Separadores

El elemento vacío `hr` se emplea para añadir una línea horizontal que delimita dos partes del documento de forma gráfica. Este elemento dispone de algunos atributos interesantes.

`size="valor numérico"` Permite indicar el grosor de la línea en cuestión.

`width="valor numérico"` Permite indicar la anchura en pixeles o el tanto por ciento del ancho de la pantalla que ocupará la línea.

`align="left|center|right"` Determina la forma de alinear la barra en la pantalla del navegador (sus valores respectivos son izquierda, derecha y centro). Por defecto el valor, es alineado a la izquierda.

`noshade` Elimina, en caso de poner este atributo, el efecto tridimensional de la barra, eliminando la sombra que proyecta sobre el documento.

4.6 Elementos textuales en línea

Hasta el momento hemos visto elementos de bloque para introducir texto. Cuando decimos un elemento de bloque queremos decir que el elemento se distribuye por la ventana del navegador de arriba a abajo, es decir, el elemento lleva implícito un salto de línea anterior al elemento y otro posterior al mismo.

Un párrafo es un elemento de bloque, ya que los párrafos van de arriba a abajo en el orden en que aparecen en el archivo HTML. Lo mismo ocurre con los elementos de encabezado. Volvamos al ejemplo anterior de elementos de encabezado agrupados

```
<hgroup>
    <h1>Introducción a la Informática</h1>
    <h2>Concepto de información</h2>
</hgroup>
```

Si visualizamos este ejemplo en el navegador veremos lo siguiente

Introducción a la Informática

Concepto de información

Figura 14.- Elementos de encabezado

Vemos que los encabezados de nivel 1 y 2 aparecen separados por un salto de línea, aunque ambos podrían perfectamente estar uno detrás de otro en la misma línea. Esto es así porque los elementos de cabecera `h1` y `h2` son elementos de bloque, forman un bloque y por tanto tienen un salto de línea antes y después del elemento.

Sin embargo hay otros elementos que se emplean para introducir texto que son elementos en línea. Esto significa que su distribución por la ventana del navegador será de izquierda a derecha en el orden en que aparezcan en el archivo HTML. No tendrán un salto de línea anterior y posterior como los elementos de bloque. Aunque existen muchos, nos limitaremos a ver los más habituales.

4.6.1 Citas textuales

El elemento `cite` representa una cita textual de algo, por ejemplo un libro, un trabajo, un poema, una canción, una película, un programa de televisión, un juego, un cuadro, etc. Esta obra puede estar entrecomillada o referenciada en detalle. Veamos el siguiente ejemplo

```
<p>Mi libro favorito es <cite>Yo, Claudio</cite> por
Robert Graves. Mi comic favorito es <cite>Sargento Rock</cite>
por DC Comics. Mi canción favorita es <cite>Every rose has its
thorns</cite> por Poison.</p>
```

Veremos el siguiente resultado.

Mi libro favorito es *Yo, Claudio* por Robert Graves. Mi comic favorito es *Sargento Rock* por DC Comics. Mi canción favorita es *Every rose has its thorns* por Poison.

Figura 15.- Uso de títulos

Como podemos ver las citas textuales aparecen en cursiva.

4.6.2 Texto entrecomillado

El elemento `q` se emplea para citar textualmente una frase o fragmento de texto de alguien. En la práctica supone que el texto del elemento `q` aparecerá entre comillas dobles. Veamos el siguiente ejemplo

```
<p>El hombre dijo <q>Las cosas que son imposibles simplemente
necesitan más tiempo</q>. No estoy de acuerdo con él.</p>
```

Si vemos el resultado en el navegador veremos que la frase textual está entrecomillada.

4.6.3 Acrónimos

Un acrónimo o abreviatura son las iniciales de un término compuesto de varias

palabras. En ocasiones, resulta más cómodo incluir en el documento web el acrónimo y que la definición completa del término aparezca como mensaje emergente. Esto se consigue con el elemento `abbr`. Veamos el siguiente ejemplo.

```
<p>El servicio <abbr title="World Wide Web">WWW</abbr> se
desarrolló en el <abbr title="Conseil Européen pour la Recherche
Nucléaire">CERN</abbr></p>
```

Como podemos observar, los acrónimos WWW y CERN aparecen en elementos `abbr`. La definición del acrónimo tiene que introducirse en un atributo `title`. En el navegador veremos el siguiente resultado

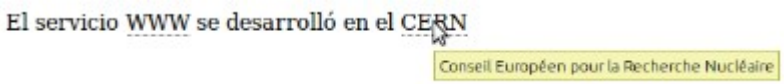


Figura 16.- Acrónimos

En la práctica, el acrónimo aparece con un ligero subrayado y si situamos el ratón encima aparece la definición completa en forma de mensaje emergente.

4.6.4 Superíndices y subíndices

El elemento `sup` representa un superíndice y el elemento `sub` representa un subíndice. Ambos se suelen utilizar en lenguaje científico. Veamos el siguiente ejemplo

```
<p>El aumento de CO<sub>2</sub> en la atmósfera está provocando
el efecto invernadero.</p>
<p>La expresión E=mc<sup>2</sup> representa la equivalencia
entre masa y energía según <cite>La Teoría de la
Relatividad</cite> de Albert Einstein</p>
```

Tendremos el siguiente resultado en el navegador

El aumento de CO_2 en la atmósfera está provocando el efecto invernadero.

La expresión $E=mc^2$ representa la equivalencia entre masa y energía según *La Teoría de la Relatividad* de Albert Einstein

Figura 17.- Superíndices y subíndices

4.6.5 Texto resaltado

El elemento `mark` se emplea para resaltar texto. Un texto introducido mediante este elemento aparecerá con color de fondo amarillo. Suele emplearse en resultados de búsquedas para resaltar el término buscado. Veamos el siguiente ejemplo.

```
<p>Tengo también algunos <mark>minino</mark>s que me visitan
estos días. Son realmente lindos. Creo que les gusta mi jardín.
Quizás debería adoptar un <mark>minino</mark>.</p>
```

En el navegador veremos el siguiente resultado

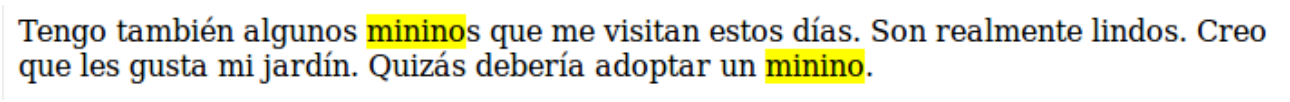


Figura 18.- Marcas

4.6.6 Elemento span

Un elemento `span` no modifica el aspecto visual del texto que contiene. Este elemento se usa para asignar formato a un fragmento de texto mediante los atributos `style`, `class` o `id` que se verán más adelante. El elemento `span` se verá con detalle al asignar formato con CSS.

4.6.7 Elementos para estilos

Hay tres elementos que permiten poner el texto en cursiva, negrita y subrayado. Son respectivamente `i`, `b` y `n`. De acuerdo a la especificación del W3C, oficialmente el elemento `i` se emplea para modificar el sentido del discurso del texto o la calidad del texto. El elemento `b` se emplea para resaltar texto sin necesidad de marcarlo y el elemento `u` se emplea para enfatizar el texto.

En la práctica supone que los elementos `i` ponen el texto en cursiva, los elementos `b` en negrita y los elementos `u` subrayados. Se recomienda no emplear estos elementos y asignar estilos de formato mediante CSS.

4.7 Referencias de caracteres con nombre

Existen algunos caracteres o símbolos que no aparecen en el teclado y que en ocasiones necesitaremos introducirlos en el texto. Símbolos como ©, ½, £, « o ». En HTML se definen una serie de referencias de caracteres con nombre para poder insertarlos en nuestros documentos web y que puedan visualizarse en pantalla.

Estos caracteres en el código HTML se forma con un & (o ampersand) seguido de uno de los tres siguientes:

1. Código número en base 10 ➔ En este caso se indica con # seguido de un entero en base 10.
2. Código numérico en base hexadecimal ➔ En este caso se indica # seguido de x o X y, a continuación el código numérico en base hexadecimal.
3. Nombre ➔ El nombre del carácter.

La referencia termina con punto y coma.

Por ejemplo, si quisiéramos incluir en nuestro documento, mezclado con el texto, un carácter de copyright © podríamos utilizar cualquier de las siguientes formas:

- ✔ `&#169;`
- ✔ `&#xA09;`
- ✔ `&copy;`

Estos códigos corresponden al valor Unicode del carácter. Podemos encontrar una referencia de todos los caracteres nombrados en https://www.w3.org/TR/html52/syntax.html#named-character-references.

5 Introducción a CSS

En HTML se unen etiquetas para contenidos, estructuras y formato en un único documento. Sin embargo, a principios de la década de los noventa, HTML no poseía todas las capacidades que posee en la actualidad y muchos desarrolladores se quejaban de lo poco que podían controlar el aspecto de sus documentos y lo complejo que resultaba el archivo HTML al estar mezclados contenidos, estructuras y formato. Por eso, el W3C decidió crear un sistema mediante el cual las instrucciones de formato se encontraran separadas de los otros elementos. Nacieron así la versión HTML 4, donde están desaconsejados los elementos de formato, y CSS (*Cascading Style Sheets*, Hojas de estilo en cascada), un lenguaje para dotar de formato a los elementos HTML.

Aunque el uso de CSS parecía al principio un poco tedioso, rápidamente se descubrió su gran utilidad. Con él se obtenía la ventaja de poder aplicar mediante una instrucción un determinado formato a todo un sitio web en vez de a un solo elemento (párrafo, imagen, tabla, etc.). Es decir, si se tenía que mantener un portal entero y se decidía cambiar algún aspecto de su apariencia, bastaba cambiar unas pocas líneas de código para modificar el estilo de numerosos elementos sin mayor esfuerzo. Sin duda, esto era mucho mejor que adentrarse en los vericuetos del archivo HTML e ir cambiando uno a uno el aspecto de los elementos. CSS permitía, en definitiva, que los sitios web tuvieran un aspecto homogéneo sin necesidad de horas y horas de trabajo.

Además, CSS ha ido evolucionando con el tiempo y actualmente ofrece muchas mas posibilidades que las etiquetas y extensiones de HTML jamás tuvieron. Con CSS se pueden cambiar el tamaño, grosor, inclinación, altura de línea, colores de fondo y primer plano, espaciado y alineamiento del texto, decidir si debería estar en subrayado, tachado o parpadeante, convertirlo en mayúsculas, o minúsculas, etcétera.

Finalmente, otro antecedente que ha hecho necesario el desarrollo de esta tecnología consiste en que las páginas web tienen mezclado en su código HTML el contenido del documento con las etiquetas necesarias para darle forma. Esto tiene sus inconvenientes ya que la lectura del código HTML se hece pesada y difícil a la hora de buscar errores o depurar las páginas. Aunque, desde el punto de vista de la riqueza de la información y la utilidad de las páginas a la hora de almacenar su contenido, es un gran problema que estos textos estén mezclados con etiquetas incrustadas para dar forma a estos: se degrada su utilidad.

5.1 ¿Qué puedo hacer con CSS?

CSS es un lenguaje de estilo que define la presentación de los documentos HTML. Por ejemplo, CSS abarca cuestiones relativas a fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo, posicionamiento avanzado y muchos otros temas.

Es posible usar HTML, o incluso abusar del mismo, para añadir formato a los sitios web. Sin embargo, CSS ofrece más opciones y es más preciso y sofisticado. CSS está soportado por todos los navegadores hoy día.

5.2 ¿Qué diferencias hay entre HTML y CSS?

HTML se usa para estructurar el contenido; CSS se usa para formatear el contenido previamente estructurado. Al principio, el lenguaje HTML sólo se usaba para añadir estructura al texto. Los autores podían marcar sus textos diciendo "esto en un título" o "esto es un párrafo", usando los elementos HTML `h1` y `p`, respectivamente.

A medida que la Web fue ganando popularidad, los diseñadores empezaron a buscar posibilidades para añadir formato a los documentos en línea. Para satisfacer esta reclamación, los fabricantes de los navegadores (en ese momento, Netscape y Microsoft) inventaron nuevos elementos HTML, entre los que se encontraban, por ejemplo, `font`, que se diferenciaba de las etiquetas originales HTML en que definían el formato y no la estructura.

Esto también llevó a una situación en la que las etiquetas estructurales originales, por ejemplo, `table`, se usaban cada vez más de manera incorrecta para dar formato a las páginas en vez de para añadir estructura al texto. Muchas nuevas etiquetas que añadían formato, por ejemplo, `blink`, sólo las soportaban un tipo determinado de navegador. "Necesitas el navegador X para visualizar esta página" se convirtió en una declaración de descargo común en los sitios web.

CSS se inventó para remediar esta situación, proporcionando a los diseñadores web con sofisticadas oportunidades de presentación soportadas por todos los navegadores. Al mismo tiempo, la separación de la presentación de los documentos del contenido de los mismos, hace que el mantenimiento del sitio sea mucho más fácil.

5.3 ¿Qué beneficios me ofrece CSS?

CSS fue toda una revolución en el mundo del diseño web. Entre los beneficios concretos de CSS encontramos:

- ✔ Control de la presentación de muchos documentos desde una única hoja de estilo;
- ✔ Control más preciso de la presentación;
- ✔ Aplicación de diferentes presentaciones a diferentes tipos de medios (pantalla, impresión, etc.);
- ✔ Numerosas técnicas avanzadas y sofisticadas.
- ✔ Posibilidad de añadir efectos visuales a los elementos

5.4 Características CSS

El modo de funcionamiento de las CSS consiste en definir, mediante una sintaxis especial, la forma de presentación que le aplicaremos a:

- ✔ Un sitio web entero, de modo que se puede definir la forma de todo el sitio web de una sola vez.
- ✔ Un documento HTML o página, se puede definir la forma, en un pequeño trozo de código en la cabecera, a toda la página.
- ✔ Una porción del documento, aplicando estilos visibles en un fragmento de la página web.
- ✔ Un elemento concreto, llegando incluso a poder definir varios estilos diferentes para un solo elemento. Esto es muy importante ya que ofrece potencia en nuestra programación. Podemos definir, por ejemplo, varios tipos de párrafos: en rojo, en azul, con margenes, sin ellos...

La potencia de la tecnología salta a la vista. Pero no solo se queda aquí, ya que además esta sintaxis CSS permite aplicar al documento formato de modo mucho más exacto. Si antes el HTML se nos quedaba corto para maquetar las páginas y teníamos que utilizar trucos para conseguir nuestros efectos, ahora tenemos muchas más herramientas que nos permiten definir esta forma:

- ✔ Podemos definir la distancia entre líneas del documento.
- ✔ Se puede aplicar identado a las primeras líneas del párrafo.
- ✔ Podemos colocar elementos en la página con mayor precisión, y sin lugar a errores.
- ✔ Y mucho más, como definir la visibilidad de los elementos, margenes, subrayados, tachados...

Además, con el HTML tan sólo podíamos definir atributos en las páginas con pixeles y porcentajes, ahora podemos definir utilizando muchas más unidades como:

- ✔ Pixels (px) y porcentaje (%), como antes.
- ✔ Pulgadas (in)
- ✔ Puntos (pt)
- ✔ Centímetros (cm)

6 Aplicando CSS a un documento HTML

CSS sirve para dotar de estilo a los elementos que componen una página o un sitio web. Aunque existen diversas formas de utilizar este lenguaje, la forma habitual de hacerlo es mediante una hoja de estilo.

Una hoja de estilo es el conjunto de instrucciones que definen los formatos de los elementos HTML de la página a la que afecta. Dicho de una forma más simple: se encuentra «vinculada» a una página HTML y dota de estilo a todos o algunos de tos elementos que la componen.

El código que compone la hoja de estilo está formado por una o más reglas de estilo. Éstas son las declaraciones de los formatos que adoptarán los elementos de la página o sitio web a la que está destinada la hoja de estilo. Mediante la regla de estilo se identifica el elemento HTML que se desea seleccionar y la apariencia que se le quiere dar (por ejemplo, que todas las fotos lleven un marco de color rojo).

Si tenemos una página web y se desea dotarla de estilo mediante CSS, ¿cómo debería indicarse? Existen cuatro métodos:

- ✔ En línea
- ✔ Mediante ámbito
- ✔ Con hojas de estilo internas
- ✔ Con hojas de estilo externas

Las dos últimas son las preferidas. Vamos a verlas en detalla

6.1 Estilo en línea

Para aplicar estilos a elementos HTML directamente, se escribe el código CSS dentro del atributo `style` del elemento HTML. Al escribir `style="códico css"` no es necesario poner el selector ya que se entiende que el elemento afectado es aquel en el que está ubicado el estilo. Un ejemplo sería:

```
<p style="color:green;">Hola</p>
```

En el ejemplo anterior, el texto del párrafo aparecerá en color rojo.

6.2 Estilo por ámbito

Consiste en introducir un elemento `style` con el atributo `scoped` dentro de un elemento al que queramos dar formato. Dentro del elemento style introduciremos las reglas de estilo y éste se aplicará a todos los elementos que haya dentro del elemento padre. Por ejemplo

```
<hgroup>
    <style scoped>
    h1 {
         font-family: Arial;
         color: blue;
         font-size: 18pt;
         border-bottom: 2px outset gray;
    }
    h2 {
         font-family: Arial;
         color:red;
         font-size: 14pt;
         font-weight: normal;
    }
```

```
    </style>

    <h1>Introducción a la Informática</h1>
    <h2>Concepto de información</h2>
</hgroup>
```

En el ejemplo anterior vemos un elemento `hgroup` dentro del cual hay un elemento `style` con el atributo vacío `scoped`. El elemento `style` sirve para definir las reglas de formato, en este ejemplo se han definido dos reglas para los elementos `h1` y `h2`. Pues bien, el formato definido en estas reglas se aplicarán a todos los elementos `h1` y `h2` que haya dentro de `hgroup`.

En este ejemplo hemos definido que los elementos `h1` aparecan con fuente Arial tamaño 18 puntos, color azul y con un borde inferior resaltado de 2 píxeles. Los elementos `h2` apareceran con fuente Arial tamaño 14 puntos, color rojo y texto no negrita. El resultado sería el siguiente

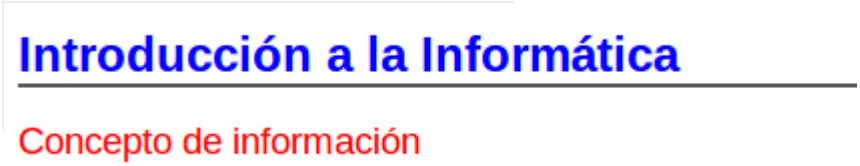


Figura 19.- Formato mediante estilo en ámbito

6.3 Hoja de estilo interna

Suelen usarse cuando se pretende aplicar el estilo sólo a la página donde se ubica. El código de la hoja de estilo interna se encontrará entre en un elemento style ubicado dentro de la cabecera (head).

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8"/>
        <title>Desarrollo web estático</title>
        <style type="text/css">
            body {
                font-family: Arial;
                font-size: 12pt;
            }
        </style>
    </head>
    <body>
…
    </body>
</html>
```

Como se puede apreciar el formato de los elementos de la página están agrupados al principio del todo.

6.4 Hoja de estilo externa

Se usan para dar un aspecto común a varias páginas de un sitio web. Todos los estilos se definen en una hoja de estilo externa para, posteriormente, hacer que todas las páginas del sitio web la importen. Así se obtendrá un estilo uniforme a todo el sitio (ya que todas las páginas web del sitio comparten una misma hoja de estilo), ahorrando al desarrollador esfuerzos mediante la técnica llamada reutilización de código.

Para crear una hoja de estilo externa se escribe su código en un documento de texto plano y se guarda con la extensión `.css`. Posteriormente se vincula a la página web mediante la inclusión de la siguiente etiqueta en la cabecera del documento HTML en cuestión:

```
<link rel="stylesheet" href="archivo.css">
```

El atributo `href` indica la URL del archivo css de donde se tomará el formato a aplicar.

En el siguiente ejemplo, se puede observar el código de una página HTML y vinculada a una hoja de estilo externa (`estilo. css`). Dado que sólo se halla el nombre de la hoja pero no su URL, se puede deducir que la hoja de estilo y la página HTML se encuentran en la misma carpeta.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Ejemplo</title>
        <link rel="stylesheet" href="estilo.css" >
    </head>
    <body>
    Contenido de la página
    </body>
<html>
```

Veamos ahora otra manera de importar una declaración externa de estilos CSS: `@import url("archivo_a_importar.css")`, que se utiliza para importar unas declaraciones de estilos guardadas en la URL que se indica entre paréntesis.

Se debe incluir en la declaración de estilos global a una página, es decir en el elemento `style` que se colocan en la cabecera del documento.

Es importante señalar que la sentencia de importación del archivo CSS se debe escribir en la primera línea de la declaración de estilos, algo parecido al código siguiente.

```
<style type="text/css">
@import url ("estilo.css");
body{
   background-color: Lightblue;
}
</style>
```

El funcionamiento es el mismo que si escribiésemos todo el fichero a importar dentro

de las etiquetas de los estilos, con la salvedad de que, si redefinimos dentro del elemento `style` estilos que están ya definidos en el archivo externo, los que se aplicarán serán los que hayamos redefinido.

Así, en el ejemplo anterior, aunque hubiésemos definido en `estilo.css` un color de fondo para la página, el color que prevalecería sería el definido a continuación de la importación, azul claro. La diferencia entre este tipo de importación del tipo y la que hemos visto anteriormente es que `@import url ("estilo.css")` se suele utilizar cuando hay unas pautas básicas en el trabajo con los estilos (que se definen en un archivo a importar) y unos estilos específicos para cada página, que se definen a continuación, dentro del elemento `style` , como es el caso del ejemplo visto anteriormente.

7 Las reglas de estilo

Una hoja de estilo se compone de una o varias reglas de estilo, que son las declaraciones de los formatos que adoptará el elemento para el cual va destinada.

Las reglas de estilo tienen dos componentes: el selector, que seleccionará los elementos sobre los que actuará la regla, y la declaración, que establece las propiedades y valores a aplicar sobre los elementos seleccionados. Así, una regla de estilo adopta la siguiente sintaxis:

```
selector { propiedadl: valor1; }
```

En el caso de declararse más de una propiedad/valor, éstas deben estar separadas por punto y coma, como por ejemplo:

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

En el caso de que sólo se quiera realizar una declaración con una propiedad/valor, no es obligatorio insertar al final el punto y coma, aunque se suele hacer. La siguiente imagen define las parte de la regla con un ejemplo.

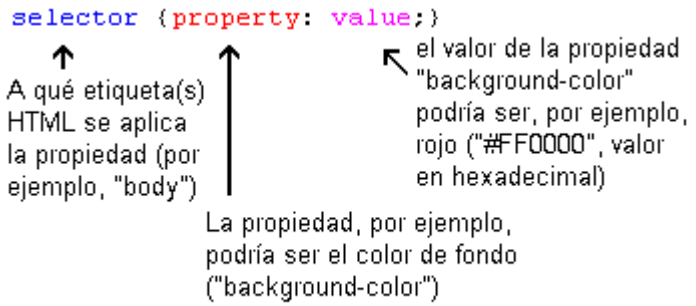


Figura 20.- Estructura de una regla

Por ejemplo, veamos la siguiente regla

En la imagen anterior se ha definido una regla. El selector es el elemento `h1`

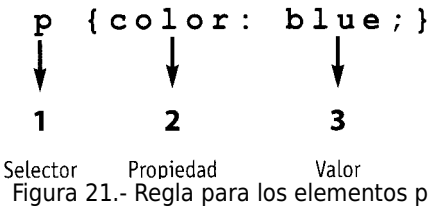


Figura 21.- Regla para los elementos p

(encabezado de nivel 1). Sólo tiene un par propiedad:valor que en este caso es `color:black` lo que significa que los encabezados de nivel 1 tendrán color negro. En el siguiente ejemplo se define una regla que aplica el color azul al texto de un elemento de

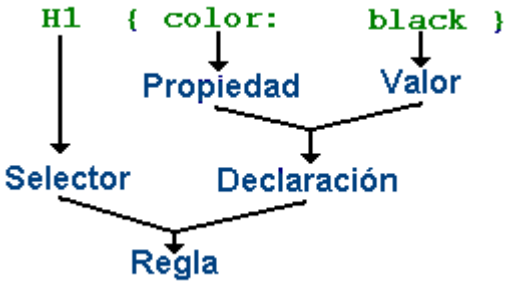


Figura 22.- Regla de formato para H1

párrafo.

7.1 El selector

El primero y más importante de los elementos que componen una regla de estilo es el selector. Sin él, el navegador web no sabría a qué elementos debe aplicarle el formato que a continuación viene definido entre llaves. Es decir, el selector determina los elementos sobre los que se aplica una regla de estilo. Por ejemplo, si se quieren formatear los elementos de párrafo con el color rojo, se tiene que crear un selector que identifique a dichos elementos:

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

Esto provocaría que todos los elementos `p` de la página afectada por la regla de estilo, adquiriesen el color rojo.

Puede ocurrir sin embargo, que sólo se desee dar un estilo determinado a unos cuantos elementos `p` y no a todos. En este caso, se deben usar unos selectores más complicados que especifiquen o seleccionen qué elementos serán formateados.

7.2 Propiedades y valores

Las propiedades es la parte de una regla que hace referencia al formato a aplicar a los elementos indicados por el selector. Existen muchas propiedades CSS para dar formato a todos los elementos que podemos introducir en una página web. Además, las propiedades toman un conjunto válido de valores específicos para cada propiedad. Sin embargo, hay varias propieades que hacen referencia al tamaño de los elementos que pueden tomar un valor expresado en diferentes unidades. También, las propiedades que permiten establecer

el color de primer plano y de fondo disponen de varias formas para especificar este color. Vamos a ver en detalle estos dos tipos de valores.

7.2.1 Unidades de medida

Existen muchas propiedades que establecen el tamaño de algo. Este algo puede ser la fuente del texto, la altura de una cabecera, la anchura de una capa, el interlineado de un párrafo, etc. CSS incluye varias medidas para establecer una propiedad de tamaño y son las siguientes:

Unidad	Descripción
%	Porcentaje. Hace referencia a un tanto por ciento.
in	Pulgada.
cm	Centimétros.
mm	Milímetros.
em	Un valor 1em es igual al tamaño actual de la fuente del elemento. 2em significa el doble del tamaño actual de fuente. Por ejemplo, si un elemento es visualizado con una fuente de 12 puntos, entonces 2em es 24 puntos. El valor em es una unidad muy útil en CSS, ya que puede adaptarse automáticamente a la fuente que el lector usa.
ex	Un ex es relativa a la altura de la letra x.
pt	Punto. Un punto es 1/72 de pulgada.
pc	Pica. Una pica son 12 puntos.
px	Pixel. Un píxel es un punto de la pantalla.

7.2.2 Valores de color

Para especificar un color también disponemos de varios métodos lo suficientemente flexibles como para poder representar cualquier color. Un color se puede especificar de cinco formas:

La primera es mediante una palabra, la cual se corresponde con el nombre de cada color en inglés. La siguiente imagen muestra 16 colores básicos

Figura 23.- Colores básicos

Esto es una lista básica, existen más colores que pueden especificarse mediante una palabra. Una tabla con todos estos colores puede verse en http://en.wikipedia.org/wiki/X11_color_names. Por ejemplo, la siguiente regla establece el color de fondo de un párrafo al valor `teal`.

```
p {
    background-color: teal;
}
```

Otra forma de indicar un color es mediante la función rgb. Esta función consiste en definir un color indicando la cantidad de color rojo, verde y azul que se debe mezclar para obtener ese color. Cada componente de color se debe indicar mediante un ńumero entre 0 y 255. Por ejemplo

```
p {
    background-color: rgb(128, 128, 128);
}
```

En el ejemplo anterior estamos poniendo un color resulatdo de mezclar el valor 128 de rojo, 128 de verde y 128 de azul. Al tener todos el mismo valor, el resultado es gris. Si ponemos un componente con valor máximo y los otros con valor mínimo, entonces el color resultado sería el del componente establecido a valor máximo. Por ejemplo

```
p {
    background-color: rgb(0, 255, 0);
}
```

En este caso tenemos que el segundo componente (verde) es 255 y el resto es 0, por tanto el color resultado será verde. El siguiente ejemplo establecería un color azul.

```
p {
    background-color: rgb(0, 0, 255);
}
```

Los componentes de un color mediante la función rgb también se pueden poner en porcentajes, siendo 0 el 0% y 255 el 100%. El ejemplo anterior equivaldría al siguiente

```
p {
    background-color: rgb(0, 0, 100%);
}
```

Otra forma de especificar el color es mediante el modelo RGB hexadecimal. Es parecido al anterior en el sentido de que hay que especificar un número para cada componente de color, pero en este caso estos números se representan en el sistema hexadecimal.

El sistema hexadecimal emplea los símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. Además de los dígitos decimales utiliza las seis primeras letras del abecedario, pero teniendo en cuenta que no son letras, representan números. Así , A en hexadecimal equivale a 10 en el sistema decimal, B al 11 y asi sucesivamente, hasta F que equivale a 15.

Para indicar un color usando este sistema hay que emplear seis dígitos, dos por cada componente, y precedido por el símbolo #. Por ejemplo

```
p {
    background-color: #FFB18C;
}
```

En el ejemplo anterior `FF` es el componente de rojo, es decir el valor máximo al tener los dos dígitos hexadecimales al valor máximo. El componente verde es `B1` y el azul `8C`. El resultado final será la mezcla de estos tres componentes en la medida establecida.

Este sistema se puede compactar utilizando un sólo dígito por componente de color, es decir, el valor serían tres dígitos hexadecimales.

```
p {
    background-color: #FFC;
}
```

Este método se emplea cuando los dos dígitos de cada componente de color son iguales. El ejemplo anterior equivale al siguiente empleando seis dígitos, dos por componente de color

```
p {
    background-color: #FFFFCC;
}
```

La siguiente forma de indicar el color es empleando los colores del sistema, que consiste en utilizar los colores actuales del sistema operativo. El color se indica mediante una palabra clave que hace referencia a algún elemento del sistema operativo. Por ejemplo, `ActiveBorder` es el color del borde de la ventana activa. En http://www.w3.org/TR/CSS21/ui.html#system-colors se puede encontrar una lista completa de estos colores.

El siguiente ejemplo define una regla para ver el cuerpo de la página web con el color de fondo de la ventana y el color en primer plano de la ventana.

```
p {
    background-color: Window;
    color: WindowText;
}
```

Finalmente, disponemos de los colores *web safe* que es una paleta de 216 colores y que hacen referencia a los colores que un desarrollador web podía utilizar y tener seguridad de que la pantalla lo mostraría indendientemente del navegador y sistema operativo utilizado. Esta paleta surgio en los 90, cuando muchos monitores no eran capaces de mostrar más de 256 colores. La lista completa de colores se puede ver en http://en.wikipedia.org/wiki/Websafe#Web-safe_colors.

7.2.3 URL's

Algunas propiedades necesitan un valor que es un recurso en la web. Por ejemplo, si queremos colocar una imagen de fondo debemos indicar esta imagen mediante una URL ya que en un documento web se puede utilizar cualquier recurso que está publicado en Internet.

Para indicar una URL hay que usar la función `url` con la siguiente sintaxis.

```
url(“URL”);
```

Como podemos ver recibe un argumento o parámetro que consiste en la URL del recurso que pretendemos cargar. Esta URL sigue el formato que se vio en un capítulo anterior. Su formato sería así

```
protocolo://nombre_servidor_web/ruta_carpetas/recurso_web
```

Por ejemplo, una URL típica podría ser http://www.apuntesdigitales.es/recursos/publico/documento.html.

Sin embargo, podemos formar una URL de dos formas:

- ✔ URL absoluta.- Significa que la URL tiene todos sus componentes y por tanto hay que indicar el protocolo, servidor, ruta y recurso. Esta URL comienza siempre por http:// y puede hacer referencia a cualquier recurso ubicado en cualquier servidor conectado a Internet.
- ✔ URL relativa.-Significa que la URL solo contiene la ruta de carpetas y el nombre del recurso (archivo) al que hace referencia. En este caso, la URL hace referencia a un recurso ubicado en el mismo servidor donde se encuentra el archivo que contiene esta URL.

Para un mismo recurso, una URL absoluta solo hay una, URL's relativas hay muchas, porque dependerá de la carpeta actual donde nos encontremos. Tomando como punto de partida la anterior URL de ejemplo. Si la raíz del servidor está en `C:\inetpub\wwwroot`,

entonces el archivo `documento.html` está almacenado en la carpeta `C:\inetpub\wwwroot\recursos\publico`.

Si dentro de este archivo hubiera una propiedad CSS que necesitara de una URL para cargar el archivo `imagen_fondo.jpg` que se encuentra en la carpeta `C:\inetpub\wwwroot\recursos\publico` habría que poner la función `url` e indicar la URL de esta imagen de la siguiente manera

```
url(“http://www.apuntesdigitales.es/recursos/publico/
imagen_fondo.jpg”)
```

Como podemos observar hemos formado esta URL con todos sus componentes. Sin embargo, `documento.html` e `imagen_fondo.jpg` están en el mismo servidor, con lo que podríamos haber puesto la siguiente URL relativa.

```
url(“imagen_fondo.jpg”)
```

Como podemos ver, ahora solamente ha quedado el nombre de archivo y han desaparecido el protocolo, nombre de servidor y ruta. Esto es así porque en las URL's relativas siempre se toma como servidor y ruta por defecto la que tiene el documento HTML o CSS que invoca la URL.

Ahora bien, supongamos que `imagen_fondo.jpg` se encuentra en la carpeta `C:\inetpub\wwwroot\recursos\publico\imagenes`. En este caso la URL relativa sería la siguiente

```
url(“imagenes/imagen_fondo.jpg”)
```

Se sigue partiendo de la carpeta `C:\inetpub\wwwroot\recursos\publico` en la está ubicado el archivo `documento.html`, pero ahora el recurso al que se pretende acceder está en la carpeta `imagenes` y por tanto hay que indicarlo en la URL.

Para formar URL's relativas disponemos de los caracteres `.` Y `..` los cuales cumplen la misma misión que en las referencias de archivos en el sistema operativo Linux o en el símbolo del sistema de Windows. El `.` hace referencia a la carpeta actual, mientras que el `..` hace referencia a la carpeta anterior en la jerarquía. El anterior ejemplo es equivalente a este

```
url(“./imagenes/imagen_fondo.jpg”)
```

En este caso el primer punto hace referencia a la carpeta actual, donde está el `documento.html` que invoca esta URL, dentro de la cual está la carpeta `imagenes` y, finalmente, dentro de ésta el archivo `imagen_fondo.jpg`.

Supongamos ahora que la carpeta imágenes está en `C:\inetpub\wwwroot\recursos` que es la carpeta anterior a `publico`, donde se encuentra `documento.html`. Si dentro de éste hacemos referencia al recurso `imagen_fondo.jpg` con una URL relativa habría que indicarlo así

```
url(“../imagenes/imagen_fondo.jpg”)
```

Los dos puntos indican ahora que hay que subir a la carpeta padre (`recursos`) de la actual (`publico`) y dentro de esta ir a la carpeta `imagenes` para cargar el archivo `imagen_fondo.jpg`.

Generalmente, el uso de `.` y `..` suele ser al principio de la URL relativa. El primero se podría omitir, pero también habría que omitir la `/` que lo separa de la siguiente carpeta.

7.3 Herencia

Hay propiedades CSS que afectan a los elementos definidos por el selector y también a los descendientes de esos elementos. Es decir, La propiedad es heredada por los descendientes. En este caso, los elementos que heredan estas propiedades se ven afectados por ellos de manera implícita. Por ejemplo

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8"/>
        <title>Desarrollo web estático</title>
        <style type="text/css">
            body {
                font-family: Arial;
                font-size: 12pt;
                color: blue;
            }
            h1 {
                font-size: 18pt;
            }
            p {
                color: gray;
            }
        </style>
    </head>
    <body>
    <h1>Introducción a CSS</h1>
    <p>
    CSS es un lenguaje de formateado de documentos web.
    </p>
    </body>
</html>
```

Primero tenemos una regla para el cuerpo del documento que establece la fuente en Arial, tamaño 12 y color azul. Estas propiedades se heredan para cualquier elemento dentro de body. La herencia se puede anular simplemente especificando un valor para la propiedad heredada.

En el ejemplo anterior, el elemento `p` tiene el color de texto gris, el cual anula el color azul que se heredó del `body`. Sin embargo, en el elemento `p` no ha definido la propiedad `font-family`, por tanto, hereda la que se definió en `body`. Por tanto los párrafos de este

documento tendrán texto con fuente Arial.

Hay algunas propiedades que no se heredan. En el caso de que se quisiera forzar la herencia de una propiedad habría que definirla con el valor especial `inherit`, el cual indica que el valor de la propiedad es el definido en el elemento donde está contenido el actual.

7.4 Introducir comentarios

Cuando se crea una hoja de estilo, es conveniente añadir comentarios, especialmente si es extensa y con numerosas líneas de código. De este modo es más fácil recordar, cuando se vuelva a editar el código, qué es lo que hace cada parte. Cuando se quieren buscar líneas de código que se desean modificar o reutilizar, bastará con leer los comentarios para identificar rápidamente el fragmento deseado.

Para añadir comentarios a las hojas de estilo, tanto internas como externas, se usa la combinación de caracteres `/*` para comenzar un comentario y este acaba con la misma combinación de caracteres pero en orden inverso, es decir, `*/`.

```
/* Esta regla de estilo define la fuente de la clase prov */
.prov {
    font-size: 9px;
    color: #FFE2A8; /* Este color queda mejor que el #FF0044 */
}
```

Entre esos caracteres se pueden introducir retornos de carro y todo aquello que se considere oportuno excepto los caracteres `*/` ya que lo que viniese después sería considerado como código CSS, provocando errores.

Otra práctica muy común y recomendable es la de comentar reglas de código que se quieren eliminar. Cuando no se quiere utilizar una regla de estilo pero se desea conservarla por si se decide usarla más adelante, se pueden poner comentarios en torno a ella. Así no será tenida en cuenta por encontrarse dentro del comentario, pero se podrá volver a «poner en funcionamiento» fácilmente eliminando los caracteres de comentarios. Por ejemplo:

```
/* Esta regla de estilo no la voy a usar por ahora
.prov {
        font-size: 9px;
        color:#FFE2A8;
        text-decoration: none;
} */
```

En este ejemplo, se ha puesto una regla de estilo dentro de un comentario; si se quisiese utilizar en un futuro, bastaría con eliminar dicho comentario.

7.5 Las nuevas propiedades de la version 3

El objetivo inicial de CSS, separar el contenido de la forma, se cumplió ya con las primeras especificaciones del lenguaje. Sin embargo, el objetivo de ofrecer un control total a los diseñadores sobre los elementos de la página ha sido más difícil de cubrir. Las especificaciones anteriores del lenguaje tenían muchas utilidades para aplicar estilos a las

webs, pero los desarrolladores aun continúan usando trucos diversos para conseguir efectos tan comunes o tan deseados como los bordes redondeados o el sombreado de elementos en la página.

La versión 1 de CSS ya significó un avance considerable a la hora de diseñar páginas web, aportando mucho mayor control de los elementos de la página. Pero como todavía quedaron muchas otras cosas que los diseñadores deseaban hacer, pero que CSS no permitía especificar, éstos debían hacer uso de trucos para el diseño. Lo peor de esos trucos es que muchas veces implica alterar el contenido de la página para incorporar nuevas etiquetas HTML que permitan aplicar estilos de una manera más elaborada. Dada la necesidad de cambiar el contenido, para alterar al diseño y hacer cosas que CSS no permitía, se estaba dando al traste con alguno de los objetivos para los que CSS fue creado, que era el separar por completo el contenido de la forma.

CSS 2 incorporó algunas novedades interesantes, que hoy ya utilizamos habitualmente, pero CSS 3 todavía avanza un poco más en la dirección, de aportar más control sobre los elementos de la página.

Así pues, la novedad más importante que aporta CSS 3, de cara a los desarrolladores de webs, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos o hacks, que a menudo complicaban el código de las web.

Sin embargo hay un pequeño problema y es que no todas las propiedades CSS3 son reconocibles por todos los navegadores. Las compañias que desarrollan navegadores están incorporando poco a poco las propiedades de la versión 3, y es posible que para algunas de ellas en lugar de usar el nombre especificado en el estándar del W3C, haya que añadir un prefijo a éste para que sea reconocible. Este prefijo es diferente y hace referencia al navegador. Estos son:

Prefijo	Navegador	Ejemplo
-moz-	Mozilla	`-moz-border-top-colors`
-webkit-	Chrome y Safari	`-webkit-border-top-colors`
-o-	Opera	`-o-border-top-colors`
-ms-	Internet Explorer	`-ms-border-top-colors`

Si una propiedad de CSS3 definida no aparece en la pantalla, entonces podemos llegar a suponer que en realidad no es reconocida por el navegador o que en el caso de que lo sea tendremos que indicar su prefijo. Si a lo largo del texto alguna propiedad está acompañada por (v3), entonces estaremos indicando que es de la versión 3. Si además, fuera necesario indicar prefijo de navegador para que sea reconocida, entonces haremos referencia a ella con sus prefijos correspondientes.

Además, para asegurarnos que la propiedad en cuestión se va a visualizar correctamente en nuestra página, cada vez que haya que indicar alguna de estas propiedades lo haremos con el nombre estándar y los demás con los prefijos. Por ejemplo

```
border-top-colors: #e00 #c30 #c50;
-moz-border-top-colors: #e00 #c30 #c50;
-webkit-border-top-colors: #e00 #c30 #c50;
-o-border-top-colors: #e00 #c30 #c50;
-ms-border-top-colors: #e00 #c30 #c50;
```

En la web http://www.w3schools.com/css3/default.asp disponemos de una referencia completa de las nuevas propiedades CSS3, los navegadores que la soportan y a partir de que versión del navegador.

8 Colores y fondos con CSS

En esta apartado aprenderemos a aplicar colores de primer plano y de fondo a los documentos web. Examinaremos también métodos avanzados para posicionar y controlar imágenes de fondo.

8.1 Color en primer plano

La propiedad `color` describe el color de primer plano de un elemento. Por ejemplo, supongamos que queremos que todos los encabezados de nivel 1 (elemento `h1`) de un documento aparezcan con color rojo oscuro. La siguiente regla de formato establece el color de los elementos `h1` como rojo oscuro.

```
h1 {
    color: #880000;
}
```

Los colores se pueden introducir como valores hexadecimales, como en el ejemplo anterior: `#880000`; o se pueden usar los nombres de los colores: `Darkred` (rojo oscuro), o bien usando la función `rgb(136, 0, 0)`. Debemos recordar que valores bajos de un componente de color obtenemos ese componente oscurecido, mientras que valores altos son colores más claros.

8.2 Color de fondo

La propiedad `background-color` describe el color de fondo de los elementos. El elemento `body` contiene todo el contenido de un documento HTML. Así pues, para cambiar el color de fondo de una página, la propiedad `background-color` debería aplicarse al elemento `body`.

También se pueden aplicar colores de fondo a otros elementos, entre ellos, a los encabezados y al texto. En el ejemplo que sigue se establecen dos reglas de formato para los elementos `body` y `h1` en las que establecemos diferente color de fondo para el cuerpo y para el encabezado.

```
body {
    background-color: #FFCC66;
}
```

```
h1 {
    color: #990000;
    background-color: #FC9804;
}
```

Fíjate cómo hemos aplicado dos propiedades a `h1` separándolas por medio de un punto y coma.

8.3 Imagen de fondo

Para definir una imagen de fondo disponemos de varias propiedades que nos permiten controlar no solo la carga de una imagen de fondo, sino también su posición, su tamaño, etc.

La propiedad `background-image` se usa para insertar una imagen de fondo. Para poner una imagen de de fondo de una página web, aplica sencillamente la propiedad `background-image` al elemento `body` y especifica la localización de la imagen mediante la función url().

```
body {
    background-color: #FFCC66;
    background-image: url("butterfly.gif");
}

H1 {
    color: #990000;
    background-color: #FC9804;
}
```

Fíjate cómo hemos especificado la localización de la imagen: `url("butterfly.gif")`. Es una URL relativa y solo se ha indicado el nombre del archivo de imagen. Esto significa que la imagen está en la misma carpeta que la hoja de estilo. También puedes hacer referencia a imágenes en otras carpetas usando por ejemplo `url("../imagenes/butterfly.gif")` o incluso imágenes de Internet si indicas la URL absoluta del fichero de imagen: `url("http://www.html.net/butterfly.gif")`.

Es muy difícil que el tamaño de una imagen se ajuste exactamente al tamaño del área de visualización del navegador. ¿Qué ocurriría si la imagen es más pequeña? En este caso la imagen se repite horizontalmente y verticalmente llenando toda la pantalla en mosaico. La propiedad `background-repeat` controla este comportamiento. La tabla siguiente resume los cuatro valores diferentes para esta propiedad

Valor	Descripción
`repeat-x`	La imagen se repite en el eje horizontal
`repeat-y`	La imagen se repite en el eje vertical
`repeat`	La imagen se repite en el ejer horizontal y vertical (por defecto)
`no-repeat`	La imagen no se repite

Por ejemplo, para evitar que se repita un imagen de fondo, el código que tendríamos que usar sería el siguiente:

```
body {
    background-color: #FFCC66;
    background-image: url("butterfly.gif");
    background-repeat: no-repeat;
}
```

Vemos que también hemos puesto un color de fondo. Si la imagen de fondo no cubre toda la pantalla, el resto se verá con el color de fondo.

La propiedad `background-attachment` especifica si una imagen está fija o se desplaza con el elemento contenedor. Una imagen de fondo fija no se moverá con el texto cuando el lector se desplace por la página, mientras que una imagen de fondo no fija se desplazará con el texto de la página web.

La tabla siguiente resume los dos valores posibles para la propiedad `background-attachment`.

Valor	Descripción
`scroll`	La imagen se desplaza con la página, no está fija
`fixed`	La imagen está fija

Por ejemplo, el siguiente código fijará la imagen de fondo.

```
body {
    background-color: #FFCC66;
    background-image: url("butterfly.gif");
    background-repeat: no-repeat;
    background-attachment: fixed;
}
```

Por defecto, una imagen de fondo que no se repite se posiciona en la esquina superior izquierda de la pantalla. La propiedad `background-position` te permitirá cambiar este valor por defecto y posicionar la imagen de fondo en cualquier lugar de la pantalla que quieras.

Hay muchas formas diferentes de establecer los valores de la propiedad `background-position`. Sin embargo, todas ellas se formatean como un conjunto de coordenadas. Por ejemplo, el valor `100px 200px` posiciona la imagen de fondo a 100 píxeles del margen izquierdo y a 200 píxeles del margen superior del la ventana del navegador.

Las coordenadas se pueden indicar como porcentajes del ancho de la pantalla, como unidades fijas (píxeles, centímetros, etc.) o puedes usar las palabras `top` (superior), `bottom` (inferior), `center` (centro), `left` (izquierda) y `right` (derecha). El modelo siguiente ilustra cómo funciona el sistema:

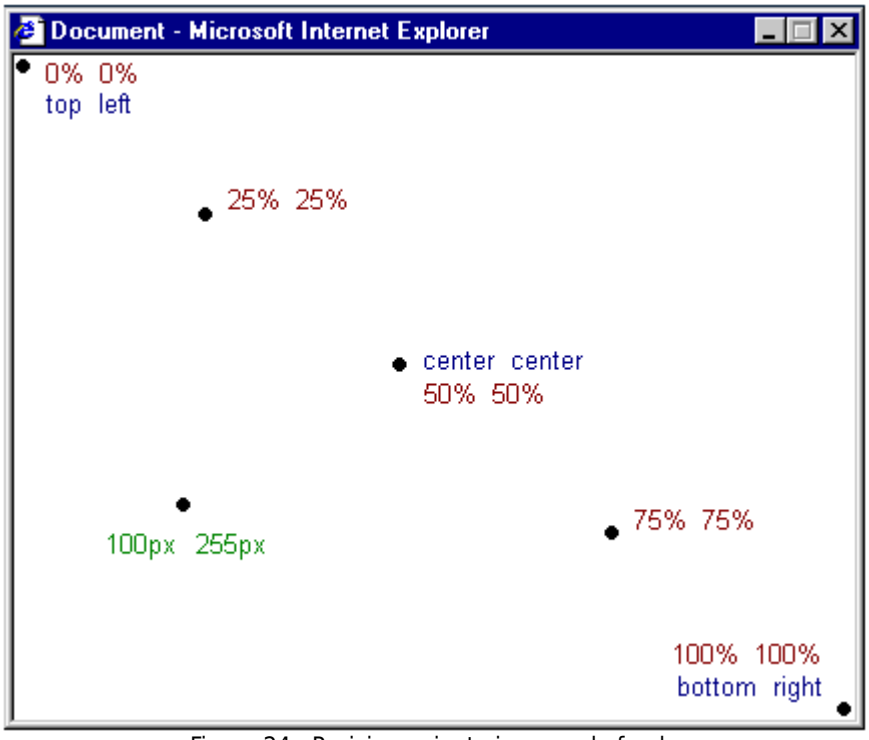


Figura 24.- Posicionamiento imagen de fondo

El ejemplo de código siguiente posiciona la imagen de fondo en la esquina inferior derecha:

```
body {
    background-color: #FFCC66;
    background-image: url("butterfly.gif");
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: right bottom;
}
```

La propiedad `background-size` (CSS3) establece el tamaño de la imagen de fondo. Esta propiedad permite reutilizar una misma imagen de fondo en diferentes contextos.

Puedes especificar el tamaño en pixeles o en porcentajes. Si se establece el tamaño como porcentaje, el tamaño es relativo al ancho y alto del elemento. Por ejemplo

```
body {
    background-color: #FFCC66;
    background-image: url("butterfly.gif");
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: right bottom;
    background-size: 80px 60px;
}
```

Si hubieramos puesto lo siguiente

```
background-size: 100% 100%;
```

Entonces, la imagen de fondo cubrirá toda la página web, al estar la regla definida para el elemento `body`. Si estuviera en otro contexto, entonces cubriría toda la superficie el elemento en cuestión. Establecer esta propiedad puede suponer deformar la imagen.

Existen una propiedad que indica desde que punto la imagen de fondo comienza a cubrir el elemento. Esta propiedad es `background-origin`. Una imagen de fondo puede comenzar en el borde (`border-box`), en el relleno (`padding-box`) o en el contenido (`content-box`). La siguiente imagen ilustra este concepto.

Figura 25.- Propiedad background-origin

En un capítulo posterior se verá en detalla el modelo de caja, sin el cuál puede resultar difícil entender esta propiedad.

Similar a la anterior tenemos la propiedad `background-clip`, pero en este caso se aplica al color de fondo. Tiene los mismos valores que `background-origin` y con el mismo significado.

8.3.1 Propiedad abreviada background

La propiedad `background` es una forma abreviada de todas las propiedades de fondo listadas a lo largo de esta lección. Con la propiedad `background` se pueden comprimir varias propiedades, y así escribir una hoja de estilo de forma más abreviada, lo que facilitará su lectura.

Por ejemplo, observa estas cinco líneas de código:

```
background-color: #FFCC66;
background-image: url("butterfly.gif");
background-repeat: no-repeat;
background-attachment: fixed;
background-position: right bottom;
```

Usando `background` se puede lograr el mismo resultado con una única línea de código:

```
background: #FFCC66 url("butterfly.gif") no-repeat fixed right
bottom;
```

El orden en que deben aparecer las propiedades individuales es el siguiente: `[background-color] | [background-image] | [background-repeat]| [background-attachment] | [background-position]`

Si se omite alguna propiedad, de forma automática ésta se establecerá con su valor por defecto. Por ejemplo, si se omiten las propiedades `background-attachment` y `background-position` del ejemplo anterior, quedando el código de la siguiente manera:

```
background: #FFCC66 url("butterfly.gif") no-repeat;
```

Estas dos propiedades que no se especifican se establecerían, sin más, con sus valores por defecto, que, como ya sabes, son `scroll` y `top left`.

8.4 Gradientes como fondo

Consiste en poner de fondo un color que se degrada desde un color inicial a un color final. Disponemos de dos funciones que se asignarían a la propiedad `background-image` o a la propiedad abreviada `background` para obtener el gradiente. Estas son:

- ✔ `linear-gradient` para obtener un gradiente lineal.
- ✔ `radial-gradient` para obtener un gradiente radial.

La primera tiene la siguiente sintaxis

```
linear-gradient( [<punto> || <angulo>,]? <parada>, <parada> [,
<parada>]* )
```

Los parámetros a utilizar son los siguientes

`<punto> || <angulo>`

El origen desde donde comenzará el degradado y/o el ángulo de disposición del gradiente de color. Podemos decir que el degradado comience desde arriba, abajo o desde una esquina cualquiera. Por defecto los degradados serán distribuidos en un gradiente en línea recta, pero además podemos indicar un ángulo distinto con el que se vaya produciendo el gradiente de color.

`<parada>, <parada> [,<parada>]*`

Los colores que deben utilizarse en el degradado, separados por comas. Se pueden poner tantos como se quieran. Además, si lo deseamos, podemos definir las paradas de color que consiste en un color seguido de un valor en píxeles o porcentaje indicando el lugar desde donde debe empezar el gradiente del color.

Veamos algunos ejemplos. Supongamos establecemos alternativamente una regla para body con las siguientes propiedades.

```
background: -webkit-linear-gradient(orange, pink);
background: -moz-linear-gradient(orange, pink);
background: -o-linear-gradient(orange, pink);
background: linear-gradient(orange, pink);

background: -webkit-linear-gradient(top left, #fff, #f66);
background: -moz-linear-gradient(top left, #fff, #f66);
background: -o-linear-gradient(top left, #fff, #f66);
background: linear-gradient(top left, #fff, #f66);
```

```
background: -webkit-linear-gradient(180deg, #f0f, #f66);
background: -moz-linear-gradient(180deg, #f0f, #f66);
background: -o-linear-gradient(180deg, #f0f, #f66);
background: linear-gradient(180deg, #f0f, #f66);

background: -webkit-linear-gradient(#00f 50%, #000);
background: -moz-linear-gradient(#00f 50%, #000);
background: -o-linear-gradient(#00f 50%, #000);
background: linear-gradient(#00f 50%, #000);

background: -webkit-linear-gradient(45deg, #66f, #f80, #ffc);
background: -moz-linear-gradient(45deg, #66f, #f80, #ffc);
background: -o-linear-gradient(45deg, #66f, #f80, #ffc);
background: linear-gradient(45deg, #66f, #f80, #ffc);

background: -webkit-linear-gradient(45deg, #66f 10%, #f80 30%,
#ffc 60%);
background: -moz-linear-gradient(45deg, #66f 10%, #f80 30%, #ffc
60%);
background: -o-linear-gradient(45deg, #66f 10%, #f80 30%, #ffc
60%);
background: linear-gradient(45deg, #66f 10%, #f80 30%, #ffc
60%);

background: -webkit-linear-gradient(45deg, #66f 160px, #f80
180px, #ffc);
background: -moz-linear-gradient(45deg, #66f 160px, #f80 180px,
#ffc);
background: -o-linear-gradient(45deg, #66f 160px, #f80 180px,
#ffc);
background: linear-gradient(45deg, #66f 160px, #f80 180px,
#ffc);

background: -webkit-linear-gradient(left, #f00, #f80, #ff0,
#0f0, #00f, #60f, #c0f);
background: -moz-linear-gradient(left, #f00, #f80, #ff0, #0f0,
#00f, #60f, #c0f);
background: -o-linear-gradient(left, #f00, #f80, #ff0, #0f0,
#00f, #60f, #c0f);
background: linear-gradient(left, #f00, #f80, #ff0, #0f0, #00f,
#60f, #c0f);
```

El resultado de todos estos ejemplos sería el siguiente

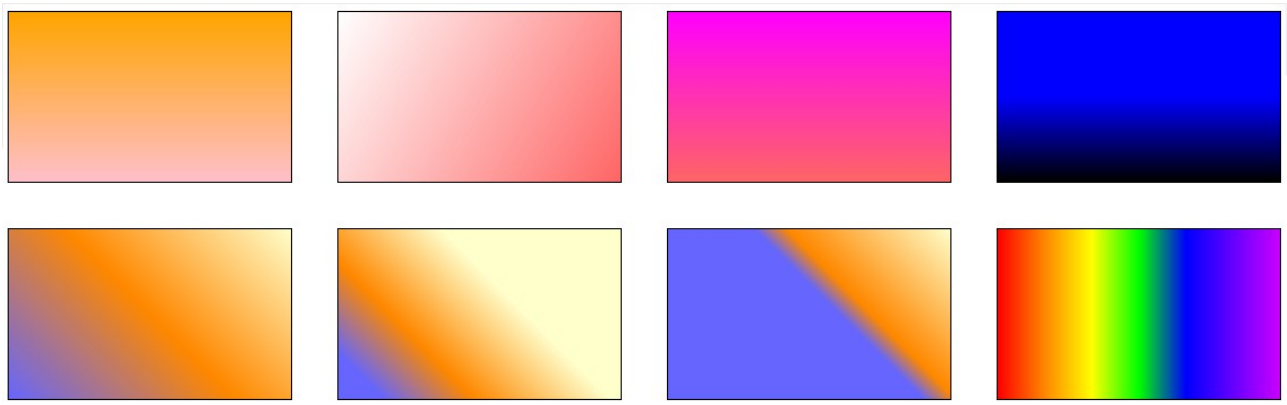


Figura 26.- Gradientes

Para la función `radial-gradient` tenemos la siguiente sintaxis

```
radial-gradient( [<posición> || <ángulo>,]? [<forma> ||
<tamaño>,]? <parada>, <parada>[, <parada>]* )
```

Los parámetros a utilizar son los siguientes:

`<posicion> || <ángulo>`

Los degradados radiales comienzan en un punto cualquiera del fondo de un elemento y se extienden hacia fuera de ese punto con formas circulares o de elipse. Luego, para definirlos, necesitaremos una forma de especificar dicho punto de inicio del degradado. El punto se especifica con una o dos coordenadas, que pueden tener distintas unidades CSS. Si se omite, se entiende que el degradado tiene que comenzar en el punto central del fondo del elemento.

`<forma>`

La forma puede ser circular o elipse, para lo cual especificamos las palabras "circle" o "ellipse".

`<tamaño>`

El tamaño lo expresamos con otra serie de palabras clave, que indican hasta donde debe crecer el círculo o elipse: closest-side | closest-corner | farthest-side | farthest-corner | contain | cover. Por ejemplo, closest-side indica que el círculo o elipse debe crecer hasta el lado más cercano. La palabra farthest-corner indicaría que debe crecer hasta la esquina más lejana. Contain sería lo mismo que decir closest-side y cover sinónimo de farthest-corner.

De manera alternativa a especificar la forma y dimensión del degradado, podemos indicar un par de medidas en cualquier unidad CSS o porcentajes. Esas medidas se utilizarían para generar un círculo o una elipse del tamaño deseado para nuestro gradiente. La primera medida sería para la anchura de la elipse y la segunda sería para la altura (si ambas son iguales se mostraría una forma circular en el degradado. Si son distintas, sería una elipse. El tamaño debe ser siempre positivo.

`<parada>, <parada> [, <parada>]*`

Colores que se deseen para el degradado, separados por comas, con la posibilidad de indicar las paradas de color que se deseen.

Veamos algunos ejemplos para Firefox,

```
background: -webkit-radial-gradient(#0f0, #06f);
background: -moz-radial-gradient(#0f0, #06f);
background: radial-gradient(#0f0, #06f);

background: -webkit-radial-gradient(top left, #fff, #f66);
background: -moz-radial-gradient(top left, #fff, #f66);
background: radial-gradient(top left, #fff, #f66);

background: -webkit-radial-gradient(200px 30px, #f0f, #000);
background: -moz-radial-gradient(200px 30px, #f0f, #000);
background: radial-gradient(200px 30px, #f0f, #000);

background: -webkit-radial-gradient(center, #00f, #000 50%);
background: -moz-radial-gradient(center, #00f, #000 50%);
background: radial-gradient(center, #00f, #000 50%);

background: -webkit-radial-gradient(circle, #66f, #f80, #ffc);
background: -moz-radial-gradient(circle, #66f, #f80, #ffc);
background: radial-gradient(circle, #66f, #f80, #ffc);

background: -webkit-radial-gradient(ellipse cover, #66f, #f80,
#ffc);
background: -moz-radial-gradient(ellipse cover, #66f, #f80,
#ffc);
background: radial-gradient(ellipse cover, #66f, #f80, #ffc);

background: -webkit-radial-gradient(10%, ellipse closest-side,
#66f 60%, #f80 85%, #ffc);
background: -moz-radial-gradient(10%, ellipse closest-side, #66f
60%, #f80 85%, #ffc);
background: radial-gradient(10%, ellipse closest-side, #66f 60%,
#f80 85%, #ffc);

background: -webkit-radial-gradient(10%, ellipse farthest-
corner, #66f 60%, #f80 85%, #ffc);
background: -moz-radial-gradient(10%, ellipse farthest-corner,
#66f 60%, #f80 85%, #ffc);
background: radial-gradient(10%, ellipse farthest-corner, #66f
60%, #f80 85%, #ffc);
```

El resultado consecutivo de estos ocho degradados radiales sería el siguiente

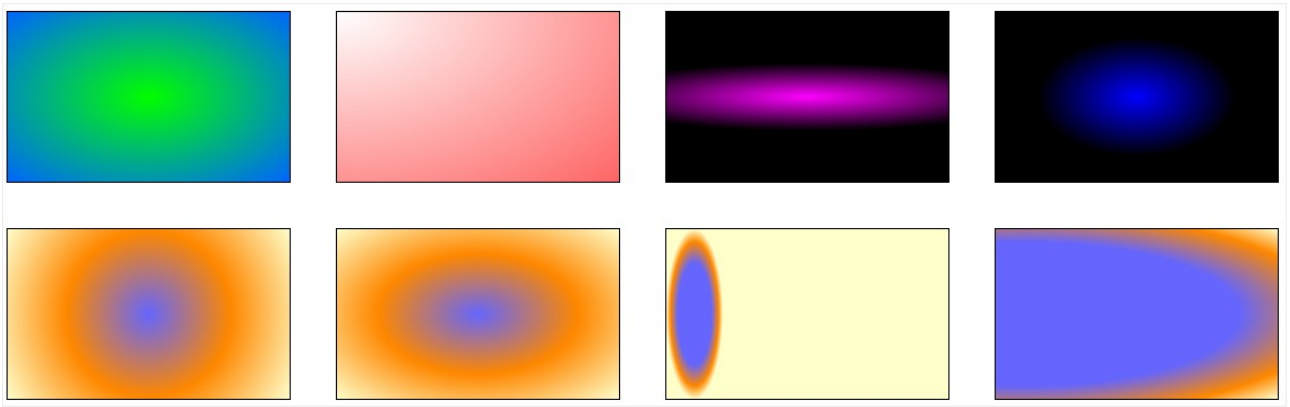


Figura 27.- Degradados radiales en Firefox

Los fondos radiales son soportados en poca medida en estos momentos en los navegadores. Todos tienen al menos algún detalle que todavía tienen que pulir. De momento ni Opera ni IE9 los muestran en ningún caso. Chrome no tiene posibilidad de hacer degradados con forma de elipse y Firefox todavía no implementa la alternativa al tamaño y forma, comentada anteriormente, para la definición del tamaño del gradiente por medio de los valores de anchura y altura.

9 Formato de texto con CSS

9.1 Fuentes

Una fuente tipográfica es un tipo de letra junto con su posición y conformación almacenados en un archivo fuente. Un archivo fuente puede incluir como se visualizan e imprimen los caracteres en minúscula, mayúscula, símbolos de puntuación, símbolos matemáticos y números.

Los archivos fuente son independientes de las aplicaciones que los usan. Estos se instalan en una determinada carpeta del sistema operativo para que estén disponibles en todos los programas que los necesiten. En la actualidad, prácticamente la totalidad de programas dentro de las interfaces gráficas de usuario admiten el uso de tipografía digital, en mayor o menor escala. Los formatos de tipografía digital más comunes son:

- ✔ TTF (TrueType Font).- Es el tipo de fuente más comúnmente utilizado ya que es un formato estándar de tipo de letra escalable.
- ✔ OTF (OpenType Font).- Originada por Microsoft, esta basada en el formato ttf pero con algunas mejoras en su estructura principal. En la actualidad se encuentra en proceso de convertirse en un estándar abierto.
- ✔ EOT (Embedded Open Type).- Creada igualmente por Microsoft, por lo general su utilidad radica enla posibilidad de convertir una fuente true type a EOT y así poder ser visualizada en el navegador Internet Explorer. Ha sido rechazada como estándar por el W3C.
- ✔ WOFF (*Web Open Font Format*).- Se desarrolló en 2009 y está en el proceso de normalización como una recomendación por el Grupo de Trabajo de Tipos de Letra

Web del W3C. Su objetivo es permitir la distribución de fuentes desde un servidor web a un equipo cliente en una red. Se espera que WOFF sea el único e interoperable formato de fuente soportado por todos los navegadores.

- ✔ SVG (Scalable Vector Graphics).- Se trata de una especificación para describir gráficos vectoriales bidimensionales en formato XML. En un archivo SVG pueden ser incrustadas varias fuentes diferentes, por eso cada una de ellas llevara un identificador único que se especifica en el momento de crear el archivo SVG.

9.1.1 Especificar fuentes en los elementos

Cuando especificamos una fuente para algún elemento de la página deberemos asegurarnos de que dicha fuente es interpretable y visible por los navegadores. Desgraciadamente, un navegador emplea las fuentes que hay instaladas en el sistema del cliente, es decir, en el ordenador de cada usuario. Por tanto, el desarrollador web no puede controlar ni saber que fuentes tiene instaladas el cliente y cuáles puede utilizar.

La propiedad `font-family` se usa para establecer una lista ordenada de fuentes que se usarán para mostrar un elemento de una página web. Si la primera fuente de la lista no está instalada en el ordenador desde el que se accede al sitio, se seguirá probando con la siguiente fuente hasta encontrar una fuente apropiada. Si ninguna fuente de la lista está presente en el sistema del cliente, entonces el navegador mostrará una fuente por defecto.

Para clasificar las fuentes se usan dos tipos de nombres: nombres de una familia y familias genéricas. La diferencia se puede ilustrar así:

Figura 28.- Familias de fuentes y fuentes

Al listar fuentes para el sitio web, por supuesto se empieza por la preferida, seguida

ésta de algunas fuentes alternativas. Se recomienda completar la lista con una familia de fuentes genérica. Así, al menos, la página se mostrará usando una fuente de la misma familia si ninguna de las especificadas están disponibles.

Un ejemplo de lista de fuentes por orden de prioridad podría tener este aspecto:

```
h1 {font-family: arial, verdana, sans-serif;}
h2 {font-family: "Times New Roman", serif;}
```

Los encabezados de nivel 1 se mostrarán usando la fuente `Arial`. Si esta fuente no está instalada en el ordenador de usuario, se usará en su lugar la fuente `Verdana`. Si ambas fuentes no están disponibles, se usará una fuente de la familia `sans-serif` para mostrarlos. Fíjate cómo el nombre de fuente `Times New Roman` contiene espacios y, por lo tanto, se encierra entre comillas.

El tamaño de la fuente se establece por medio de la propiedad `font-size`. A la hora de describir el tamaño de las fuentes podemos emplear las unidades de medida que vimos anteriormente. Como ejemplo, podemos incluir:

```
h1 {font-size: 30px;}
h2 {font-size: 12pt;}
h3 {font-size: 120%;}
p  {font-size: 1em;}
```

La propiedad `font-style` define el estilo para la fuente elegida. Puede ser `normal` o `italic` (`oblique` es sinónimo). En el ejemplo que sigue, todos los encabezados de nivel 2 aparecerán en cursiva.

```
H2 {
    font-family: "Times New Roman", serif;
    font-style: italic;
}
```

La propiedad `font-variant` se usa para elegir entre las variantes `normal` o `small-caps` (versales) de una fuente. La fuente a la que se aplica el valor `small-caps` es una fuente que usa letras en mayúscula y la letra inicial tiene un tamaño algo mayor que las escritas específicamente en minúscula. Por ejemplo, la siguiente regla se emplea para poner los títulos de nivel 2 con una fuente en versales.

```
H2 {
    font-family: "Times New Roman", serif;
    font-variant: small-caps;
}
```

La propiedad `font-weight` describe qué intensidad o "peso" en negrita debería tener la fuente. Toda fuente puede tener los valores `normal` o `bold` (negrita). También se pueden especificar números entre 100 y 900, de cien en cien, para describir el peso de dicha fuente.

```
p  {font-family: arial, verdana, sans-serif;}
h2 {
    font-family: arial, verdana, sans-serif;
    font-weight: bold;
}
```

Con la propiedad `font-stretch` podemos establecer el ancho de la fuente. Dispone de los siguientes valores: `wider`, `narrower`, `ultra-condensed`, `extra-condensed`, `condensed`, `semi-condensed`, `normal`, `semi-expanded`, `expanded`, `extra-expanded`, `ultra-expanded` e `inherit`. Estos valores son suficientemente autoexplicativos. Por ejemplo, podemos poner un párrafo con una fuente condensada con la siguiente regla.

```
p  {
    font-family: arial, verdana, sans-serif;
    font-stretch: condensed;
}
```

9.1.2 Propiedad abreviada font

Al igual que ocurría con las propiedades para establecer el fondo de los elementos, disponemos de una propiedad abreviada para especificar el formato de la fuente. Con la propiedad `font` es posible incluir todas las propiedades diferentes relativas a una fuente en una única propiedad. Por ejemplo, supongamos la siguiente regla que usamos para describir las propiedades de fuente para el elemento de párrafo.

```
p {
    font-style: italic;
    font-weight: bold;
    font-size: 30px;
    font-family: arial, sans-serif;
}
```

Usando la propiedad abreviada, el código se puede simplicar así:

```
P { font: italic bold 30px arial, sans-serif; }
```

El orden de los valores para la propiedad `font` es:

```
font-style | font-variant | font-weight | font-size | font-
family
```

9.1.3 Fuentes en el servidor

A la hora de incluir una fuente para algún elemento de nuestra página podemos estar seguros de que si empleamos algunas de las familias estándar, estos elementos se visualizarán perfectamente en la pantalla de cualquier navegador. Sin embargo, la fuente de estas familias genéricas no son muy atractivas y si queremos valor añadido a nuestro sitio web con diseños atractivos tendremos que emplear fuentes poco habituales en los sistemas operativos de hoy en día, lo que aumenta la probabilidad de que no se visualicen en la

pantalla de los navegadores. Es por ello que el abanico de fuentes que podíamos utilizar, con la garantía que funcionasen bien en la mayoría de los visitantes, está reducido a las típicas Arial, Verdana, Times New Roman y poco más.

Es evidente que en esta situación, el desarrollador web pierde control en el diseño y desarrollo de un sitio web al no poder definir con total seguridad la fuente que se empleará para visualizarlo. La solución a este problema es la regla `@font-face`.

Esta regla nos permite definir en nuestra hoja de estilos una fuente, para lo cual necesitaremos el archivo de fuente. En la práctica supone poder utilizar en nuestro sitio web cualquier fuente que nosotros deseemos. Esta regla surgió en la versión 2 de CSS pero hasta la versión 3 no empieza a funcionar y prosperar. Actualmente, IE9 solamente soporta formatos de fuente EOT, mientras que Safari, Firefox, Chrome y Opera soportan tanto fuentes TTF como OTF. Todos los navegadores soportan fuentes WOFF. IE8 y versiones anteriores no soportan la regla `@font-face`.

Su sintaxis es la siguiente:

```
@font-face{
    font-family:    <nombre_fuente>;
    src:            <URL_archivo_fuente> format(<formato>)
                    [,<URL_archivo_fuente> format(<formato>)]*;
    [font-stretch:  <ancho>;
    [font-weight:   <peso>];
    [font-style:    <estilo>];
}
```

Con esto definimos el tipo de letra y su ubicación en nuestro servidor. Si queremos utilizar dicha fuente tan solo tenemos que asignarla con la propiedad `font-family` en las reglas de estilo que deseemos empleando el nombre de fuente que hemos indicado en la propiedad `font-family` de la regla `@font-face`.

Con la propiedad `src` especificamos la URL al archivo de fuente y el formato. Podemos especificar varios separados por comas. Es recomendable utilizar varios archivos fuente que abarquen todos los formatos reconocibles por los navegadores. Así, tendremos la seguridad de que el elemento configurado con dicha fuente se verá perfectamente.

Veamos un ejemplo. Supongamos que hemos obtenido en Internet los archivos de fuente de la fuente Open Sans Light (la empleada para escribir este documento). Para emplearla utilizaríamos esta regla

```
@font-face {
font-family: "Open Sans Light";
src:     url("/fuentes/OpenSans-Light.eot?") format("eot"),
         url("/fuentes/OpenSans-Light.woff") format("woff"),
         url("/fuentes/OpenSans-Light.ttf") format("truetype"),
         url("/fuentes/OpenSans-Light.svg#OpenSans-Light")
format("svg"),
    url("/fuentes/OpenSans-Light.otf") format("opentype")
font-weight:normal;
```

```
font-style:normal;
}
```

Hemos definido la fuente con nombre `Open Sans Light` cuyos archivos fuente se puede obtener por parte del navegador en la carpeta `fuentes` de la raíz del sitio web. La fuente tiene un peso y un estilo normal. Nótese que para la definición de la fuente en format SVG ha habido que añadir el identificador de la fuente ya que un archivo SVG puede incluir varias fuentes.

Ahora podemos emplear esta fuente en cualquier elemento del sitio web. En el siguiente ejemplo se asigna a los elementos de párrafo.

```
p { font-family: "Open Sans Light"; }
```

9.1.4 ¿Donde puedo conseguir nuevas fuentes?

En Internet disponemos de muchos sitios donde podemos descargarnos las fuentes que más nos gusten para componer nuestros sitios web. Uno de los más populares es Open Font Library (www.openfontlibrary.org) en la que existen miles de fuentes para descargar e instalar en nuestro sistema.

Otro es www.fonts2u.com donde además de los archivos fuente en diferentes formatos nos ofrecen el código CSS con la regla `@font-face` para añadirla a nuestra hoja de estilo.

En www.fontsquirrel.com también disponemos de un amplio catálogo de fuentes para descargar y utilizar, además de un generador online de la regla `@font-face`.

Además, en estos sitios también podemos subir un archivo fuente en un formato y que nos lo convierta a otro, para así poder disponer de todos los formatos necesarios a la hora de que los elementos con dicha fuente puedan visualizarse en todos los navegadores y/o dispositivos.

También existe otra posibilidad que facilita la tarea de los desarrolladores web y consiste en utilizar el repositorio de fuentes de Google. En www.google.com/fonts disponemos de un catálogo que crece lenta pero progresivamente. Lo mejor de usar las fuentes de Google es que solamente tendremos que añadir un elemento `link` a la cabecera de la página web y posteriormente solamente hay que usar la fuente elegida. No es necesario generar la regla `@font-face` ni tener los archivos fuente ya que el `link` de Google los incluye.

Una vez elegida la fuente hacemos clic en la sección *Quick use* y ahí nos indicará el elemento `link` que tenemos que añadir a nuestra web y el nombre de la fuente a utilizar en las reglas CSS.

9.2 Propiedades para elementos textuales

En este apartado presentaremos las propiedades que ofrece CSS a la hora de añadir presentación al texto.

La propiedad `text-indent` permite añadir un toque de elegancia a los párrafos de texto al aplicar sangría a la primera línea de dicho párrafo. En el ejemplo siguiente se ha aplicado una sangía de 30 píxeles a todos los párrafos.

```
p {
    text-indent: 30px;
}
```

La propiedad `text-align` se emplea para especificar la alineación del texto. Los valores posibles de esta propiedad son: `left` (texto alineado a la izquierda), `right` (texto alineado a la derecha), `center` (texto con alineación centrada) o `justify` (alineación justificada).

En el ejemplo que sigue, el texto de los encabezados de primer nivel se han alineado a la derecha, mientras que encabezados de segundo nivel aparecen centrados. Además, los párrafos están justificados:

```
h1 {
    text-align: right;
}
h2 {
    text-align: center;
}

p {
    text-align: justify;
}
```

La propiedad `text-decoration` permite añadir diferentes efectos al texto. Por ejemplo, se puede subrayar el texto, tacharlo o ponerle un subrayado superior. Los posibles valores son:

- ✔ `none`.- Define texto normal. Es el valor por defecto.
- ✔ `underline`.- Define una línea debajo del texto.
- ✔ `overline`.- Define una línea encima del texto.
- ✔ `line-through`.- Define una línea a través del texto.

En el ejemplo siguiente, el elemento `h1` aparecerá subrayado, el elemento `h2` aparecerá con un subrayado por encima del texto y el elemento `h3` tendrá el texto tachado.

```
h1 {
    text-decoration: underline;
}

h2 {
    text-decoration: overline;
}

h3 {
```

```
    text-decoration: line-through;
}
```

El espaciado entre los caracteres de texto se puede especificar usando la propiedad `letter-spacing`. El valor de esta propiedad corresponde, sencillamente, al ancho deseado. Por ejemplo, si queremos un espaciado de 3 píxeles entre los caracteres de un párrafo y 6 píxeles entre los caracteres de los encabezados de primer nivel, usaríamos el siguiente código:

```
h1 {
    letter-spacing: 6px;
}

p {
    letter-spacing: 3px;
}
```

La propiedad `text-transform` controla la escritura en mayúsculas de un texto sin importar cómo aparece el texto original en el código HTML. Puedes elegir entre los valores:

- ✔ `capitalize`.- Pone en mayúscula la primera letra de cada palabra.
- ✔ `uppercase`.- El texto aparece en mayúscula.
- ✔ `lowercase`.- El texto aparece en minúscula.

Por ejemplo, la siguiente regla define que los encabezados de nivel 1 aparezcan con la primera letra de cada palabra en mayúsculas y los de nivel 3 aparezcan en minúsculas.

```
h1 {
    text-transform: capitalize;
}

h3 {
    text-transform: lowercase;
}
```

Por último podemos aplicar una sobra al texto. La propiedad text-shadow permite aplicar una sombra al texto. Su sintaxis es la siguiente

```
text-shadow: h-shadow v-shadow difuminado color;
```

Los diferentes parámetros son:

- ✔ `h-shadow`, la posición de la sombra en horizontal. Si el valor es positivo es hacia la derecha, si es negativo hacia la izquierda.
- ✔ `v-shadow`, la posición de la sombra en vertical. Si el valor es positivo es hacia abajo y si es negativo hacia arriba.
- ✔ `difuminado`, aplica difuminado a la sombra. Este valor es una unidad de medida.

- ✔ `color`, color de la sombra.

Por ejemplo, la siguiente regla define una sombra desplazada 5 píxeles a la derecha y abajo con un difuminado de un píxel y color gris a los encabezados de nivel 1.

```
h1 {
    text-shadow: 5px 5px 1px gray;
}
```

10 Imágenes en documentos web

Una imagen digital es la representación de una imagen para que pueda ser tratada por un ordenador. Las imágenes digitales se pueden obtener de varias formas:

- ✔ Por medio de dispositivos de conversión analógica-digital como los escáneres y las cámaras digitales.
- ✔ Mediante programas informáticos, como por ejemplo realizando dibujos con el ratón o mediante un programa de renderización 2D.
- ✔ Dibujando con una tableta digitalizadora o gráfica, que es un periférico que permite al usuario introducir gráficos o dibujos a mano, tal como lo haría con lápiz y papel.

Existen dos tipos de imágenes digitales los mapas de bits y los gráficos vectoriales.

10.1 Gráficos vectoriales

Las imágenes vectoriales están compuestas por entidades geométricas simples: segmentos y polígonos básicamente (de hecho, una curva se reduce a una sucesión de segmentos). Cada una de estas entidades está definida matemáticamente por un grupo de parámetros (coordenadas inicial y final, grosor y color del contorno, color del relleno, etc.) Por compleja que pueda parecer una imagen, puede reducirse a una colección de entidades geométricas simples.

Al estar compuestas por entidades geométricas simples, las imágenes vectoriales se pueden cambiar de escala, para ampliarlas o reducirlas, sin que la imagen pierda calidad. Esta es su gran ventaja, porque proporcionan siempre imágenes de colores planos con contornos limpios, sin importar el tamaño al que se muestran.

10.2 Mapas de bits

Las imágenes de mapa de bits consiste en una representación matricial de la imagen. Esta es dividida en un arreglo bidimensional o matriz de filas por columnas. La intersección de una fila con una columna es un elemento de imagen o píxel. Cada uno de los píxeles de la imagen tiene asignado un número que representa el color de ese punto en la imagen. La sensación obtenida al ver la imagen formado por todos los píxeles juntos, cada uno con su color es el resultado de integrar visualmente, en la retina, las variaciones de color y luminosidad entre píxeles vecinos.

Las imágenes de mapa de bits, también llamadas bitmap, son la alternativa ideal para

reproducir objetos sutilmente iluminados y escenas con gran variación tonal. De hecho, es el tipo de imagen utilizado para la fotografía y el cine. Obviamente, la calidad de la imagen dependerá de la cantidad de píxeles utilizados para representarla.

Las imágenes bitmap no permiten el cambio de escala. Si aumentamos una imagen bitmap se perderá definición en la imagen ya que los píxeles aumentan de tamaño y los cuadraditos se aprecian más. Este efecto, que se conoce con el nombre de pixelado se hace más evidente en las líneas curvas y en las zonas en las que hay cambios bruscos de luminosidad.

Existen tres factores que influyen a la hora de representar una imagen bitmap: resolución, profundidad de color y dimensiones de la imagen.

10.2.1 Resolución

La resolución de una imagen es la cantidad de píxeles que la componen. Suele medirse en píxeles por pulgada (ppi) o píxeles por centímetro (pcm). Cuanto mayor es la resolución de una imagen más calidad tendrá su presentación pero, desgraciadamente, más espacio ocupará en el disco el archivo gráfico que la contiene.

Por ejemplo, una imagen con una resolución de 72 ppi, que es muy común en las páginas web, necesitará 5184 píxeles en cada pulgada cuadrada, que es un cuadrado de 2,54 centímetros de lado. Una resolución de 72 ppi es adecuada para imágenes que se muestran en el monitor de un ordenador. La nitidez de los detalles es suficiente y la reproducción de las distintas tonalidades es correcta. Sin embargo, podría ser insuficiente para una impresión en papel.

10.2.2 Profundidad de color

Cada uno de los píxeles de una imagen bitmap está coloreado con un color homogéneo. Así pues, el archivo que contiene los datos de la imagen debe contener la información del color de cada uno de los píxeles. ¿Cuántos bit se emplean para albergar esta información? Eso es lo que se conoce con el término profundidad de color de una imagen.

Profundidad de color es el número de bits utilizados para describir el color de cada pixel de la imagen. Es obvio que, cuanto mayor sea la profundidad de color de una imagen, más colores tendrá la paleta disponible y, por tanto, la representación de la realidad podrá hacerse con más matices, con colores más sutiles.

Por ejemplo, si sólo disponemos de 1 bit para describir el color de cada pixel, tan sólo podremos elegir entre dos colores: un color si el bit tiene el valor 0 (habitualmente negro) y otro color si el bit vale 1 (habitualmente blanco).

Si disponemos de 8 bit para describir el color de cada pixel, podremos elegir entre 256 colores, porque 2^8=256. Esta es una profundidad de color suficiente para las imágenes construidas en el modo denominado escala de grises, porque con 8 bits cada pixel puede adoptar un tono entre 256 valores posibles de gris, entre el negro absoluto (00000000) y el blanco absoluto (11111111).

Si los 8 bit disponibles para la profundidad deben designar colores, entonces se utiliza

una tabla con los 256 colores más frecuentes, que incluyen obviamente el negro, el blanco y varios tonos de gris, para componer la imagen. Cada una de las 256 combinaciones posibles de unos y ceros de los 8 bits es un índice que permite acceder a la tabla. Por eso, a este tipo de imágenes se les conoce como de color indexado., que es el más frecuente en la web.

Y así, cuanto mayor sea la profundidad se utilizará una cantidad mayor de colores para describir la imagen. En la tabla siguiente tienes el cálculo de los colores disponibles para cada profundidad:

Profundidad de color (bit)	Nº de colores
1	2
4	16
8	256
16	65536
32	4294967296

Una imagen bitmap de calidad está compuesta por varias capas: una para cada color básico (rojo, verde y azul, por ejemplo) y una para la luminosidad (de oscuro absoluto a luz absoluta). Por encima de 16 bits de profundidad, la descripción del color se divide por capas. Si la profundidad de color es de 16 bits, por ejemplo, se dedican 4 bits (128 niveles) a cada capa. Y si la profundidad es de 32 bits, cada capa utiliza 8 bits (256 niveles) para ajustar el color.

10.2.3 Dimensión de la imagen

Las dimensiones de una imagen se expresan, como es habitual, en cm o mm. Por ejemplo, una imagen de 10 x 15 cm medirá 10 cm de ancho y 15 cm de alto. A veces, sin embargo, los programas expresan el tamaño de una imagen en píxeles. Para calcular el tamaño de una imagen en píxeles basta con multiplicar las dimensiones lineales, en centímetros por ejemplo, por la resolución en píxeles por centímetro. Hay que poner atención para utilizar las mismas unidades de longitud.

10.2.4 El tamaño de archivo de imagen

El tamaño del archivo es una cifra, en bits o en bytes, que describe la cantidad de memoria necesaria para almacenar la información de la imagen en un soporte (disco duro, CD, tarjeta de memoria, etc). Y, como ya te imaginas, el tamaño del archivo dependerá de varios factores y, especialmente, de la resolución (R), las dimensiones de la imagen (Largo x Ancho) y la profundidad de color (P). Puedes calcular el tamaño de un archivo con la siguiente fórmula:

Tamaño = R^2*L*A*P

Por ejemplo, una imagen de 10 x 15 cm (3,94 x 5,91 pulgadas), con una resolución de 96 ppi (38 pcm) y una profundidad de color de 32 bits, tendrá un tamaño bruto de:

9216 x 3,94 x 5,91 x 32 = 6.857.144 bits = 858.393 Bytes = 838 KBytes

Recuerda que 1 byte son 8 bits y que 1 Kilobyte equivale a 1024 bytes.

10.3 Formato de archivo de imagen

Una vez creada nuestra imagen, ya sea capturada con la cámara o creada a mano, la guardamos en un archivo. El archivo, con un nombre y una extensión, no sólo contiene la información de cada pixel. Tiene también una cabecera en la que se guarda información destinada al programa encargado de abrir la imagen y mostrarla en el monitor.

Aunque, por regla general, los archivos vectoriales tienen tamaños mucho menores que los archivos bitmap, todos los archivos gráficos suelen tener tamaños muy grandes. Este gran consumo de espacio en disco hizo necesario el desarrollo de tecnologías capaces de comprimir archivos gráficos.

Cada sistema de compresión utiliza un algoritmo matemático propio para reducir la cantidad de bits necesarios para describir la imagen, y marca el archivo resultante con una extensión característica: bmp, wmf, jpg, gif, png, etcétera.

Algunos de estos algoritmos están patentados, son propiedad de una empresa, y hay que pagar por utilizarlos. Otros algoritmos, en cambio, son de dominio público y pueden utilizarse libremente. También se distinguen entre si por las pérdidas producidas en la información de la imagen durante el proceso de compresión. Así pues hay algoritmos con pérdidas y sin pérdidas. Veamos algunos de los formatos de compresión más utilizados:

- ✔ JPG.- Es un formato de compresión con pérdidas, pero que desecha en primer lugar la información no visible, por lo que las pérdidas apenas se notan. El algoritmo jpg está basado en el hecho de que el ojo humano percibe peor los cambios de color que las variaciones de luminosidad. jpg divide la información de la imagen en dos partes: color y luminosidad y las comprime por separado. Admite modos en escala de grises con una profundidad de 8 bits y en color hasta 24 bits. Permite la carga progresiva en un navegador, lo que lo ha convertido en el formato estándar en la web. No es un formato adecuado para imágenes con alto contraste de color. Además, hay que tener en cuenta que la compresión se produce automáticamente cada vez que se guarda el archivo, por lo que es aconsejable guardar en este formato una única vez, cuando la imagen esté ya terminada.
- ✔ GIF.- Es un formato que devuelve imágenes de tamaño muy reducido. Esa reducción se consigue indexando los colores, es decir, asimilándolos a uno de los 256 colores de su tabla. Su profundidad de color máxima, por tanto, es de 8 bits. El formato gif permite hacer algunas cosas curiosas: puede hacerse transparente uno de los colores indexados en la tabla, lo que permite suprimir fondos. También permite enlazar varias imágenes gif en una secuencia, lo que se conoce con el nombre gif animado. El pequeño tamaño de los archivos gif hizo que fuera el formato más extendido en los primeros tiempos de Internet. Pero su principal defecto consiste en que es un formato propietario (CompuServe Inc.), lo que ha provocado la aparición del formato libre png que, además, comprime mejor que gif.

- ✔ PNG.- Es el formato de más rápido crecimiento en la web, porque reúne lo mejor de jpg y gif. Se trata de un formato de compresión sin pérdidas, con una profundidad de color de 24 bits. Soporta hasta 256 niveles de transparencia, lo que permite fundir la imagen perfectamente con el fondo. Entre sus inconvenientes hay que citar que no soporta animaciones y que el tamaño de los archivos png, debido a la capa de transparencia, siempre es mayor que el de los archivos jpg.
- ✔ BMP.- Es un formato de compresión sin pérdidas. Admite cualquier tipo de resolución y una profundidad de color máxima de 24 bits. Es el formato nativo de Microsoft y se usa en todas sus aplicaciones (Windows, Office, etc.). Por esta razón es muy frecuente encontrar archivos bmp, pero su tasa de compresión es ridículamente baja. Entre los navegadores, sólo es soportado por Internet Explorer.

10.4 Incluir una imagen en la web

En HTML disponemos del elemento vacío `img` para introducir imágenes. Este elemento necesita el atributo `src` para indicar la URL de la imagen que va a mostrar. Por ejemplo, supongamos que tenemos la página web `documento.html` el cual reside en una carpeta donde también está el archivo `imagen.png` que vamos a introducir en el mismo. Habría que incluir el siguiente elemento

```
<img src="imagen.png"/>
```

El valor del atributo src tiene que ser la URL de la imagen lo que implica que podemos cargar en una página web cualquier imagen accesible en Internet. En el ejemplo anterior hemos utilizado una imagen cuyo archivo reside en la misma carpeta en la que reside la página web.

Si la imagen estuviera en el mismo servidor, pero en otra carpeta, podemos utilizar una URL relativa, como en el siguiente ejemplo

```
<img src="imagenes/imagen.png"/>
```

En el ejemplo anterior suponemos que en la carpeta donde reside la página web hay otra carpeta llamada `imagenes` que es donde se encuentra el archivo de imagen.

Si queremos también, podemos cargar una imagen que reside en otro servidor web, pero en esta ocasión necesitamos especificar una URL absoluta.

```
<img src="http://www.loquesea.com/publico/imagenes/imagen.png"/>
```

Un elemento de imagen es un elemento en línea, por lo que aparecerá en la página web con texto por delante y por detrás.

Además, es obligatorio incluir el atributo `alt` el cual contiene el texto que muestra la imagen en el caso de que no se cargue. Por ejemplo

```
<img src="imagen.png" alt="Logotipo de la empresa"/>
```

10.5 Formato de imagen

A la imagen podemos definirle el siguiente formato

10.5.1 Tamaño de la imagen

Para indicar el tamaño de la imagen podemos emplear los atributos `width` (anchura) y `height` (altura). A estos atributos podemos especificar valores en píxeles o en porcentajes. Por ejemplo

```
<img src="imagen.png" alt="Logotipo de la empresa" width="200"
height="400"/>
```

En el ejemplo anterior hemos establecido una anchura de 200 píxeles y una altura de 400 píxeles. También podríamos haber indicado porcentajes.

```
<img src="imagen.png" alt="Logotipo de la empresa" width="80%"
height="75%"/>
```

Cuando aumentamos el tamaño original de la imagen estamos perdiendo definición y la imagen no se apreciaría bien. Si empleamos porcentajes, aumentamos su tamaño cuando indicamos un valor por encima del 100%.

Si cambiamos el tamaño original de la imagen y especificamos la anchura y la altura seguramente estaremos provocando la deformación de la imagen. Así que lo mejor es indicar solamente una dimensión (ancho o alto) para evitarlo. Por lo general, si la imagen tiene una forma apaisada cambiamos el ancho, mientras que si tiene una orientación vertical, cambiamos el alto.

También disponemos de las propiedades CSS `width` y `height` para establecer el ancho y alto de una imagen con la ventaja de que podemos utilizar más unidades de medida para indicar el tamaño de la imagen. Si repetimos el primer ejemplo tendríamos lo siguiente utilizando estas propiedades.

```
<img src="imagen.png" alt="Logotipo de la empresa"
style="width:200px;height:400px;"/>
```

10.5.2 Alineación

Para alinear una imagen disponemos de la propiedad CSS `float`. Si el valor de la propiedad es `left`, entonces la imagen quedará a la izquierda y el texto subyacente rodeará la imagen por la derecha, mientras que si el valor es `right`, la imagen queda a la derecha y el texto la rodea por la izquierda.

Veamos el siguiente ejemplo

```
<p>En el párrafo de abajo hemos añadido una imagen flotando a la
derecha. El resultado es que la imagen queda a la derecha y el
texto la rodea por la izquierda.
</p>
```

```
<p>
<img src="logocss.gif" width="95" height="84"
style=”float:right;”/>
Para alinear una imagen disponemos de la propiedad CSS float. Si
el valor de la propiedad es left, entonces la imagen quedará a
la izquierda y el texto subyacente rodeará la imagen por la
derecha, mientras que si el valor es right, la imagen queda a la
derecha y el texto la rodea por la izquierda. La propiedad float
solamente permite alinear una imagen horizontalmente, es decir,
situarla a la izquierda o a la derecha, pero no arriba o abajo.
</p>
```

El resultado sería el siguiente

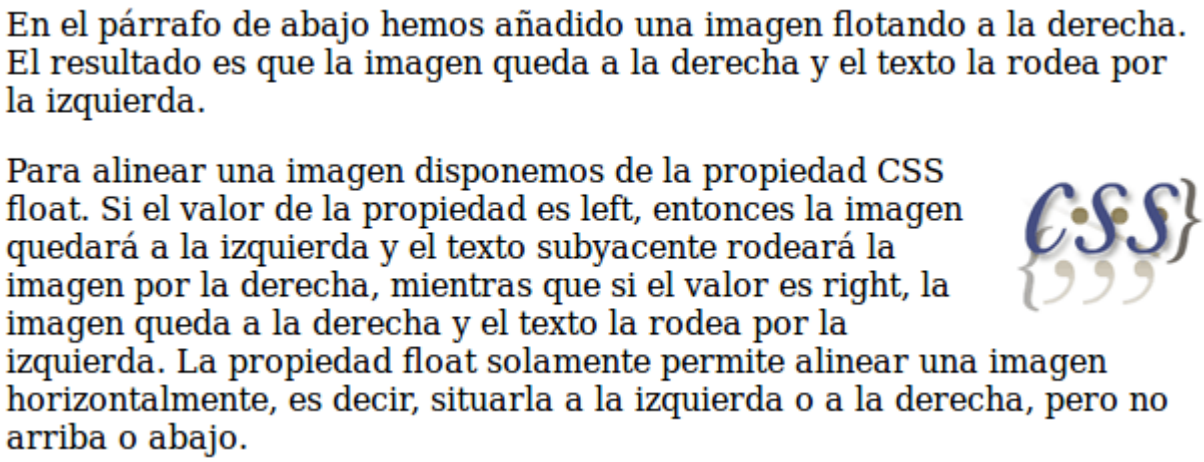


Figura 29.- Imagen floatando a la derecha

Si cambiamos el valor de la propiedad a `left`, entonces la imagen aparecerá a la izquierda.

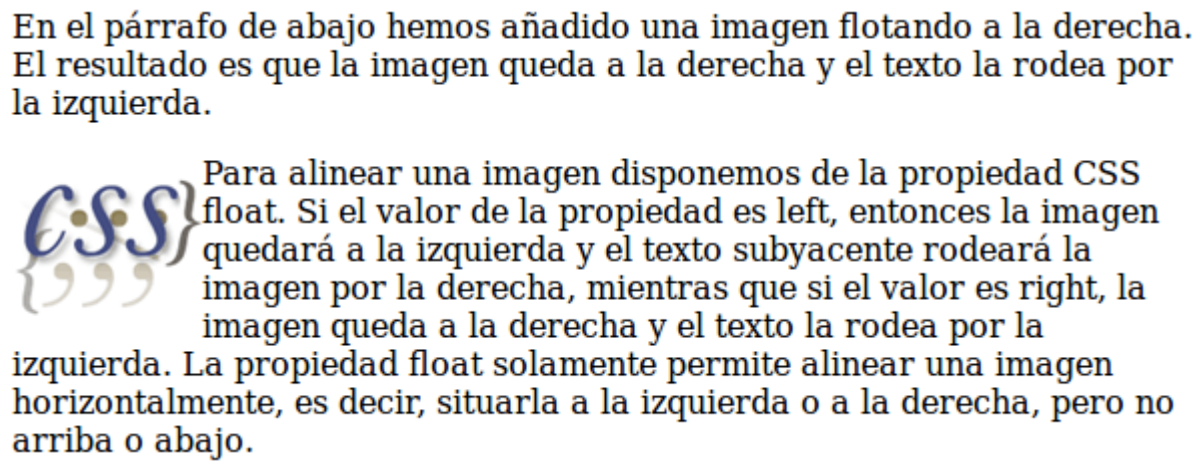


Figura 30.- Imagen flotando a la izquierda

10.5.3 Elementos figure y figcaption

Generalmente, en libros y revistas, las imágenes que se emplean para ilustrar un texto suelen tener un título. Antes de HTML5 no había forma de incluir semánticamente este tipo de contenido. Ahora disponemos de los elementos `figure` y `figcaption`.

Según la especificación del W3C el elemento `figure` representa una unidad de contenido, y opcionalmente con un título, que puede ser movido a otra parte sin que el significado del texto del documento se vea afectado. El elemento `figcaption` representa un título o leyenda para un elemento `figure`.

El elemento `figcaption` es opcional y puede aparecer antes o después del contenido en el elemento `figure`. Solo puede haber un elemento `figcaption` dentro de un elemento `figure`, aunque el elemento `figure` puede contener otros elementos `figure`. Veamos el siguiente ejemplo.

```
<figure>
  <img src="./orangutan.jpg" alt="Un bebe orangután colgado de
una cuerda"/>
</figure>
```

En el ejemplo anterior hemos utilizado un elemento `figure` para una imagen. Si añadimos un elemento figcaption tendremos lo siguientes.

```
<figure>
    <img src="./orangutan.jpg" alt="Un bebe orangután colgado de
una cuerda"/>
    <figcaption>Un bebe orangután colgado de una cuerda en la
selva de Borneo,<br> por <span style="font-
style:italic;">Richard Clark</span>
    </figcaption>
</figure>
```

El resultado sería el siguiente

Un bebe orangután colgado de una cuerda en la selva de Borneo, por *Richard Clark*

Figura 31.- Elementos figure y figcaption

11 Enlaces

Este capítulo introduce el enlace, el elemento esencial para construir documentos de hipertexto. Un enlace es una conexión de un recurso web con otro. Aunque el concepto es muy simple, el enlace ha sido la principal causa del éxito de la Web.

Un enlace tiene dos puntos finales, llamadas anclas, y una dirección. En enlace comienza en un ancla origen y apunta a un ancla destino, el cual puede ser cualquier recurso web: una imagen, un video clip, un archivo de sonido, un programa, otro documento HTML o un elemento dentro de un documento HTML.

Los recursos a los que apuntan los enlaces están en la web almacenados en forma de archivos. Todos los nombres de archivos y sus extensiones a las que se hacen referencia en los hiperenlaces, deben de ponerse en su totalidad en minúsculas, para evitar posibles errores, ya que muchos servidores web residen en sistemas operativos que son sensibles a mayúsculas y minúsculas al nombrar archivos. Como normal general se debe escribir todos los nombres de enlaces, directorios y archivos, en minúsculas, debido principalmente a esta característica de la mayoría de los servidores HTML de Internet.

Por el mismo motivo, en Internet la mayoría de las referencias a directorios se emplean con la barra / y no con la barra \, ya que las máquinas UNIX/Linux antes nombradas, hacen referencias a directorios con la barra de división. Debemos de tener estas 2 consideraciones muy en cuenta a la hora de hacer hiperenlaces desde nuestro documento HTML, ya que sino son tenidas en cuenta, nos llevará a tener sucesivos errores al intentar enlazar.

Una vez hayamos hecho nuestra página web, puede que queramos que nuestros visitantes tengan referencias de otras páginas que nosotros creamos convenientes que visiten, bien porque tienen que ver con el tema que se trata en nuestra páginas, bien porque consideremos que pueden ser interesantes.

Otra razón puede ser que tengamos un documento web demasiado grande y queramos dividirlo en otros más pequeños y conectarlos mediante enlaces.

Las funcionalidades hipertexto se basan en el concepto de enlace. Un enlace es una referencia a un documento HTML, o a cualquier otro objeto, expresada por medio de un formato, universalmente aceptado. Este permite que el visualizador, cuando detecta que se requiere ”saltar” a un documento determinado, sea capaz de identificarlo y obtenerlo, de forma que nos sea visible a nosotros en nuestra pantalla del navegador.

De cara al usuario, los hiperenlaces aparecerán como texto en otro color, realzado o subrayado, incluso como veremos pueden ser imágenes.

11.1 Incluir enlaces en un documento web

Para incluir un hiperenlace, se usa el elemento `a`. El texto que se halle entre la etiqueta de apertura y la de cierre será sensible al ratón, lo que indica que una pulsación con el ratón sobre ella, o una pulsación con la barra espaciadora cuando el enlace tiene el foco, nos lleva al destino del enlace.

Este elemento posee el atributo `href`, que sirve para indicar la URL de destino del enlace. Entre las etiquetas de apertura y cierre podemos incluir texto o una imagen descriptivas del enlace al que estamos haciendo referencia. Por ejemplo,

```
<a href=“http://www.google.es”>Buscador Google</a>
```

Aparecería el texto `Buscador Google` subrayado con distinto color al resto del texto. Al colocar el ratón sobre ella, aparecería en la barra de estado del navegador, la dirección URL hacia la que apunta (en este caso `http://www.google.es`). Una vez pinchado el enlace, el navegador se comporta como si hubieramos escrito la URL en la barra de dirección del navegador y pulsado la tecla Intro.

En lugar de texto, podríamos incluir una imagen que actuara de hiperenlace:

```
<a href=“http://www.google.es”><img src=“google.gif”
alt=”Google”/></a>
```

En este caso la imagen, al aparecer como enlace a otro documento HTML, estaría bordeada con un marco azul, de similar tonalidad al que subrayaba anteriormente al enlace de tipo texto. Esto, nos indicaría que esa imagen está actuando de enlace a otra página. Al igual que como ocurría con el texto, al colocar al ratón sobre cualquier área de la imagen, en la barra de estado nos indica la dirección URL del enlace a la que apunta la imagen.

11.2 Destino del enlace

El elemento a dispone del atributo target para indicar donde se visualizará el destino del enlace. Este atributo puede tener los siguientes valores

Valor de target	Descripción
`_blank`	Abre una nueva ventana o pestaña en el navegador y visualiza la URL

	de destino del enlace.
_parent	
_self	
_top	
<nombre-de-marco>	

11.3 Marcadores

Por defecto, cuando enlazamos con una página web, esta se presenta en la pantalla del navegador desde el principio. Sin embargo, existe la posibilidad de enlazar con un punto concreto dentro de la página web.

Dentro de las páginas web podemos definir marcadores o puntos del documento donde podemos establecer el destino de un enlace. Esto permite que al enlazar con una página web vayamos a un marcador concreto e incluso podemos enlazar con un marcador dentro del mismo documento donde está el enlace.

El atributo id puede usarse para crear un marcador dentro del documento web.

```
<a id="nombre_del_marcador">...</a>
```

Todo el contenido del elemento a ya sea texto, imagen o lo que queramos poner como símbolo del enlace, es ahora el marcador, de forma que si dentro del documento hay un enlace a este marcador, el navegador nos lleve hasta ella.

Supongamos el siguiente ejemplo. El archivo documento.html contiene el siguiente marcador.

```
<a id="consejos">Consejos Útiles</a>
```

Ahora podemos crear un enlace en otro documento que enlaza directamente con este marcador de la siguiente manera.

```
<a
href="http://www.apuntesdigitales.es/publico/documento.html#cons
ejos>Veamos una consejos últiles</a>
```

Como podemos apreciar hemos añadido #consejos a la URL del enlace. Esto indica que tiene que enlazar con el marcador consejos definido en la página web de destino.

También, podríamos haber creado un enlace a este marcador desde el propio documento.html.

```
<a href="#consejos">Veamos más abajo unos consejos útiles</a>
```

En el momento que vemos la almohadilla, sabremos siempre que hace una referencia a un marcador de un documento.

El atributo `id` es común a todos los elementos y se emplea para identificar al elemento de forma unívoca dentro de un documento web, es decir, no puede haber dos elementos en el mismo documento web que tengan el mismo valor de atributo `id`. El valor por tanto que se pone a este atributo es una cadena de texto diferente para cada elemento.

Un uso habitual de los marcadores es para definir tablas de contenido. Por ejemplo

```
<h1>Tabla de contenidos</h1>
<P><a href="#seccion1">Introducción</a><br/>
<a href="#seccion2">Cap&iacute;tulo 1</a><br/>
<a href="#seccion2.1">Una nota personal</a><br/>
...resto de la tabla de contenidos...
...el cuerpo del documento...
<h2 id="seccion1">Introduction</h2>
...secci&oacute;n 1...
<h2 id="seccion2">Cap&iacute;tulo 1</h2>
...secci&oacute;n 2...
<h3 id="seccion2.1">Una nota personal</h3>
...secci&oacute;n 2.1...
```

11.4 Enlaces a otros recursos

También se podría realizar un hiperenlace a un fichero cualquiera, sin tener por qué ser un documento .html (podría ser un documento de Word .doc, de Adobe Acrobat. pdf, de dibujo de AutoCad .dxf, ... etc). En este caso el navegador intentará visualizar el fichero, y si no puede hacerlo, nos preguntará que queremos que haga con él. Una de las opciones suele ser grabarlo (el archivo), en nuestro ordenador de modo local. Es una forma de permitir a nuestros visitantes copiar ficheros a su ordenador. Por ejemplo, el siguiente enlace sive para descargar un archivo comprimido.

```
<A href="manual.zip">manual comprimido</A>
```

11.5 Formato del enlace

Por defecto, los enlaces aparecen con el siguiente formato:

- ✔ Un enlace no visitado está subrayado y con el texto en azul.
- ✔ Un enlace visitadao aparece subrayado y con el texto en morado.
- ✔ Un enlace activo aparece subrayado y con el texto en rojo.

Como podemos apreciar, vemos que los enlaces aparecen en diferente formato en función de su estado: no visitado, visitado y activo. En CSS disponemos de las pseudo-clases para especificar formato a elementos que pueden estar en diferente estado. Las pseudo-clases clasifican a los elementos según el estado del mismo. Es decir, los elementos son agrupados en función de los eventos que les hayan ocurrido: que el enlace haya sido ya visitado, que el cursor del ratón se encuentre encima o que deje de estar sobre él, etc. Así pues, con las pseudo-clases se selecciona no tanto al propio elemento como sus diferentes estados.

De los diferentes elementos con pseudo-clases existentes hasta la fecha, los más utilizados son los referentes a los enlaces. Concretamente, para los enlaces disponemos de las siguientes pseudo-clases.

Pseudo-clase	Significado
`:link`	Vínculos que todavía no han sido visitados
`:visited`	Vínculos ya visitados
`:hover`	Vínculos sobre los que está el cursor del ratón
`:active`	Vínculos sobre los que se está haciendo clic
`:focus`	Vínculos que están siendo seleccionados mediante teclado

La pseudo clase `:link` se usa para enlaces que dirigen a páginas que el usuario no ha visitado. En el ejemplo de código que sigue, los enlaces no visitados tendrán un color azul claro.

```
a:link {
    color: #6699CC;
}
```

La pseudo clase `:visited` se usa para enlaces que dirigen a páginas que el usuario ya ha visitado. Por ejemplo, el código siguiente hará que todos los enlaces visitados sean de color púrpura oscuro:

```
a:visited {
    color: #660099;
}
```

La pseudo clase `:active` se usa para enlaces que están activos. El código de este ejemplo hace que el color de fondo para los enlaces activos sea amarillo:

```
a:active {
    background-color: #FFFF00;
}
```

La pseudo clase `:hover` se usa cuando el puntero del ratón pasa por encima de un enlace. Esta pseudo-clase se puede usar para crear efectos interesantes. Por ejemplo, si queremos que nuestros enlaces sean de color naranja y estén en cursiva cuando el cursor pase sobre ellos, el código CSS que utilizaremos será el siguiente:

```
a:hover {
    color: orange;
    font-style: italic;
}
```

Es bastante popular crear diferentes efectos cuando el cursor está encima de un enlace. Por lo tanto, examinaremos unos cuantos ejemplos más relacionados con la pseudo-clase `:hover`.

Como recordarás, el espaciado entre los caracteres se puede ajustar usando la propiedad `letter-spacing`. Esta propiedad se puede aplicar a los enlaces para crear un efecto especial:

```
a:hover {
    letter-spacing: 10px;
    font-weight:bold;
    color:red;
}
```

La propiedad `text-transform` sirve para intercambiar entre letras en mayúscula y minúscula. Esto se puede usar también para crear un determinado efecto en los enlaces:

```
a:hover {
    text-transform: uppercase;
    font-weight: bold;
    color: blue;
    background-color: yellow;
}
```

Estos dos ejemplos dan una idea sobre las posibilidades casi infinitas al combinar diferentes propiedades.

Una pregunta muy recurrente es: ¿cómo quito el subrayado de los enlaces? Deberías considerar cuidadosamente la necesidad de quitar el subrayado, pues esto podría disminuir la usabilidad de tu sitio web de forma significativa. La gente está acostumbrada al subrayado azul de los enlaces en las páginas web y saben que pueden hacer clic en ellos. Si cambias el subrayado y el color de los enalces, existe una buena posibilidad de que los usuarios se confundan y, por lo tanto, no saquen todo el provecho al contenido de tu sitio web.

Dicho esto, es muy sencillo quitar el subrayado de los enlaces. Como recordarás, la propiedad `text-decoration` se puede usar para determinar si el texto aparece subrayado o no. Para quitar el subrayado, establece el valor de `text-decoration` a `none`.

```
a {
    text-decoration:none;
}
```

De forma alternativa, puedes establecer la propiedad `text-decoration` junto con otras propiedades para las cuatro pseudo clases.

```
a:link {
    color: blue;
    text-decoration:none;
}

a:visited {
    color: purple;
    text-decoration:none;
}

a:active {
    background-color: yellow;
    text-decoration:none;
}
```

```
a:hover {
    color:red;
    text-decoration:none;
}
```

12 Tablas

Una table es una estructura bidimensional formada por un conjunto de filas y columnas. A la intersección de una fila con una columna se le denomina celda. La información está contenida en las celdas.

El modelo de tabla de HTML permite al desarrollador web distribuir datos en filas y columnas de celdas. Cada tabla tiene asociada una descripción corta. Las filas de la tabla se pueden agrupar en una cabecera, pie y cuerpo. Los grupos de filas incluyen información que pueden usar los agentes de usuario para enfatizar su estructura y soportar desplazamiento del cuerpo fijando el pie y la cabecera. Cuando una tabla grande se imprime, cada página puede repetir la información del pie y la cabecera.

También se pueden agrupar las columnas para dar información adicional a la estructura de la tabla. Se pueden definir propiedades de las columnas al principio de la tabla para que el navegador pueda presentar la tabla incrementalmente en lugar de esperar toda la tabla entera. Las celdas pueden contener información de la cabecera o datos y pueden expandirse en mútiples filas y columnas.

12.1 Estructura de una tabla en HTML

El elemento para introducir una tabla en una página web es `table`. No es preciso determinar inicialmente el número de filas o columnas, el navegador se encarga de averiguarlo a medida que profundice en el contenido de la tabla que está cargando en pantalla. Cada fila se limita con el elemento `tr` y cada celda se limita con el elemento `td` o `th`.

Por tanto, una tabla en HTML, en su forma más básica, tiene la siguiente forma

```
<table>
    <tr>
        <th>Celda 1,1</th><th>Celda 1,2</th>...<th>Celda
1,n</th>
    </tr>
    <tr>
        <td>Celda 2,1</td><td>Celda 2,2</td> ... <td>Celda
2,n</td>     </tr>
    <tr>
        <td>Celda 3,1</td><td>Celda 3,2</td> ... <td>Celda
3,n</td>
    </tr>
…
    <tr>
```

```
        <td>Celda m,1</td><td>Celda m,2</td> ... <td>Celda
m,n</td>
    </tr>
</table>
```

Como se puede observar el elemento `table` contiene varios elementos `tr`, uno por fila. A su vez, cada elemento `tr` tiene elementos `th` o `td`, uno por celda. La información se introduce dentro de los elementos de celda, nunca fuera de ella.

Debemos tener cuidado de introducir el mismo número de celdas en cada fila, para que la tabla quede cuadrada. De lo contrario la tabla quedaría coja, es decir, habría celdas de más o de menos en algunas filas.

Si queremos ver los bordes de la tabla tenemos que emplear el atributo `border` con un valor númerico que hace referencia al grosor del borde. Sin embargo, más adelante en este mismo capítulo veremos que es mejor utilizar CSS para poner los bordes a la tabla.

12.2 Celdas de cabecera y celdas de datos

El elemento `th` se emplea para incluir una celda en una fila de cabecera. Estas celdas se caracterizan porque su contenido aparece por defecto en negrita y centrado. Podemos poner un elemento th en cualquier fila de la tabla, pero semánticamente solamente debe ponerse en celdas de la cabecera o el pie de una tabla.

Por otro lado, el elemento `td` se emplea para incluir una celda de datos. Estas celdas no tienen un formato especial por defecto.

En una celda de tabla, ya sea de cabecera o normal, puede aparecer cualquier tipo de elemento, como texto, enlaces, imágenes, encabezados, etc.

El siguiente ejemplo representa una tabla con el horario de un centro educativo

```
<table>
    <tr>
        <th></th>
        <th>Lunes</th>
        <th>Martes</th>
        <th>Miércoles</th>
        <th>Jueves</th>
        <th>Viernes</th>
    </tr>
    <tr>
        <td>8:30</td>
        <td>Lengua</td>
        <td>Lengua</td>
        <td>Inglés</td>
        <td>Francés</td>
        <td>Educación Física</td>
    </tr>
    <tr>
        <td>9:30</td>
```

```
        <td>Matemátias</td>
        <td>Lengua</td>
        <td>Educación Física</td>
        <td>Inglés</td>
        <td>Francés</td>
    </tr>
    <tr>
        <td>10:30</td>
        <td>Francés</td>
        <td>Educación Física</td>
        <td>Inglés</td>
        <td>Lengua</td>
        <td>Matemáticas</td>
    </tr>
</table>
```

Si vemos el resultado actual su aspecto es bastante frio.

	Lunes	Martes	Miércoles	Jueves	Viernes
8:30	Lengua	Lengua	Inglés	Francés	Educación Física
9:30	Matemátias	Lengua	Educación Física	Inglés	Francés
10:30	Francés	Educación Física	Inglés	Lengua	Matemáticas

Figura 32.- Ejemplo de tabla

Conforme vayamos añadiendo nuevos elementos y le demos formato su aspecto mejorará considerablemente.

12.3 Título de tabla

Si queremos que la tabla posea un título poseemos un elemento que es `caption` que va incluida dentro de la definición de la tabla y antes del comienzo de la definición de filas. Siguiente con el ejemplo anterior pondríamos lo siguiente.

```
<table>
    <caption>Horario de Clase</caption>
    <tr>
        <th></th>
        <th>Lunes</th>
        <th>Martes</th>
        <th>Miércoles</th>
        <th>Jueves</th>
        <th>Viernes</th>
    </tr>
    ...
```

Y el resultado sería este

Horario de Clase

	Lunes	Martes	Miércoles	Jueves	Viernes
8:30	Lengua	Lengua	Inglés	Francés	Educación Física
9:30	Matemátias	Lengua	Educación Física	Inglés	Francés
10:30	Francés	Educación Física	Inglés	Lengua	Matemáticas

Figura 33.- Título de la tabla

Vemos que el título aparece centrado respecto al espacio ocupado por la tabla.

12.4 Expansión de celdas

Muchas veces, estaremos interesados en que una celda de una fila ocupe el ancho de 2 celdas normales de la tabla, de forma que no haya separación entre las celdas. Por ejemplo

En este ejemplo la 2 fila tiene en su primera celda una extensión de 2 celdas normales del resto de la tabla. O también puede ser que queramos que una fila ocupe más filas de altura que el normal de la tabla. Veamos otro ejemplo:

En este ejemplo, las tres celdas de la segunda columna están formando una sola.

Para expandir una celda en filas o columnas existen dos atributos que nos sirven para indicar al navegador que la tabla poseerá un ancho de fila o columna doble, triple, cuádruple... del estándar definido para el resto de la tabla. Estos atributos pertenecen a los elementos `th` o `td` y son:

- ✔ `colspan="<valor_numérico>"` indica cuántas columnas ocupará la celda en la que se defina el atributo.
- ✔ `rowspan="<valor_numérico>"` ilndica cuántas filas ocupará la celda en la que se defina el atributo.

El valor numérico de estos elementos indica cuantas columnas o filas ocupara la celda. Veamos el ejemplo anterior modificado. A continuación mostramos el elemento `tr` correspondiente a la segunda fila.

```
…
<tr>
      <td>8:30</td>
      <td colspan="2">Lengua</td>
```

```
        <td>Inglés</td>
        <td>Francés</td>
        <td>Educación Física</td>
    </tr>
    ...
```

También, para poder apreciar bien la expansión de la celda hemos añadido un atributo `border` al elemento `table`. El resultado sería este

Horario de Clase

	Lunes	Martes	Miércoles	Jueves	Viernes
8:30	Lengua		Inglés	Francés	Educación Física
9:30	Matemátias	Lengua	Educación Física	Inglés	Francés
10:30	Francés	Educación Física	Inglés	Lengua	Matemáticas

Figura 34.- Expansión de celdas

Vemos que la celda correspondiente al lunes y martes de las 8:30 aparece ocupando dos columnas. En el siguiente ejemplo hemos añadido una fila para el recreo, el cual es todos los días a las 11:30. Habría que añadir el siguiente elemento tr al final de la tabla

```
    …
    <tr>
        <td>11:30</td>
        <td colspan=”5”>RECREO</td>
    </tr>
    ...
```

Quedaría de la siguiente forma

Horario de Clase

	Lunes	Martes	Miércoles	Jueves	Viernes
8:30	Lengua		Inglés	Francés	Educación Física
9:30	Matemátias	Lengua	Educación Física	Inglés	Francés
10:30	Francés	Educación Física	Inglés	Lengua	Matemáticas
11:30	RECREO				

Figura 35.- Expansión de celdas

Una expansión vertical implica que una celda ocupa varias filas. El siguiente ejemplo vemos

```
<table border="1">
    <caption>Horario de Clase</caption>
    <tr>
        <th></th>
        <th>Lunes</th>
        <th>Martes</th>
        <th>Miércoles</th>
        <th>Jueves</th>
```

```
            <th>Viernes</th>
        </tr>
        <tr>
            <td>8:30</td>
            <td colspan="2">Lengua</td>
            <td>Inglés</td>
            <td>Francés</td>
            <td rowspan="3">Educación Física</td>
        </tr>
        <tr>
            <td>9:30</td>
            <td>Matemátias</td>
            <td>Lengua</td>
            <td>Educación Física</td>
            <td>Inglés</td>
        </tr>
        <tr>
            <td>10:30</td>
            <td>Francés</td>
            <td>Educación Física</td>
            <td>Inglés</td>
            <td>Lengua</td>
        </tr>
        <tr>
            <td>11:30</td>
            <td colspan="5">RECREO</td>
        </tr>
    </table>
```

En este ejemplo tenemos la última celda de la segunda vila con el atributo `rowspan="3"` que indica que la celda ocupa tres filas. El resultado sería el siguiente

Horario de Clase

	Lunes	Martes	Miércoles	Jueves	Viernes
8:30	Lengua		Inglés	Francés	Educación Física
9:30	Matemátias	Lengua	Educación Física	Inglés	
10:30	Francés	Educación Física	Inglés	Lengua	
11:30	RECREO				

Figura 36.- Expansión de columnas

A la hora de expandir una celda deberemos tener cuidado de que la tabla no quede desequilibrada. Siempre que añadimos una expansión hay que quitar celdas en otro lugar de la tabla de la siguiente forma:

- ✔ Si se expande una celda horizontalmente (`colspan`) entonces deberé quitar celdas de la misma fila.
- ✔ Si se expande una celda verticalmente (`rowspan`) entonces tendré que quitar celdas de las filas siguientes.

12.5 Grupos de filas

Las filas de la tabla pueden agruparse en una cabecera, cuerpo y pie. Una tabla solo puede tener un grupo de cabecera, un grupo de pie, pero puede tener varios grupos de filas en el cuerpo. Para ello se emplean los elementos `thead` para la cabecera, `tfoot` para el pie y `tbody` para el cuerpo. Cuando están presentes, cada elemento define un grupo de filas con al menos una fila definida con un elemento `tr`.

Esta división posibilita al navegador realizar desplazamiento de los cuerpos de la tabla independientemente de la cabecera y el pie. Cuando una tabla grande se imprime, la información de la cabecera y el pie puede aparecer en cada página que contiene los datos de la tabla.

La cabecera y el pie de table deberían contener información sobre las columnas de la tabla y el cuerpo debe de contener las filas de datos.

Para crear un grupo de filas empleamos el elemento `thead`. Todos los elementos de fila `tr` incluidos entre la apertura y cierre de un `thead` serán filas de cabecera. El siguiente ejemplo ilustra el orden y la estructura de la cabecera, pie y cuerpo de tabla.

```
<table>
<thead>
     <tr> ...Celdas de cabecera...</tr>
</thead>
<tfoot>
     <tr> ...celdas del pie...</tr>
</tfoot>
<tbody>
     <tr> ...primera fila del bloque 1 de datos...</tr>
     <tr> ...segunda fila del bloque 2 de datos...</tr>
</tbody>
<tbody>
     <tr> ...primera fila del bloque 2 de datos...</tr>
     <tr> ...segunda fila del bloque 2 de datos...</tr>
     <tr> ...tercera fila del bloque 2 de datos...</tr>
</tbody>
</table>
```

`tfoot` debe aparecer antes de `tbody` dentro de un elemento `table` que el navegador utiliza para recibir el pie antes de cualquier número de filas de datos. Lo siguiente resume que elementos se necesitan y cuales pueden omitirse.

- ✔ El elemento `tbody` siempre es requerido, excepto cuando la tabla solamente contiene un cuerpo y no tiene secciones de cabecera o pie.
- ✔ El elemento `thead` y `tfoot` son requeridos cuando la tabla tiene cabecera y pie respecticamente.

Las secciones de cabecera, pie y cuerpo deben contener el mismo número de columnas.

12.6 Grupos de columnas

Los grupos de columnas permiten al desarrollador web crear divisiones estructurales dentro de una tabla. Se puede resaltar esta estructura mediante hojas de estilo en cascada o con el atributo `rules` del elemento `table`, aunque este último no está soportado en HTML5.

Una tabla puede contener un grupo de columna implícito, el cual abarca todas las columnas de la tabla, o cualquier número de grupos de columnas explícitos, cada uno de los cuales se define con el elemento `colgroup`. El elemento `col` permite compartir atributos entre varias columnas sin realizar un agrupamiento de columnas.

El elemento `colgroup` crea un grupo de columnas explícito. El número de columnas en el grupo puede especificarse de dos formas mutuamente exclusivas:

1. Definir el valor del atributo `span`, por defecto a 1.
2. Definir un elemento `col` dentro de un `colgroup`, el cual representa uno o más columnas en el grupo.

La ventaja de usar el atributo `span` es que se puede establecer el ancho de las columnas. Por ejemplo, si la tabla contiene 40 columnas, todas ellas con un ancho de 20, podría definirse así:

```
<table>
<colgroup span="40" style=”width:100px;"></colgroup>
…
</table>
```

Cuando es necesario establecer el formato a una o varias columnas dentro del grupo, es mejor utilizar el elemento `col`. Por ejemplo, supongamos que definimos un grupo de 40 columnas con ancho 100 píxeles, pero la columna 15 tiene un ancho de 300 píxeles. Esta tabla se definiría así

```
<table>
<colgroup style=”width:100px;”>
    <col span=”14”>
    <col span=”1” style=”width:200px;”>
    <col span=”25”>
</colgroup>
…
</table>
```

La propiedad CSS `width` dentro del elemento `colgroup` es heredado por las 40 columnas. El primer elemento `col` se refiere a las primeras 14 columnas, no haciendo nada especial con ellas, y el segundo asigna un ancho de 200 a la columna 15. El último `col` se refiere a las siguientes 25 columnas y tampoco hace nada con ellas.

El atributo `span` es ignorado por el navegador si `colgroup` contiene elementos `col`. Este elemento también define el atributo `span` y con el mismo significado.

El elemento `col` permite agrupar columnas para cambiar formato, pero no las agrupa estructuralmente, eso lo hace `colgroup`. Elementos `col` pueden aparecer dentro o fuera de `colgroup`.

12.7 Calcular el número de columnas de una tabla

Hay dos formas de determinar el número de columnas en una tabla. Son las siguiente en orden de precedencia:

1. Si la tabla contiene cualquier elemento `colgroup` o `col`, el navegador calculará el número de columnas sumando por cada elemento `col` el valor del atributo `span`.
2. Por cada elemento `colgroup` conteniendo al menos un elemento `col`, ignora el atributo `span` de `colgroup`. Por cada elemento `col` realiza el cálculo anterior.
3. Por cada elemento `colgroup` vacío toma el valor de su atributo `span`.
4. Si la tabla no contiene grupos de columnas, el navegador basa el número de columnas en las requeridas por las filas. El número de columnas será el número de columnas de la fila que tenga más columnas, incluyendo celdas que se expanden horizontalmente.

Por ejemplo, para cada una de las tablas siguientes, los dos metodos de cálculo de columnas deberían resultar 3 columnas.

```
<table>
    <colgroup span="3"></colgroup>
    …

</table>

<table>
    <colgroup>
         <col>
         <col span="2">
    </colgroup>
    …

</table>

<table>
    <colgroup>
         <col>
    </colgroup>
    <colgroup span="2"></colgroup>
    …

</table>

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

```
  <td></td>
  <td></td>
</tr>
```

13 Formato de tabla con CSS

Como el contenido de las tablas puede ser cualquier elemento visto hasta ahora y algunos más que aún tenemos que ver, el formato CSS que podemos especificar a estos elementos es el mismo que hemos visto en capítulos anteriores. Podemos asignar a la tabla, fila o celdas fuentes, estilo de fuentes, efectos de fuente, alineación, color en primer plano, color o imagen de fondo, etc. Veamos un ejemplo

```
th {
    background-color: #FFC;
    font: italic bold 12pt arial, sans-serif;
    color: darkgray;
    text-align: left;
    font-variant: small-caps;
}
```

Con la regla anterior estamos dando a las celdas de cabecera una fuente, color de fondo, color en primer plano, alineación de texto y fuente en versales. El resultado sería el siguiente.

	Lunes	Martes	Miércoles	Jueves	Viernes

Figura 37.- Formato de celdas de cabecera

Además de lo anterior, en las tablas es muy común establecer el alto y ancho de las celdas y/o tabla, junto con los bordes. El alto y ancho de elementos lo hemos visto anteriormente, aunque no en profundidad, ya que se verá en detalle en el próximo capítulo. Los bordes aún no lo hemos visto, pero se van a introducir aquí.

Respecto a la alineación utilizaremos la propiedad `text-align` vista anteriormente para alinear el contenido de las celdas e introduciremos la propiedad `vertical-align` para la alineación vertical.

13.1 Alto y ancho de tabla y celdas

¿Qué ancho ocupa una tabla? Por defecto el navegador asigna a cada celda el ancho necesario para albergar su contenido. Por tanto, el ancho de una columna será el mismo que el de la celda más ancha. Finalmente, sumando el ancho de todas las columnas, se obtiene el ancho de la tabla. Esto equivale a asignar la propiedad `width` el valor `auto`.

Para establecer el ancho usamos la propiedad `width`. Esta propiedad actúa de la siguiente manera:

- ✔ Si se establece la propiedad `width` para una tabla indica el ancho total de la tabla. Este ancho prevalece sobre la suma de los anchos de las columnas

- ✔ Si se establece la propiedad `width` para una tabla y también para una celda, visualizará la columna donde se encuentra la celda con ese ancho siempre que el ancho total de la tabla y el resto de las columnas lo permitan. Es decir, la suma de los anchos de las columnas no superan el ancho de la tabla.
- ✔ Si el ancho de las columnas superan el ancho total de la tabla, entonces la tabla se muestra con el ancho establecido para ella y ajustará el ancho de las columnas de forma equitativa.
- ✔ Si no se especifica la propiedad width a la tabla y si a las columnas, entonces la tabla ocupará todo el ancho de la pantalla si la suma del ancho de las celdas supera el ancho de la pantalla. De lo contrario la tabla tendrá de ancho la suma de los anchos de las columnas.
- ✔ La propiedad `width` no se aplica a filas.

Si indicamos ancho de tabla estaremos bastante limitados. Así que solamente lo aplicaremos en casos en los que sea estrictamente necesario. Es mejor asignar ancho a las columnas estableciendo la propiedad `width` en una celda de cada columna o definiendo grupos de columnas mediante el elemento `colgroup`.

Para el alto estableceremos la propiedad `height`, la cual actuará de la siguiente manera:

- ✔ Si establecemos la propiedad `height` para la tabla, la tabla tendrá esa altura siempre que la suma de las alturas establecidas para las filas y/o celdas sea inferior a la misma.
- ✔ Si se establece la propiedad `height` para una fila, la fila tendrá esa altura siempre que sea superior a la altura de cualquiera de sus celdas.
- ✔ Si se establece la propiedad `height` para una celda, la fila donde se encuentre dicha celda tendrá la altura establecida.

Como se puede observar la altura establecida sobre una celda tiene precedencia sobre la indicada en su fila, y esta tiene precedencia sobre la que se indica en la tabla.

Por ejemplo, supongamos que asignamos las siguientes alturas y anchuras a nuestra tabla de ejemplo.

```
table {
    width: 700px;
    height: 500px;
}

tr {
    height: 120px;
}

th {
    width: 100px;
```

```
    height: 150px;
}

td {
    width: 150px;
    height: 200px;
}
```

El resultado es el siguiente

Horario de Clase

	Lunes	Martes	Miércoles	Jueves	Viernes
8:30	Lengua		Inglés	Francés	Educación Física
9:30	Matemátias	Lengua	Educación Física	Inglés	

Figura 38.- Altura y anchura

Analicemos el ejemplo anterior. Vemos que a la tabla le hemos dado una anchura de 700 píxeles, y a cada celda de datos le hemos dado 150 píxeles. Pues bien, si tenemos 6 columnas y multiplicamos por 150 píxeles nos da un total de 900 píxeles. Sin embargo, como la propiedad `width` establecida en la tabla tiene precedencia, entonces el ancho total de la tabla es 700 píxeles y el ancho de las columnas se reparte equitativamente. Nótese que las celdas de cabecera tienen 100 píxeles, pero en el caso de que no se hubiera definido ancho

para la tabla y se aplicará el ancho de las columnas, siempre se aplica el mayor, por tanto, en este caso sería el de las celdas de datos.

Respecto a la altura, tenemos en la tabla una altura de 500 píxeles y la fila 120 píxeles. Como la altura de la fila tiene precedencia sobre la tabla y hay 5 filas, entonces la altura sería, por ahora, de 600 píxeles. Sin embargo, vemos que las celdas de cabecera tienen 150 píxeles de altura, que tiene precedencia sobre la altura de su fila, y las celdas de datos tienen 200 píxeles de altura, que también tienen precedencia sobre los de su correspondiente fila. Por tanto la altura total de la tabla es de 150 + 4x200 = 950 píxeles.

Con las reglas anteriores estamos asignando igual ancho y alto a todas las filas de la tabla, cosa no muy común. Por ahora, y hasta que en el siguiente capítulo veamos las clases y los identificadores, podríamos establecer diferentes anchos y altos de fila aplicando estilo de forma local mediante el atributo `style` en el correspondiente elemento.

13.2 Bordes

Hasta ahora hemos visto el atributo `border` del elemento `table` para poner bordes a las celdas. Este borde es muy pobre porque no permite afinar su grosor ni establecer ninguna otra característica de formato más, como el color o el estilo de borde. Para establecer bordes disponemos de las siguientes propiedades.

La anchura del borde se define por medio de la propiedad `border-width`, que dispone de los valores `thin`, `medium` y `thick`, o de un valor numérico indicado en píxeles. La siguiente imagen ilustra cómo funciona.

Figura 39.- Bordes

La propiedad `border-color` define el color del borde. Los valores de esta propiedad son los valores de rigor vistos anteriormente.

Se puede elegir entre diferentes estilos de borde. Más abajo se muestran 8 estilos de borde según los interpreta. Todos los ejemplos se muestran con el valor del color a oro y el valor de la anchura a `thick`, pero se pueden mostrar, por supuesto, en otros colores y grosores. Si no queremos mostrar ningún borde, se puede usar los valores `none` o `hidden`.

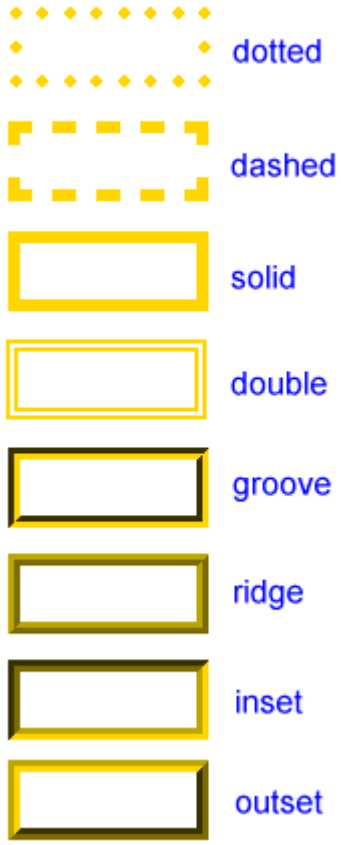


Figura 40.- Estilo de bordes

Veamos algunos ejemplos de bordes combinando las tres propiedades anteriores. Las tres propiedades descritas anteriormente se pueden unir para cada elemento y así producir diferentes bordes. Para ilustrar esto, asignaremos bordes a las celdas de la tabla. El resultado puede que no sea demasiado bonito pero ilustra gráficamente algunas de las muchas posibilidades.

```
table {
    border-width: 3px;
    border-style: dashed;
    border-color: brown;
}

th {
    border-width: thick;
    border-style: dotted;
    border-color: teal;
}

td {
    border-width: 5px;
    border-style: outset;
    border-color: red;
}
```

El resultado sería el siguiente

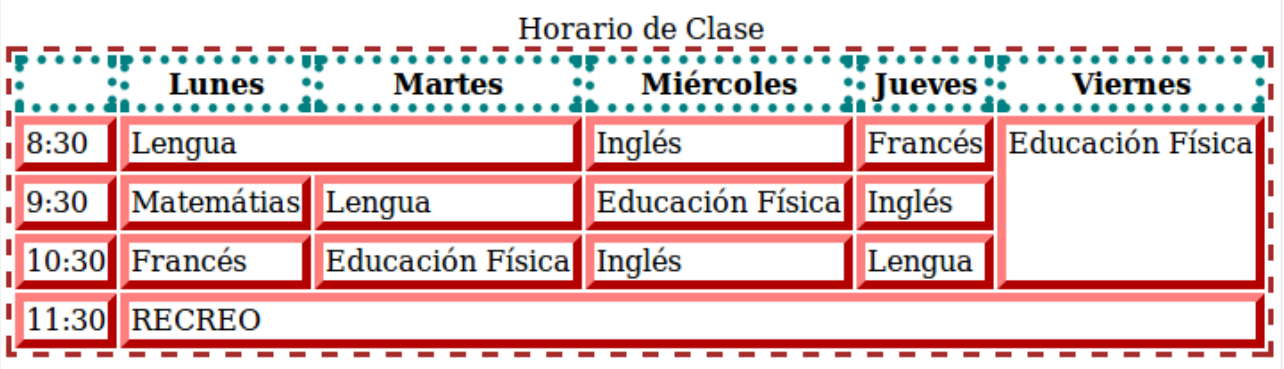


Figura 41.- Bordes en tabla

Los bordes no se pueden establecer para una fila, solamente para celdas o la tabla completa. En el ejemplo anterior hemos suprimido el atributo `border` de la tabla. Si está presente, se aplicaría en el caso de que no estuvieran definidos los bordes mediante CSS.

También es posible establecer diferentes bordes para el borde superior (`top`), inferior (`bottom`), derecho (`right`) e izquierdo (`left`). En el siguiente ejemplo vemos las propiedades para ello:

```
th {
    border-top-width: thin;
    border-top-style: dotted;
    border-top-color: teal;

    border-right-width: medium;
    border-right-style: solid;
    border-right-color: cyan;

    border-bottom-width: thick;
    border-bottom-style: outset;
    border-bottom-color: lightblue;

    border-left-width: 5px;
    border-left-style: dashed;
    border-left-color: marine;

}
```

Como ocurre con el fondo y la fuente, los bordes también tienen una propiedad abreviada llamada `border`. Veamos un ejemplo:

```
td {
    border-width: 5px;
    border-style: outset;
    border-color: red;
}
```

La declaración anterior se puede combinar así:

```
td {
```

```
    border: 5px outset red;
}
```

Hay que poner en orden el grosor, el estilo y el color del borde. También hay propiedades abreviadas para los elementos individuales de cada borde, como en el siguiente ejemplo.

```
th {
    border-top: thin dotted teal;

    border-right: medium solid cyan;

    border-bottom: thick outset lightblue;

    border-left: 5px dashed marine;
}
```

13.3 Alineación

La propiedad `text-align` se emplea para alinear el texto horizontalmente. Esta propiedad la vimos anteriormente y cuando se establece para una celda, fila o tabla se aplica en este orden de precedencia.

También disponemos de la propiedad `vertical-align`, la cual se emplea para establecer la alineación vertical del contenido de la celda. Esto es útil cuando el alto de fila es mayor que el necesario para albergar el contenido de sus celdas. Disponemos de los valores `top` (arriba), `middle` (en medio) y `bottom` (abajo). Por defecto, las celdas se alinean en medio.

Esta propiedad no se aplica a `table`, pero si a `tr`, `td` y `th`. Si se especifica sobre una celda tiene precedencia sobre la especificada sobre la fila. Veamos el siguiente ejemplo.

```
tr {
    text-align: right;
    vertical-align: bottom;
}
td {
    text-align:left;
    vertical-align: top;
}
```

El resultado es este

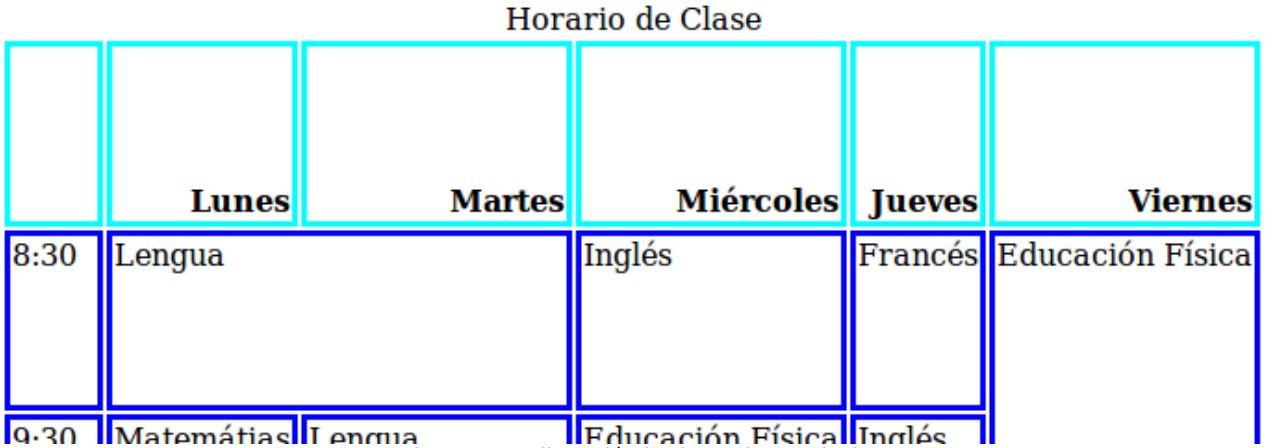


Figura 42.- Alineación horizontal y vertical

Como a las celdas de cabecera no hemos indicado ningún formato, se aplica el de la fila, mientras que las celdas de datos se les aplica su formato.

13.4 Relleno

El contenido de las celdas aparece por defecto muy pegado a los bordes. Podemos establecer un espacio de separación entre el borde de la celda y su contenido mediante la propiedad `padding`. Por ejemplo

```
td {
    border: medium solid cyan;
    padding: 5px;
}
```

El resultado es el siguiente

Horario de Clase

	Lunes	Martes	Miércoles	Jueves	Viernes
8:30	Lengua		Inglés	Francés	Educación Física
9:30	Matemátias	Lengua	Educación Física	Inglés	
10:30	Francés	Educación Física	Inglés	Lengua	
11:30	RECREO				

Figura 43.- Relleno

Vemos que las celdas de datos están más espaciadas por el relleno de 5 píxeles que le hemos dado. Podemos también especificar un relleno diferente arriba, abajo, izquierda y derecha mediante las propiedades `padding-top`, `padding-bottom`, `padding-left` y `padding-right`. Vamos el siguiente ejemplo

```
th {
    border: medium solid cyan;
    padding-top: 5px;
    padding-right: 10px;
```

```
    padding-bottom: 15px;
    padding-left: 20px;
}
```

El resultado es

Horario de Clase

	Lunes	Martes	Miércoles	Jueves	Viernes
8:30	Lengua		Inglés	Francés	Educación Física
9:30	Matemátias	Lengua	Educación Física	Inglés	
10:30	Francés	Educación Física	Inglés	Lengua	
11:30	RECREO				

Figura 44.- Relleno

Ahora la fila de cabecera tienen diferente separación de cada borde con el contenido de la celda.

13.5 Propiedades específicas para elemento table

Las siguientes propiedades solamente se aplican al elemento `table` y permiten definir el comportamiento, el diseño y la ubicación de los elementos que componen cada tabla.

13.5.1 La ubicación del título

La propiedad `caption-side` nos permite ubicar el título de la tabla por encima o por debajo de la misma. La alineación horizontal del mismo se puede establecer con la propiedad `text-align`.

```
table {caption-side: top | bottom | inherit }
```

Por ejemplo

```
table { caption-side: bottom; }
```

Provocará que el título aparezca debajo de la tabla, detrás de la última fila.

13.5.2 Formato de las tablas

La propiedad `table-layout` se usa para diseñar las filas, columnas o celdas de una tabla. Entre las posibilidades de diseño podemos definir si las mismas van a tener el tamaño fijo que estipulemos (`fixed`) o se adecuarán al contenido sin importar la medida que hayamos establecido (`auto`).

```
table { table-layout: auto | fixed | inherit }
```

13.5.3 Espacio entre celdas

La propiedad `border-spacing` especifica la separación entre celdas adyacentes. Si especificamos un solo valor, este actúa sobre toda la tabla. Si especificamos 2 valores el primero define la separación horizontal y el segundo la vertical.

```
table { border-spacing: distancia_h distancia_v | inherit }
```

Veamos el siguiente ejemplo

```
table {
    border-spacing:5px;
}
```

El resultado es

Horario de Clase

	Lunes	Martes	Miércoles	Jueves	Viernes
8:30	Lengua		Inglés	Francés	Educación Física
9:30	Matemátias	Lengua	Educación Física	Inglés	
10:30	Francés	Educación Física	Inglés	Lengua	
11:30	RECREO				

Figura 45.- Separación de celdas

13.5.4 Modelo de los bordes

La propiedad `border-collapse` nos permite seleccionar la apariencia de los bordes de cada celda de la tabla. Existen 2 modelos diferentes de bordes: separados y continuos.

```
table { border-collapse: collapse | separate | inherit }
```

El valor `collapse` hace que los bordes de las celdas se junten con los de las celdas adyacentes. Hay que tener cuidado porque si las celdas tienen diferentes bordes el resultado podría no ser el esperado. Los conflictos se resuelve según las reglas de resolución de conflictos de bordes de tabla de la especificación CSS2 del W3C.

Por defecto están separados, o lo que es lo mismo, se podría aplicar la propiedad `border-spacing`. Sin indicamos la propiedad `border-collapse`, entonces no se aplica `border-spacing`. Veamos el siguiente ejemplo:

```
table {
    border-collapse: collapse;
}

th {
    border: medium solid cyan;
}
```

```
td {
    border: medium solid cyan;
}
```

El resultado es

Horario de Clase

	Lunes	Martes	Miércoles	Jueves	Viernes
8:30	Lengua		Inglés	Francés	
9:30	Matemátias	Lengua	Educación Física	Inglés	Educación Física
10:30	Francés	Educación Física	Inglés	Lengua	
11:30	RECREO				

Figura 46.- Bordes continuos

13.5.5 Comportamiento de las celdas vacias

La propiedad `empty-cells` nos permite controlar la visualización de los bordes y fondos de una celda vacía.

```
table { empty-cells: show | hide | inherit }
```

Entendemos como celda vacía aquella que no tiene ningún contenido. Por ejemplo, la primera celda de la primera fila de la tabla de ejemplo no tiene ningún valor. El elemento `th` aparece así

```
<th></th>
```

Si establecemos la propiedad `empty-cells` con el valor `show`, entonces esta celda se verá con su borde y color de fondo. Este es el valor por defecto como podemos apreciar en los ejemplos anteriores.

Sin embargo, si el valor es `hide`, entonces no se mostrará ni el color de fondo ni el borde. Veamos el siguiente ejemplo

```
table {
    empty-cells:hide;
}

th {
    border: medium solid blue;
    background-color: lightblue;

}
```

El resultado es

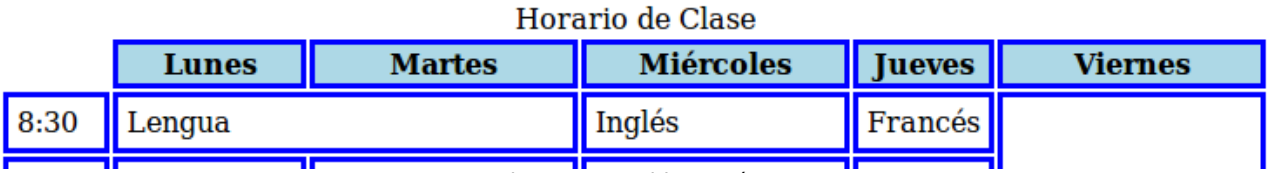


Figura 47.- Celdas vacías

14 Maquetación web

La maquetación web consiste en la disposición de los elementos visuales de una pagina web con el fin de crear la interfaz de usuario, es decir, aquellos elementos con los que el usuario interactua.

Hace unos años, la maquetación de las páginas web se realizaba utilizando tablas. Una vez entendido este proceso podía resultar sencillo, aunque si no se dominaban las tablas, podía convertirse en algo tedioso. El problema de las tablas es que generaban un página muy encorsetada, y el código se volvía complejo de entender. Además, algunos buscadores encontraban problemas al analizar la estructura de la página.

Actualmente, la maquetación con tablas ha caído en desuso, y se realiza utilizando capas, también llamadas divisiones o contenedores. La colocación de las capas en la página se realiza a través de CSS. Esto permite, por ejemplo, que podamos pasar de un diseño con un menú lateral a otro con el menú en la parte superior, sólo cambiando la hoja de estilos.

En este capítulo veremos las técnicas de maquetación más comunes, aunque hay que decir que existen diversas formas de conseguir una misma maquetación.

14.1 Clases e identificadores

En muchas ocasiones se necesita aplicar un estilo especial a un elemento concreto o a un grupo concreto de elementos. En este capítulo examinaremos cómo usar los atributos globales `class` e `id` para especificar propiedades para los elementos seleccionados.

Hasta el momento las reglas de formato que hemos visto tienen un selector muy simple. Consiste en el nombre del elemento para asignar el formato especificado en las propiedades de la regla a todos los elementos. Pero vamos a encontrarnos en situaciones en la que necesitemos dar un formato específico a un título concreto de forma diferente a los otros títulos del sitio web, o agrupar varios elementos en una categoría y dar a cada categoría un estilo especial. Para poder hacer esto necesitamos definir clases o identificadores de elementos.

14.1.1 Clases

Un atributo global de los elementos HTML es `class`. Global significa que todos los elementos HTML tienen este atributo. Su finalidad es la de agrupar los elementos por clases o grupos para que así, CSS u otro lenguaje pueden seleccionarlos y distinguirlos de los demás. Su uso es opcional aunque recomendable. Su sintaxis en HTML sería, por ejemplo:

```
<img src="casa.jpg" class="fotos">
```

Un elemento puede pertenecer a varias clases a la vez ya que éstas no son excluyentes. Para ello se especifica dentro del atributo `class` los nombres de las clases separados por espacios:

```
<img src="chalet.jpg" class="fotos casas">
```

Si un elemento HTML ha sido identificado siguiendo este método, es posible darle formato mediante una regla de estilo:

```
.fotos {border-width: 1px;}
.casas {border-width: 3px;}
```

En el primer ejemplo, se crearía un borde de 1 píxel a todos los elementos (imágenes, párrafos, etc.) que pertenezcan a la clase `fotos`. En el segundo, los elementos de la clase `casas` tendrían un marco de 3 píxeles; es decir, `chalet.jpg` tendría dicho marco mientras que `casa.jpg` no (ya que no pertenece a la clase `casas`).

En la práctica, la asignación de elementos a más de una clase proporciona muy buenos resultados, ya que permite La especificación de características concretas a grupos heterogéneos de elementos. Sin embargo, puede resultar un poco complicado para principiantes o personas poco experimentadas.

Veamos otro ejemplo. Digamos que tenemos dos listas de enlaces con los diferentes tipos de uva usados para el vino tinto y el blanco. El código HTML sería el siguiente:

```
<p>
Uvas para el vino blanco:<br/>
<a href="ri.html">RiesLIng</a><br/>
<a href="ch.html">Chardonnay</a><br/>
<a href="pb.html">Pinot Blanc</a><br/>
</p>

<p>
Uvas para el vino tinto:<br/>
<a href="cs.html">Cabernet Sauvignon</a><br/>
<a href="me.html">Merlot</a><br/>
<a href="pn.html">Pinot Noir</a><br/>
</p>
```

Así pues, queremos que los enlaces relativos al vino blanco sean amarillos, los enlaces relacionados con el vino tinto sean rojos, y el resto de enlaces de la página web sigan siendo azules.

Para lograr esto, dividiremos los enlaces en dos categorías. Esto se hace asignando una clase para cada tipo de enlace, usando el atributo `class`. Intentemos especificar algunas clases en el ejemplo anterior:

```
<p>
Uvas para el vino blanco:<br/>
<a href="ri.html" class=”vinoblanco”>RiesLIng</a><br/>
```

```
<a href="ch.html" class=”vinoblanco”>Chardonnay</a><br/>
<a href="pb.html" class=”vinoblanco”>Pinot Blanc</a><br/>
</p>

<p>
Uvas para el vino tinto:<br/>
<a href="cs.html" class=”vinotinto”>Cabernet Sauvignon</a><br/>
<a href="me.html" class=”vinotinto”>Merlot</a><br/>
<a href="pn.html" class=”vinotinto”>Pinot Noir</a><br/>
</p>
```

A partir de aquí, podemos definir propiedades especiales para los enlaces que hacen referencia al vino tinto y al vino blanco, respectivamente.

```
a {
    color: blue;
}

a.vinoblanco{
    color: #FFBB00;
}

a.vinorojo {
    color: #800000;
}
```

Como se muestra en el ejemplo, se pueden definir las propiedades para los elementos que pertenecen a una clase usando `.nombredelaclase` en la hoja de estilo del documento.

En el ejemplo anterior definimos el formato para los enlaces de las clases vinoblanco y vinorojo, pero son elementos `a`. Si queremos definir una clase genérica que se aplica a cualquier elemento solamente es necesario especificar el punto y el nombre de la clase, no el elemento. Por ejemplo

```
.vinoblanco{
    color: #FFBB00;
}
```

En este ejemplo no hemos definido elemento, solo una clase. Ahora, cualquier elemento que tenga vinoblanco como valor de atributo class tendrá el texto del color especificado.

14.1.2 Identificadores

Los elementos HTML disponen de un atributo global llamado `id` (identificador) cuya finalidad es la de asignar una identificación unívoca al elemento. De esta forma, CSS u otro lenguaje podrá seleccionarlo y distinguirlo de los demás. A través del atributo `id` se asigna al elemento un nombre o una expresión que lo distingue de los demás. Sería el caso del siguiente párrafo:

```
<p id="despedida">
```

El uso de los identificadores es opcional por lo que no tiene sentido definirlos si después no se prevé que CSS u otro lenguaje vaya a utilizarlos. Sin embargo, es recomendable su uso cuando se emplea CSS ya que permite hacer una selección más ajustada de los elementos HTML que se desean formatear.

Los valores de este atributo son libres y deben comenzar por una letra seguida de un número cualquiera de caracteres alfanuméricos. Además, se recomienda que sean valores que describan o clasifiquen de forma breve o esquemática al elemento y que sean fácilmente reconocibles para el programador.

Por ejemplo, si se quiere indicar que la fuente de los elementos `p` identificados como despedida, debe tener un tamaño de 14 píxeles, se tendría que escribir la siguiente regla de estilo:

```
p#despedida { font-size: 14px;}
```

Con el símbolo almohadilla (#) y el nombre del identificador se seleccionarán sólo los párrafos del documento HTML identificados como `despedida`. La fuente de estos párrafos tendrá un tamaño de 14 píxeles; la del resto tendrá el tamaño especificado en el documento HTML.

Lo que hace especial al atributo `id` es que no pueden existir dos elementos dentro del mismo documento con el mismo `id`. Cada id tiene que ser único. En cualquier otro caso, se debería usar el atributo `class` en su lugar. Ahora, examinemos un ejemplo de un posible uso del atributo id.

```
<h1>Capítulo 1</h1>
...
<h2>Capítulo 1.1</h2>
...
<h2>Capítulo 1.2</h2>
...
<h1>Capítulo 2</h1>
...
<h2>Capítulo 2.1</h2>
...
<h3>Capítulo 2.1.2</h3>
...
```

El código anterior podría hacer referencia a encabezados de cualquier documento dividido en capítulos y párrafos. Sería normal asignar un id a cada capítulo de la siguiente manera:

```
<h1 id="c1">Capítulo 1</h1>
...
<h2 id="c1-1">Capítulo 1.1</h2>
...
```

```
<h2 id="c1-2">Capítulo 1.2</h2>
...
<h1 id="c2">Capítulo 2</h1>
...
<h2 id="c2-1">Capítulo 2.1</h2>
...
<h3 id="c2-1-2">Capítulo 2.1.2</h3>
...
```

Digamos que el título del capítulo 1.2 tiene que estar en rojo. Usando el código CSS necesario, se podría hacer así:

```
#c1-2 {
    color: red;
}
```

Como se muestra en el ejemplo anterior, se pueden definir las propiedades de un elemento específico usando `#nombredelidentificador` en la hoja de estilo del documento.

14.2 Agrupación de elementos

Los elementos `span` y `div` se usan para agrupar y estructurar un documento, y se usarán, a menudo, junto con los atributos `class` e `id`. Ya hicimos una introducción del elemento `span` en el capítulo dedicado a la introducción de texto en HTML. Ahora lo veremos en profundidad.

En esta sección revisaremos el uso de los elementos `span` y `div`, ya que estos dos elementos de HTML son, precisamente, de importancia clave en lo que se refiere a CSS.

14.2.1 Elemento SPAN

El elemento `span` es lo que se podría denominar un elemento neutro que no añade semnática al documento en sí. Pero con CSS `span` se puede usar para añadir características visuales distintivas a partes específicas de texto en los documentos las cuales no forman el contenido completo de un elemento.. Un ejemplo de esto podría ser esta cita de Benjamin Franklin:

```
<p>El que pronto se acuesta y pronto se levanta,es hombre
saludable, rico y sabio.</p>
```

Digamos que queremos que lo que el señor Franklin considera como las ventajas de no pasarse todo el día durmiendo, aparezca enfatizado en rojo. Para este fin, podemos marcar dichas ventajas con el elemento `span`. A cada elemento `span` se le añade el atributo `class`, que podemos definir así en nuestra hoja de estilo:

```
<p>
El que pronto se acuesta y pronto se levanta,es hombre <span
class="ventaja">saludable</span>,<span
class="ventaja">rico</span>y <span class="ventaja">sabio</span>.
```

```
</p>
```

El código CSS necesario para producir este efecto es el siguiente:

```
.ventaja {
    color:red;
    font-weight: bold;
}
```

Por supuesto, se puede usar también el atributo `id` para añadir estilo a los elementos definidos con `span`. Pero recuerda que tendrás que aplicar siempre un atributo `id` único para cada uno de los tres elementos `span`, tal como vimos anteriormente.

14.2.2 Elemento DIV

Mientras que `span` se usa dentro de un elemento a nivel de bloque como vimos en el ejemplo anterior, `div` se usa para agrupar uno o más elementos a nivel de bloque.

Aparte de esta diferencia, la agrupación con `div` funciona más o menos igual. Veamos un ejemplo con dos listas de presidentes de los EE.UU., divididas según su filiación política.

```
<div id="democratas">
    <h1>Presidentes demócratas</h1>
    <UL>
    <LI>FrankLIn D. Roosevelt</LI>
    <LI>Harry S. Truman</LI>
    <LI>John F. Kennedy</LI>
    <LI>Lyndon B. Johnson</LI>
    <LI>Jimmy Carter</LI>
    <LI>Bill CLInton</LI>
    </UL>
</div>

<div id="republicanos">
    <UL>
    <LI>Dwight D. Eisenhower</LI>
    <LI>Richard Nixon</LI>
    <LI>Gerald Ford</LI>
    <LI>Ronald Reagan</LI>
    <LI>George Bush</LI>
    <LI>George W. Bush</LI>
    </UL>
</div>
```

En nuestra hoja de estilo podemos utilizar la agrupación del mismo modo que antes:

```
#democratas {
    background-color:blue;
}

#republicanos {
    background-color:red;
```

```
}
```

14.3 Capas

En los ejemplos anteriores, sólo hemos usado `div` y `span` con cosas muy sencillas como, por ejemplo, texto y colores de fondo. Por ahora nos sirve a modo de introducción pero ambos elementos tienen el potencial para realizar cosas más avanzadas, sobre todo `div` el cual se usa para definir capas.

Como ya hemos visto en el apartado anterior, `span` sirve para aplicarle estilo a una pequeña parte de una página HTML. Por ejemplo, con ella podríamos hacer que una parte de un párrafo se coloree en rojo. Con `span` no es habitual englobar un trozo muy grande de texto, por ejemplo el que comprenda a varios párrafos. Con `div` también podemos aplicar estilo a partes de la página HTML.

La diferencia entre `span` y `div` es que con esta última si que podemos aplicar estilo a una parte más amplia de la página, por ejemplo a tres párrafos. Además que el elemento `div` tiene un uso adicional que es el de crear divisiones o fragmentos en la página a las que podremos aplicar un formato específico. Su uso más destacado es el de convertir esa división en una capa.

Una capa es una división, una parte de la página, que tiene un comportamiento muy independiente dentro de la ventana del navegador, ya que la podemos colocar en cualquier parte de la misma y le aplicaremos un formato concreto a los elementos incluidos dentro de la capa.

Mediante los atributos `style`, `class` e `id` podremos modificar de una manera muy exhaustiva la presentación de los contenidos en las capas en la página. Sin embargo, a la hora de dar formato a una capa no solo dispondremos de las propiedades CSS que hemos visto hasta ahora, sino que también disponemos de otras propiedades nos sirven para posicionar la capa en la página.

Aparte de los elementos vistos hasta ahora, es posible dotar a una página web de una buena estructura mediante el uso de capas. Éstas son divisiones o secciones lógicas dentro del documento en cuestión y se obtienen mediante la inclusión en el mismo de elementos `div`. Este elemento permite dividir el documento en grandes secciones (a diferencia de `span`, destinado a especificar pequeñas porciones del mismo) y dotar a éstas de un comportamiento relativamente independiente con el uso de las propiedades adecuadas.

La capa, en cualquier caso, no es una simple división, sino más bien, un cuadro. Se compone de un área de contenido, el espacio alrededor de esa área (`padding` o relleno), el lado exterior del relleno (`border` o borde) y el espacio invisible alrededor del borde (`margin` o margen). Además, la capa tendrá un posicionamiento (lugar que ocupa en la pantalla) y unas dimensiones (alto y ancho).

El tratamiento de los elementos como capas puede resultar muy práctico para el desarrollador. Por una parte, identificar los distintos elementos y colocarlos en capas siguiendo un orden lógico servirá para tener el documento perfectamente estructurado en

secciones o divisiones. Por otra parte, a través de las propiedades adecuadas se puede determinar tanto la apariencia como la posición del cuadro de cada elemento y al hacerlo, tener bajo control la presentación de la página, estableciendo dónde y cómo queremos que se sitúen las capas, es decir, los grupos de textos, imágenes, etc.

Para situar las capas se debe:

- ✔ Dejar la capa en el flujo del documento. De esta forma será tratada como un elemento y será situada de acuerdo al orden en que aparece en el código. Esta opción es la que se usa por defecto si no se tienen definidas hojas de estilo que controlen el posicionamiento de los elementos.
- ✔ Especificar sus coordenadas exactas. Se debe hacer con respecto a su elemento padre o a la ventana del navegador. Además, si las capas se solapan, es necesario especificar el orden en que lo hacen. Ésta es la opción que se usará para controlar la ubicación de los elementos de la página.

Una vez determinado el lugar en que aparecen las capas, se podrá controlar su apariencia, o sea, su relleno, borde, márgenes, tamaño, alineación o color. Todo ello a través de una serie de propiedades que hacen referencia al modelo de caja de CSS y que se verán en el siguiente apartado.

Decir también, que las capas pueden anidarse, es decir, podemos crear capas dentro de otras capas.

14.4 El modelo de caja

Es uno de los elementos básicos de CSS y por lo tanto su correcta interpretación resulta fundamental a la hora de lograr los resultados deseados en un diseño, por más simple que éste resulte.

Para entrar en tema, vamos a construir un sencillo ejemplo utilizando un único elemento `div` al que aplicaremos un estilo. El resultado producido será analizado con la ayuda de una figura en la que hemos modelado el orden de apilado de los elementos del `div` en una disposición tridimensional que nos ayudará a comprenderlo.

Supongamos el siguiente código HTML.

```
<div>
<p>Este es el contenido de nuestra caja.</p>
<p>Este es el contenido de nuestra caja.</p>
<p>Este es el contenido de nuestra caja.</p>
<p>Este es el contenido de nuestra caja.</p>
</div>
```

El código HTML anterior tiene el siguiente estilo

```
div {
    background-color: #be4061;
    background-image: url('cabeza.jpg');
```

```
    border: 10px solid #e7a219;
    margin: 10px;
    padding: 20px;
}

p {
    margin: 0 0 20px 0; /*margen inferior de 20 px para el
párrafo*/
    padding: 0;
}
```

El código anterior generará una caja como la que muestra la figura siguiente, en la que adicionalmente se ha dado color a los elementos transparentes (margen y relleno) solo para hacerlos visibles.

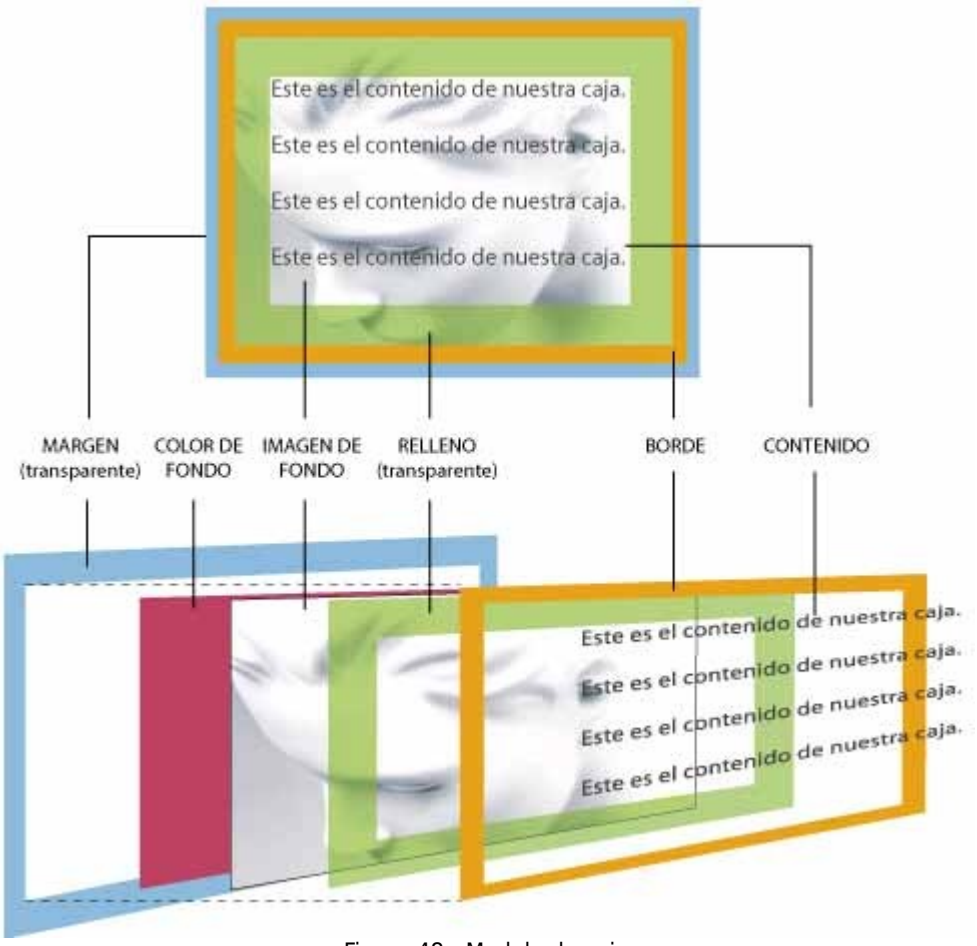


Figura 48.- Modelo de caja

Un detalle interesante que puede apreciarse en la representación tridimensional en que la capa superior del apilamiento no es el borde, como podría suponerse intuitivamente. La capa situada encima de todas las demás es la de Contenido.

Cada elemento HTML está modelado por una caja como la anterior, cajas que se pueden ajustar usando CSS. A continuación se explican las propiedades que se aplican para definir el modelado de caja de CSS.

14.4.1 Margen

Todo elemento tiene cuatro lados: derecho, izquierdo, superior e inferior. La propiedad `margin` hace referencia a la distancia desde cada lado respecto al elemento colindante, o respecto a los bordes del documento.

En un primer ejemplo, veremos cómo definir los márgenes del documento en sí, es decir, del elemento `body`. La imagen siguiente muestra cómo queremos que sean los márgenes de nuestras páginas.

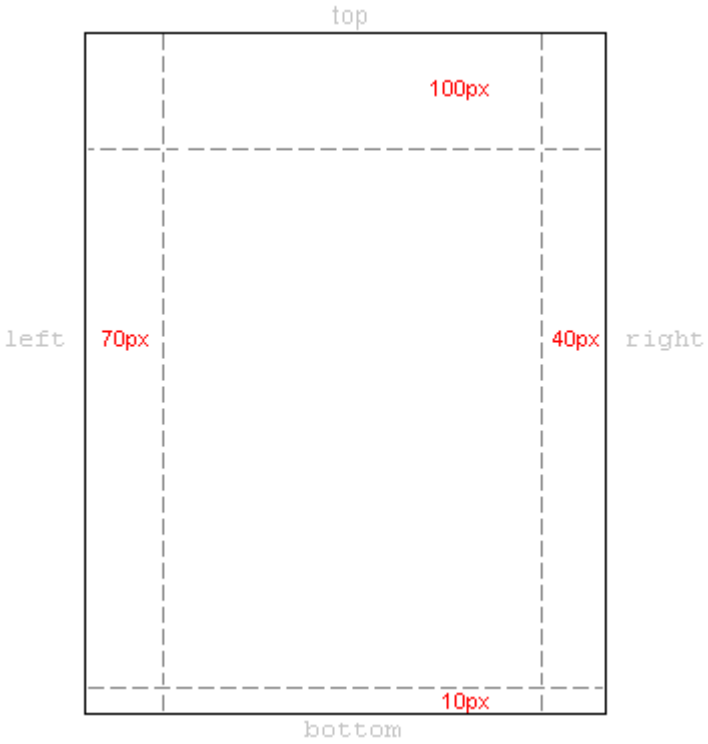


Figura 49.- Márgenes del documento

El código CSS necesario para esto es el siguiente:

```
BODY {
    margin-top: 100px;
    margin-right: 40px;
    margin-bottom: 10px;
    margin-left: 70px;
}
```

O podrías elegir usar la versión combinada de `margin`, que queda como más elegante:

```
BODY {
    margin: 100px 40px 10px 70px;
}
```

Se puede establecer los márgenes de casi todos los elementos del mismo modo. Por ejemplo, podemos elegir definir márgenes para todos los párrafos de texto marcados con el elemento `p`

```
BODY {
    margin: 100px 40px 10px 70px;
}
P {
    margin: 5px 50px 5px 50px;
}
```

Si se aplica los márgenes a una capa, elemento `div`, entonces indica el margen que se deja entre el contenido de la capa y los bordes de la misma.

14.4.2 Relleno

La propiedad `padding` puede entenderse como "relleno". Esto tiene sentido puesto que el relleno no afecta a la distancia de un elemento respecto a otros elementos, sino que sólo define la distancia interior entre el borde y el contenido del elemento.

El uso de la propiedad `padding` se puede ilustrar viendo un sencillo ejemplo en el que todos los títulos tienen diferentes colores de fondo. Al definir el `padding` para los títulos, cambiamos la cantidad de "relleno" que habrá alrededor del texto en cada uno de ellos:

```
h1 {
    background: yellow;
    padding: 20px 20px 20px 80px;
}

h2 {
    background: orange;
    padding-left:120px;
}
```

14.4.3 Bordes

La propiedad `border` se vio en detalla en el capítulo dedicado a tablas en el que lo empleabamos para poner bordes a las celdas y la tabla. Lo explicado entonces es aplicable a cualquier elemento a que queramos poner un borde: encabezados, párrafos, capas, etc.

Aquí vamos a explicar otras propiedades nuevas de CSS3 que tienen que ver con los bordes y que son las siguientes:

14.4.3.1 Bordes con esquinas redondeados

En este apartado vamos a explicar cómo realizar bordes redondeados con CSS3. Tenemos la propiedad `border-radius`, que permite definir bordes redondeados en las esquinas, especificando las medidas del radio que deben darse a la curva de las esquinas. El navegador Internet Explorer soporta esta propiedad a partir de su versión 9. Veamos otro ejemplo

Por el momento, para navegadores Mozilla y WebKit que son los primeros en adaptarse a CSS3, podemos utilizar el atributo `border-radius` de la siguiente manera:

```
div {
    border: 1px solid #000000;
    border-radius: 7px;
    padding: 10px;
}
```

Con esto conseguimos que todas los elementos `div` tengan un borde redondeado en las esquinas de radio de 7 píxeles. Pero la propiedad `border-radius` tiene otras posibles configuraciones, en la que se pueden definir los valores para el radio de las cuatro esquinas por separado. De esta manera:

```
border-radius: 7px 27px 100px 10px;
```

Así estaríamos definiendo un borde redondeado con radio de 7 píxeles para la esquina superior izquierda, luego 27 píxeles para la esquina superior derecha, de 100 píxeles para la inferior derecha y 10 píxeles para la inferior izquierda.

14.4.3.2 Sombra

El atributo `box-shadow` se emplea para crear sombra a los elementos. Este requiere varios valores para especificar las características de la sombra, como difuminado, separación de la sombra y la propia caja o color. La sintaxis es

```
box-shadow: <horizontal> <vertical> <difuminado> <color>
```

Por orden de aparición, los valores que se indican en `box-shadow` son:

- ✔ Desplazamiento horizontal de la sombra: La sombra de un elemento suele estar un poco desplazada con respecto al elemento que la produce y su posición será en función del ángulo con el que llegue la luz. Si este valor es positivo quiere decir que la sombra aparecerá desplazada a la derecha. Si quisiéramos que apareciera hacia la izquierda del elemento original que la produce, pondríamos un valor negativo a esta propiedad. Cuanto más desplazamiento tenga una sombra, el elemento que la produce parecerá que está más separado del lienzo de la página.
- ✔ Desplazamiento vertical de la sombra: El segundo valor que colocamos en el atributo `box-shadow` es el desplazamiento vertical de la sombra con respecto a la posición del elemento que la produce. Este valor es similar al desplazamiento horizontal. Valores positivos indican que la sombra aparecerá hacia abajo del elemento y valores negativos harán que la sombra aparezca desplazada un poco hacia arriba.
- ✔ Difuminado: El tercer valor indica cuánto queremos que esté difuminado el borde de la sombra. Si el difuminado fuera cero, querría decir que la sombra no tiene ningún difuminado y aparece totalmente definida. Si el valor es mayor que cero quiere decir que la sombra tendrá un difuminado de esa anchura.
- ✔ Color de la sombra: El último valor que se indica en el atributo `box-shadow` es el color de la sombra. Generalmente las sombras en el mundo real tienen un color negro o grisaceo, pero con CSS3 podremos indicar cualquier gama de color para

hacer la sombra, lo que nos dará bastante más versatilidad a los diseños gracias a la posible utilización de sombras en distintos colores, que puedan combinar mejor con nuestra paleta.

En el ejemplo siguiente vamos a crear una capa con una sombra desplazada 5 píxeles a la derecha, 9 píxeles arriba, 3 píxeles de difuminado y con color gris.

```
#caja_con_sombra {
    background-color: cyan;
    width: 300px;
    height: 400px;
    padding: 10pt;
    border: 1px solid black;
    box-shadow: 5px -9px 3px gray;
}
```

En el siguiente ejemplo es para una sombra un poco menor, también desplazada hacia abajo y a la derecha y con un difuminado de 2 píxeles. Además hemos indicado un color amarillo claro para la sombra, por lo que, para verla bien, tendríamos que colocar este elemento sobre un fondo oscuro.

```
#sombraclara{
   width: 200px;
   padding: 10px;
   background-color: #999;
   color: #fff;

   box-shadow: 2px 2px 2px #ffc;
}
```

En este tercer ejemplo tenemos un caso curioso de sombra, pues está aplicada sobre un elemento que tiene las esquinas redondeadas con CSS 3. Así pues, la sombra también debe tener las sombras redondeadas, para ajustarse al elemento que la produce.

```
#sombraredondeada{
   background-color: #090;
   color: #fff;
   width: 400px;
   padding: 10px;
   -moz-border-radius: 7px;
   -webkit-border-radius: 7px;

   box-shadow: 15px -10px 3px #000;
}
```

14.4.3.3 Borde con imágenes de fondo

Vamos a hablar rápidamente sobre uno de los nuevos atributos de las CSS 3, que sirve para personalizar los bordes de los elementos HTML con imágenes. Para poner una imagen como borde hay que especificar las siguientes propiedades:

- ✔ `border-image-source`: Para indicar la URL de la imagen que vamos a utilizar

como borde.

- ✔ `border-image-slice`: Indica el espacio de la imagen que será visible como borde, en los cuatro lados del elemento, es decir, top, right, bottom y left. Si indicamos un solo valor significa que es el mismo en los cuatro lados.
- ✔ `border-image-width`: Para indicar la anchura del borde.
- ✔ `border-image-outset`: Nos sirve para indicar el área en la que la imagen de borde se extiende más allá del área del elemento, en los 4 lados del mismo.
- ✔ `border-image-repeat`: Permite indicar si se desea que se repita la imágen del borde haciendo un mosaico (valor `repeat`), que se estire (valor `stretch`) o que se redondee (valor `round`).

Además disponemos de la propiedad abreviada `border-image` la cual tiene la siguiente sintaxis:

```
border-image: <imagen> <arriba> <derecha> <abajo> <izquierda>
<repeticion>;
```

Por ejemplo, tenemos el siguiente elemento HTML:

```
<div id="capaborde">
Esta capa le voy a poner un borde arriba
</div>
```

Y ahora podríamos aplicar estilos para crear un borde en la imagen:

```
#capaborde{
    border-image: url(sello.png) 10% 10% 10% 10% stretch;

    padding:20px;
    width: 100px;
}
```

La imágen que estamos utilizando como borde es la siguiente:

Figura 50.- Imagen para borde

Analicemos la propiedad `border-image`.

```
border-image: url(sello.png) 10% 10% 10% 10% stretch;
```

En primer lugar hemos puesto la imagen almacenada en el archivo sello.png. Hay que usar solamente el 10% de la imagen en cada una de las coordenadas. Como las cuatro son

iguales, tabmién podríamos haber puesto solamente 10% una vez. Si este tamaño es diferente para cada uno hay que especificarlos en el orden de las agujas del reloj, es decir, arriba, derecha, abajo e izquierda. Se pueden expresar en porcentajes o sin ninguna unidad de medida para expresar píxeles exactos. La última parte, donde hemos puesto el valor `stretch`, determina cómo se van a escalar y cortar los bordes de los lados y superior e inferior de nuestro elemento. Hay tres posibles valores:

- ✔ `stretch` (estirar)
- ✔ `repeat` (repetir)
- ✔ `round` (redondear)

Si especificamos sólo un valor es como poner dos veces el mismo. El valor por defecto es “stretch”.

14.4.3.4 Contorno

Una nueva propiedad de la versión 3 de CSS nos permite establecer contornos sobre los elementos. Un contorno es una línea alrededor del elemento, pero a diferencia del borde, los contornos no ocupan espacio y no tienen por que ser rectangulares. Además, el contorno no se puede diferenciar arriba, abajo, izquierda o derecha. Si se establece, será el mismo para todos.

Disponemos de tres propiedades para establecer un contorno:

- ✔ `outline-color`, color del contorno.
- ✔ `outline-style`, estilo del contorno. Se admiten los mismos valores que para la propiedad `border-style`.
- ✔ `outline-width`, ancho del contorno.

Además disponemos también de la propiedad abreviada `outline` con la siguiente sintaxis.

```
outline: color style width;
```

Por ejemplo

```
div {
    outline: darkblue solid 1px;
    padding: 10px;
}
```

14.4.4 Dimesión

Con la propiedad `width` se puede definir la anchura concreta de un elemento. El sencillo ejemplo que sigue nos proporciona una caja en la que se puede introducir texto:

```
.box {
```

```
    width: 200px;
    border: 1px solid black;
    background: orange;
}
```

La altura de un elemento queda definida por defecto por el contenido de la misma. Se puede influir en la altura de un elemento con la propiedad `height`. Por ejemplo, probemos a fijar la altura de la caja en 500px:

```
.box {
    height: 500px;
    width: 200px;
    border: 1px solid black;
    background: orange;
}
```

Sin embargo, si a los elementos no se les establece altura o anchura tomarán el espacio que necesiten para visualizar su contenido. Para controlar la altura y anchura mínima y máxima disponemos de las siguientes propiedades.

- ✔ `min-height`.- Establece la altura mínima del elemento.
- ✔ `min-width`.- Establece la anchura mínima del elemento.
- ✔ `max-height`.- Establece la altura máxima del elemento.
- ✔ `max-width`.- Establece la anchura máxima del elemento.

Sus posibles valores y unidades de medida son similares a los de las propiedades `width` y `height`.

14.4.5 Desborde

La propiedad `overflow` sirve en el modelado de cajas para indicar al navegador qué es lo que debe hacer con el contenido que no cabe dentro de una capa, según las dimensiones que se le han asignado.

Como sabemos, a las capas (elementos `div`) podemos asignarles un tamaño, en anchura y altura. Pero muchas veces el contenido que colocamos en la capa sobrepasa el espacio que hemos destinado a ella. Entonces lo que suele ocurrir es que la capa crece lo suficiente para que el contenido colocado dentro quepa. Habitualmente las capas crecen en altura, por lo que a más contenido más tamaño tendrá en altura. Este es un comportamiento que podemos alterar con el uso de la propiedad `overflow`.

Dicho de otro modo, `overflow` permite que se recorte el contenido de una capa, para mostrar únicamente el contenido que quepa, según sus dimensiones. Para acceder al contenido que no se muestra, porque no cabe en la capa, se puede configurar `overflow` para que aparezcan unas barras de desplazamiento.

Sus posibles valores son

- ✔ `visible`: Este valor indica que se debe mostrar todo el contenido de la capa, aunque no quepa en tamaño con la que la hemos configurado. La capa tiene el tamaño marcado, pero el contenido se sigue viendo, aunque fuera del espacio donde de la capa, pudiendo superponerse a un texto o imagen que hubiera debajo.
- ✔ `hidden`: Este valor indica que los contenidos que, por el tamaño de la capa, no quepan en la misma, se deben recortar. Por ello, la capa tendrá siempre el tamaño configurado, pero los contenidos en ocasiones podrán no verse por completo.
- ✔ `scroll`: Este valor indica que la capa debe tener el tamaño que se haya configurado inicialmente y que además se deben mostrar unas barras de desplazamiento, para mover el contenido de la capa dentro del espacio de la misma. Las barras de desplazamiento siempre salen, se requieran o no.
- ✔ `auto`: Con este valor también se respetarán las dimensiones asignadas a una caja. El contenido será recortado, pero aparecerán las barras de desplazamiento para moverlo. Sin embargo, en este caso las barras de desplazamiento podrán salir o no, depende de si son necesarias o no para ver todo el contenido de la capa.

Así pues, la propiedad `overflow` nos permitirá tener un mayor control sobre los espacios que destinamos a cada caja de nuestro diseño. Es muy utilizado para mostrar textos largos, que se desean integrar dentro de otro texto o una interfaz donde no tenemos espacio disponible para colocarlos o no deseamos que crezcan más de la cuenta. Por ejemplo

```
<div style="overflow: auto; width: 300px; height: 100px;
background-color:#ededed; border: 1px solid #990000;">
CONTENIDO....
</div>
```

En este ejemplo si el contenido supera las dimensiones de la capa aparecerá unas barras de desplazamiento.

Con CSS3 han aparecido las propiedades `overflow-x` y `overflow-y` para controlar el contenido de la capa a lo ancho y a lo largo.

14.5 Elementos flotantes

Los elementos se pueden hacer flotar a la derecha o a la izquierda usando la propiedad `float`. Es decir, que la caja con su contenido flota bien a la derecha o la izquierda de un documento (o de la caja contenedora). La siguiente imagen muestra este principio

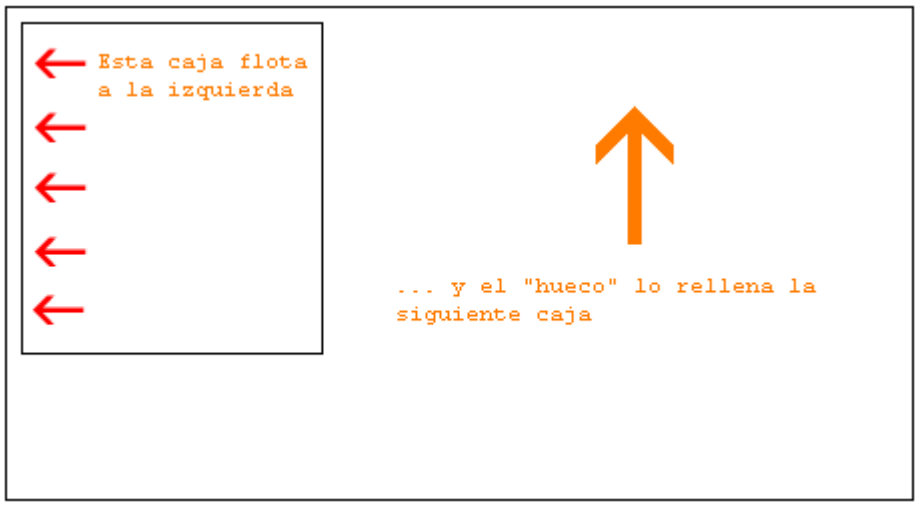


Figura 51.- Elementos flotantes

Por ejemplo, si quisiéramos texto con ajuste de línea alrededor de una imagen, el resultado sería el siguiente

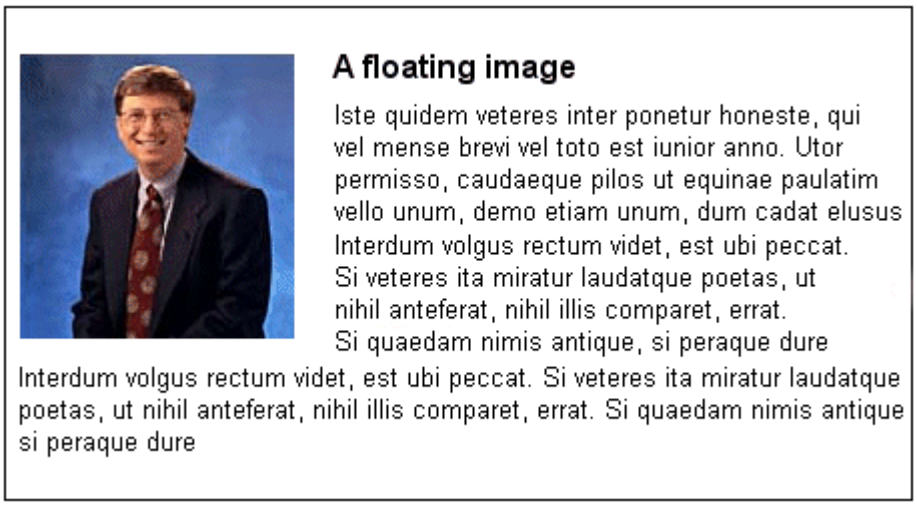


Figura 52.- Imagen flotando a la izquierda

El código HTML del ejemplo anterior es el siguiente:

```
<div id="picture">
    <img src="bill.jpg" alt="Bill Gates">
</div>
<h3>A floating image</h3>
<p>Iste quidem veteres inter ponetur … </p>
```

Para conseguir que la imagen flote a la izquierda y el texto se ajuste a su alrededor, sólo hay que definir el ancho de la caja que rodea la imagen y, después de eso, fijar la propiedad `float` con el valor `left`.

```
#picture {
    float:left;
    width: 100px;
}
```

Una propiedad muy relacionada con `float` es `clear` se usa para controlar cómo se comportarán los elementos que siguen a los elementos flotantes de un documento.

Por defecto, los elementos siguientes se mueven hacia arriba para rellenar el espacio disponible que quedará libre al flotar una caja hacia un lado. Echa un vistazo al ejemplo anterior en el que el texto se desplaza de forma automática hacia arriba junto a la imagen de Bill Gates.

La propiedad `clear` puede tomar los valores: `left`, `right`, `both` o `none`.

- ✔ Si un elemento tiene establecido el valor en `left` significa que no puede tener elementos flotantes a su izquierda, es decir, aparecerá al borde izquierdo de la página.
- ✔ Si un elemento tiene establecido el valor en `right` significa que no puede tener elementos flotantes a su derecha, es decir, aparecerá sin nada a su derecha.
- ✔ Si un elemento tiene establecido el valor en `both` significa que no puede tener elementos flotantes a su derecha e izquierda. Estará solo en el borde izquierdo de la página.
- ✔ Si un elemento tiene establecido el valor en `none` (por defecto) significa que podrá tener elementos flotantes a su izquierda y derecha.

Si en el ejemplo anterior quisieramos evitar que el párrafo rodeara a la imagen tendríamos que asignar al párrafo la propiedad `float` con valor `both`.

```
#picture {
    float: left;
    width: 100px;
}

.floatstop {
    clear:both;
}
```

14.6 Visibilidad

Para definir si un elemento se visualiza o no tenemos dos propiedades.

La propiedad `visibility` se utiliza para definir la visibilidad de un elemento. Con esta propiedad podemos decir que ciertos elementos de la página sean visibles o invisibles, pero atención, aunque un elemento sea invisible, continúa ocupando espacio en la página. Si queremos que no sea invisible y no se le reserve espacio en la página, hay que utilizar la propiedad `display`.

Los posibles valores de `visibility` son:

- ✔ `visible`, que hace que el elemento se vea (valor por defecto).
- ✔ `hidden`, que hace que el elemento sea invisible, aunque continúe ocupando espacio.

Los posibles valores de `display` son:

- ✔ `none`, elemento no se visualiza y no ocupa espacio.
- ✔ `block`, el elemento se visualiza como un elemento de bloque.
- ✔ `inline`, el elemento se visualiza en línea.
- ✔ `inline-block`, el elemento se visualiza en linea pero se comporta como un bloque.

Esta propiedad tiene más valores pero son utilizados en aplicaciones específicas. Para más información consultar la especificación del W3C.

14.7 Opacidad

Con la propiedad opacity establecemos el nivel de opacidad de un elemento. Admite valores entre 0 (completamente transparente, es decir, el elemento no se ve) hasta 1 (completamente opaco, el elemento se ve completamente). Valores intermedios dan al elemento un grado de transparencia. Veamos un ejemplo

```
div {
    border: 1px solid #000000;
    opacity: 0.5;
    padding: 10px;
}
```

En el ejemplo anterior, la capa tiene la mitad de la opacidad o de transparencia, según se mire.

14.8 Posicionamiento

Con posicionamiento CSS, se puede colocar un elemento en el lugar exacto que se quiera de la página. Antes de ver el posicionamiento debemos de comprender el sistema de coordenadas que emplea CSS. Imagina la ventana de un navegador como un sistema de coordenadas como el siguiente.

Figura 53.- Sistema de coordenadas

Los principios que rigen el posicionamiento CSS consisten en que se puede colocar cualquier caja en cualquier lugar del sistema de coordenadas.

Las propiedades para establecer el posicionamiento de un elemento son las siguientes:

- ✔ `position`.- Indica el tipo de posicionamiento.
- ✔ `top`.- Establece la coordenada superior del elemento.
- ✔ `left`.- Establece la coordenada izquierda del elemento.
- ✔ `rigth`.- Establece la coordenada derecha del elemento.
- ✔ `bottom`.- Establece la coordenada inferior del elemento.

Sin embargo, la posición de los elementos en una página dependerá de su tipo de posicionamiento y de donde se encuentre colocados en el código HTML. Vamos a ver como se comportarían los elementos en sus diferentes tipos de posicionamiento.

14.8.1 Posicionamiento estático

El valor `static` es el valor predeterminado la propiedad `position` y el posicionamiento normal de los elementos en la página. Quiere decir que los elementos se colocarán según el flujo normal del HTML, es decir, según estén escritos en el propio código HTML. Por decirlo de otra manera, `static` no provoca ningún posicionamiento especial de los elementos y por tanto, los atributos `top`, `left`, `right` y `bottom` no se tendrán en cuenta.

14.8.2 Posicionamiento absoluto

Para posicionar un elemento de forma absoluta, la propiedad position se establece con el valor `absolute`. Posteriormente puedes usar las propiedades `left`, `right`, `top`, y `bottom` para colocar la caja.

El valor `absolute` en la propiedad `position` permite posicionar elementos de manera absoluta, esto es de manera definida por valores de los atributos `top`, `left`, `bottom` y `right`, que indican la distancia con respecto a un punto. Las capas o elementos con posicionamiento absoluto quedan aparte del flujo normal del HTML, quiere decir esto que no se afectan por el lugar donde aparezcan en el código HTML y tampoco afectan ellas a otros elementos que sí que formen parte del flujo normal del HTML.

Los valores `top`, `left`, `bottom` y `right` son una distancia relativa al primer elemento contenedor que tenga un valor de position distinto de `static`. Si todos los contenedores donde esté la capa posicionada con position `absolute` (todos sus padres hasta llegar al elemento `body`) son `static`, simplemente se posiciona con respecto al lado superior de la página, para el caso de `top`, el inferior para `bottom`, del lado izquierdo para `left` o el derecho, en el caso de utilizar `right`.

Como ejemplo de posicionamiento absoluto, vamos a colocar 4 cajas en cada esquina del documento.

```
#box1 {
    position:absolute;
    top: 50px;
    left: 50px;
}

#box2 {
    position:absolute;
    top: 50px;
    right: 50px;
}

#box3 {
    position:absolute;
    bottom: 50px;
    right: 50px;
}

#box4 {
    position:absolute;
    bottom: 50px;
    left: 50px;
}
```

14.8.3 Posicionamiento relativo

Para posicionar un elemento de forma relativa, la propiedad `position` se establece como `relative`. La diferencia entre posicionamiento absoluto y relativo consiste en cómo se calcula la posicion.

La posición para un elemento que se posiciona de forma relativa se calcula desde la posición original en el documento en el caso de que tuviera posicionamiento estático. Esto significa que se mueve el elemento hacia la derecha, la izquierda, arriba o abajo. De este modo, el elemento sigue ocupando espacio en el documento después de haberse

posicionado.

Veamos el siguiente ejemplo

```
<h1>Título 1</h1>
<div style="background-color: #606; color:#ffc; padding:10px;
text-align: center; width: 300px;">
Capa 1 de prueba
</div>

<div style="position: relative; width: 300px; padding: 10px;
background-color: #066; color:#ffc; top:100px; left: 30px;">
Capa 1 de prueba con posicionamiento relativo<BR>
Se tiene en cuenta esta capa para posicionar las siguientes.
</div>

<h1>Título 1</h1>
```

El resultado sería

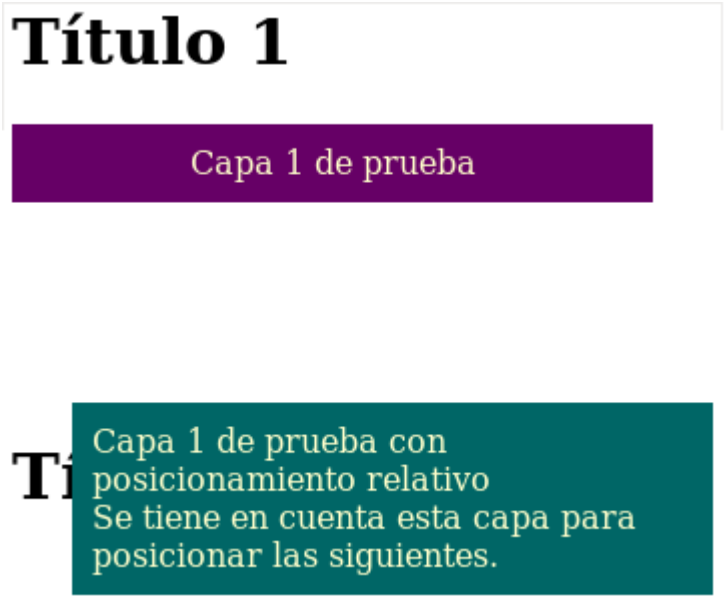


Figura 54.- Posicionamiento relativo

Podemos ver que la segnda capa se ha desplazado 30 píxeles desde la izquierda y 100 píxeles hacia abajo.

14.8.4 Posicionamiento fijo

Para posicionar un elemento de forma fija, la propiedad `position` se establece como `fixed`. Esto sirve para posicionar una capa con posicionamiento absoluto, pero su posición final será siempre fija, es decir, aunque se desplace el documento con las barras de desplazamiento del navegador, siempre aparecerá en la misma posición.

El lugar donde se "anclará" la capa siempre es relativo al cuerpo (el espacio disponible del navegador para la página). Si utilizamos `top` y `left`, estaremos marcando su posición con respecto a la esquina superior izquierda y si utilizamos `bottom` y `right` su posición

será relativa a la esquina inferior derecha. Veamos un ejemplo.

```
<div style="position: fixed; width: 300px; height: 140px; top:
100px; left: 30px; background-color: #ff8800; color: #fff;
padding: 15px;z-index: 1;">
Esta capa tiene posicionamiento fixed.
<br>
<br>
Me permite especificar top y left para colocarla con respecto a
la esquina superior izquierda.
</div>

<div style="position: fixed; width: 700px; height: 30px;
padding: 10px; background-color: #d0f; top: 150px; left: 10px;
z-index: 2;">Posicionamiento fixed</div>

<h1>Título 1</h1>

<div style="position: fixed; width: 100px; height: 30px;
padding: 10px; background-color: #0df; bottom: 10px; right:
10px; z-index: 4;">Posicionamiento fixed</div>
<br>
<br>
<br>
<br>
Pongo texto para que se vea!!
<br>
<br>
<br>
<br>
Esto hace desplazamiento, con tanto BR
<br>
<br>
...
<br>
```

Se puede ver que hay varias capas con position: `fixed` y un montón de elementos salto de línea `br` para que la página pueda tener un desplazamiento. Si vemos la página en marcha y nos desplazamos hacia abajo con la barra de desplazamiento, veremos que las capas fixed siempre mantienen la misma posición.

14.9 Apilamiento

CSS funciona sobre tres dimensiones: altura, anchura y profundidad. En apartados anteriores hemos examinado las dos primeras dimensiones. En este apartado, aprenderemos cómo hacer que diferentes elementos se conviertan en capas. En pocas palabras, esto hace referencia al orden en que los elementos se superponen unos con respecto a otros.

Para tal propósito, se puede asignar a cada elemento un número por medio de la propiedad `z-index`. El sistema consiste en que el elemento con un número mayor se superpone al elemento con un número menor.

Supongamos que estamos jugando al poquer y tenemos una escalera de color. La mano se puede presentar de tal manera que cada carta tiene un número asignado por medio de `z-index`:

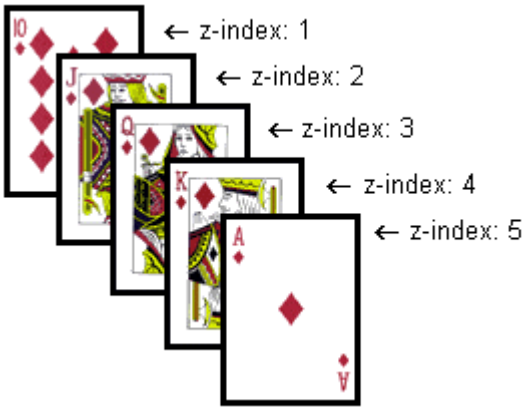


Figura 55.- Elementos apilados

En este caso, los números son consecutivos (yendo del 1 al 5), aunque se puede lograr el mismo resultado usando cinco numeros diferentes. Lo importante es la secuencia cronológica de los números (el orden). El código del ejemplo de las cartas quedaría así:

```
#diez_de_diamantes {
    position: absolute;
    left: 100px;
    top: 100px;
    z-index: 1;
}

#sota_de_diamantes {
    position: absolute;
    left: 115px;
    top: 115px;
    z-index: 2;
}

#reina_de_diamantes {
    position: absolute;
    left: 130px;
    top: 130px;
    z-index: 3;
}

#rey_de_diamantes {
    position: absolute;
    left: 145px;
    top: 145px;
    z-index: 4;
}

#as_de_diamantes {
    position: absolute;
    left: 160px;
```

```
    top: 160px;
    z-index: 5;
}
```

El método es relativamente sencillo pero las posibilidades que ofrece son múltiples. Es posible colocar imágenes sobre el texto, texto sobre imágenes, etc.

15 Listas

Una lista es un conjunto de elementos con algún tipo de relación entre ellas. Por ejemplo, los ingredientes de una receta de cocina o una lista de la compra. Una forma habitual de organizar la información en un texto es hacer uso de:

- ✔ Listas de términos, precedidos de un punto o cualquier otro símbolo.
- ✔ Listas numeradas alfabéticamente, o con números arábicos o romanos, y cuyos elementos se diferencia con este número o letra.
- ✔ Listas simples sin indicaciones o separadores.
- ✔ Términos y sus definiciones.
- ✔ Listas en las que los elementos se distribuyen en columnas.
- ✔ Listas de varios de los tipos anteriores anidadas las unas en las otras

Todos estos tipos de listas, son las que podremos definir con ayuda de las directivas que HTML dispone para ello. Todas se explican a continuación, pese a ser empleadas solamente unas cuantas de ellas.

15.1 Listas sin numerar

El elemento `ul` se emplea para crear una lista donde sus elementos no tienen ninguna relación de orden. Es decir, cambiar el orden de los elementos de la lista no altera su significado. Dentro de este introduciremos un elemento `li` por cada uno de la lista. Veamos un ejemplo:

```
<h2>Ingredientes para una buena tortilla de patatas</h2>
<ul>
    <li>1 Kg de patatas.
    <li>Aceite de oliva
    <li>2 cebollas medianas.
    <li>2 pimientos verdes
    <li>4 huevos
    <li>sal
</ul>
```

El resultado sería

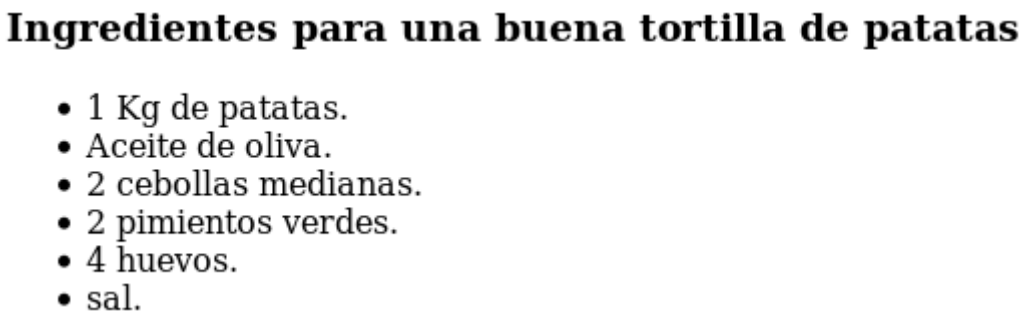


Figura 56.- Lista sin orden

Esto nos define una lista de ingredientes, con un *bullet* o punto al lado del nombre de cada ingrediente. Debe de hacerse notar que el elemento `li` hace referencia siempre a un elemento de un tipo de lista, y que tras el siguiente `li`, hace un retorno de carro, pero no deja una línea en blanco.

15.2 Listas ordenadas

Son semejantes a las anteriores, sólo que en lugar del *bullet* o punto negro que aparece por defecto, en este tipo de listas los elementos aparecen numerados con algún tipo de numeración. Por defecto, números arábigos.

En este caso el elemento que delimita la lista es `ol` (*Ordered List*, lista ordenada), y cada elemento es delimitado por un elemento `li`. Siguiendo con el ejemplo anterior podemos escribir los ingredientes de la tortilla de patatas.

```
<h2>Sigue estos pasos para hacer la tortilla</h2>
<ol>
    <li>Pelar y cortar las patatas.
    <li>Poner el aceite a hervir en una sartén.
    <li>Pelar y cortar las cebollas y los pimientos.
    <li>Mezclar las patatas con las cebollas y los pimientos.
    <li>Añadir a la mezcla sal.
    <li>Freir todo en la sartén hasta que las patatas doren.
    <li>Batir los huevos en un bol.
    <li>Añadir las patatas frintas a los huevos y remover bien.
    <li>Poner la mezcla en una sartén.
    <li>Cuando la tortilla cuaje por un lado darle la vuelta.
    <li>Cuando la tortilla cuaje por el otro lado servir.
</ol>
```

El resultado será el siguiente

Sigue estos pasos para hacer la tortilla

1. Pelar y cortar las patatas.
2. Poner el aceite a hervir en una sartén.
3. Pelar y cortar las cebollas y los pimientos.
4. Mezclar las patatas con las cebollas y los pimientos.
5. Añadir a la mezcla sal.
6. Freir todo en la sartén hasta que las patatas doren.
7. Batir los huevos en un bol.
8. Añadir las patatas frintas a los huevos y remover bien.
9. Poner la mezcla en una sartén.
10. Cuando la tortilla cuaje por un lado darle la vuelta.
11. Cuando la tortilla cuaje por el otro lado servir.

Figura 57.- Lista ordenada

Vemos que por defecto los elemento de la lista se numeran con números arábigos.

15.3 Listas de definiciones

También conocidas como *glossary list* (listas de glosario). Se trata de una lista de 2 niveles: cada elemento se descompone en un nombre que ocupa la primera línea, y una definición que ocupa la siguiente.

La lista está delimitada por el elemento `dl` (*definition list*, lista de definición). En estas listas el elemento a definir debe de llevar la directiva `dt` en lugar de `li` anterior, y la definición de este elemento la directiva `dd`.

Un ejemplo de este tipo de listas sería:

```
<dl>
    <dt>Coche
    <dd> Vehículo de 4 o más ruedas que sirve para desplazar
cosas, personas o animales.
    <dt>Moto
    <dd>Vehículo de dos, tres o cuatro ruedas que sirve para
desplazar una o dos personas o animales.
</dl>
```

El resultado es

Coche
: Vehículo de 4 o más ruedas que sirve para desplazar cosas, personas o animales.

Moto
: Vehículo de dos, tres o cuatro ruedas que sirve para desplazar una o dos personas o animales.

Figura 58.- Lista de definiciones

15.4 Lista multinivel

Hay veces que necesitamos subdividir un elemento de lista, es decir, de un elemento parte otra lista. Con mucha frecuencia nos encontramos con listas de varios niveles, o también denominadas listas anidadas.

¿Cómo hacerlo?, sólo hay que tener presente que un elemento de una lista puede ser cualquier otro elemento HTML, por ejemplo otra lista. Este es un ejemplo de lista anidada:

```
<h2>Contenidos del curso</h2>
<ol>
    <li>Introducción a la informática
    <li>LibreOffice
    <ul>
         <li>Writer
         <li>Calc
         <li>Impress
    </ul>
    <li>Internet
    <ol>
         <li>Servicio WWW
         <ul>
              <li>Navegadores web
              <li>Búsquedas en Internet
         </ul>
         <li>Correo electrónico
         <li>Redes sociales
    </ol>
    <li>Amenazas en Internet
</ol>
```

Como podemos observar si sustituimos un elemento `li` por un elemento `ol` o `ul` entonces estamos creando un nuevo nivel de la lista. El resultado sería el siguiente

Contenidos del curso

1. Introducción a la informática
2. LibreOffice
    - Writer
    - Calc
    - Impress
3. Internet
    1. Servicio WWW
        - Navegadores web
        - Búsquedas en Internet
    2. Correo electrónico
    3. Redes sociales
4. Amenazas en Internet

Figura 59.- Lista multinivel

15.5 Formato de listas

Más allá de la marcación del documento HTML, con la ayuda de CSS es posible cambiar por completo la apariencia de las listas y crear elementos atractivos y de gran impacto visual, tanto para un simple listado como para la creación de una barra de navegación.

15.5.1 Estilo de la viñeta

Podemos establecer el estilo de la viñeta en las listas sin numerar con la propiedad `list-style-type`. Su sintaxis es la siguiente

```
ul | ol { list-style-type:  none |
                         disc |
                         circle |
                         square |
                         decimal |
                         decimal-leading-zero |
                         lower- roman |
                         upper-roman |
                         lower-alpha |
                         upper-alpha |
                         lower-greek |
                         lower-latin |
                         upper-latin |
                         hebrew |
                         armenian |
                         georgian |
                         cjk-ideographic |
                         hiragana |
                         katakana |
                         hiragana-iroha |
                         katakana-iroha }
```

15.5.2 Imágenes como viñetas

En el ejemplo anterior se demostró que es posible variar el estilo de las listas usando los valores que nos da CSS de una forma muy sencilla. Asimismo, en muchas ocasiones, nuestros diseños requerirán de soluciones más estéticas que un simple círculo. Para ello CSS nos da la opción de incluir imágenes en nuestras listas utilizando dos métodos: reemplazar la viñeta por una imagen o añadirla como fondo. La segunda opción suele ser más práctica ya que es posible determinar la ubicación de la imagen desplazándola horizontal o verticalmente e intentar así una visualización similar en distintos navegadores.

Para sustituir la viñeta por una imagen utilizamos la propiedad list-style-image de la siguiente manera.

```
ul | ol { list-style-image: url(url_imagen); }
```

Esto hace que se reemplace la viñeta de la lista por la imagen indicada en el fichero `url_imagen`. Naturalmente la imagen de la viñeta debe ser pequeño, de unos 16 o 22 pixeles como máximo.

Hay que tener en cuenta que cada navegador presenta en pantalla los elementos con algunas pequeñas diferencias entre sí. En esta captura de pantalla se puede observar como varía la posición de la viñeta (y el interlineado también) en tres navegadores distintos:

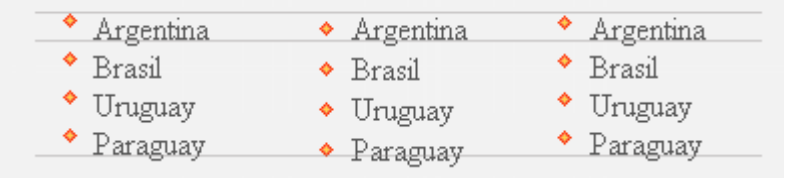


Figura 60.- Viñetas personalizadas

En el ejemplo anterior se tomaron los valores por defecto de cada navegador, es decir, no se modificó el tamaño ni tipo de letra ni el interlineado de la lista.

La otra opción es anular la viñeta y añadir una imagen de fondo a los elementos `li`. Esto permite desplazar la imagen en el eje X e Y. Por ejemplo

```
li{
    list-style: none;
    background-image: url('rombo.png');
    background-position: left bottom;
    background-repeat: no-repeat;
    padding-left: 15px;
}
```

El resultado sería el siguiente

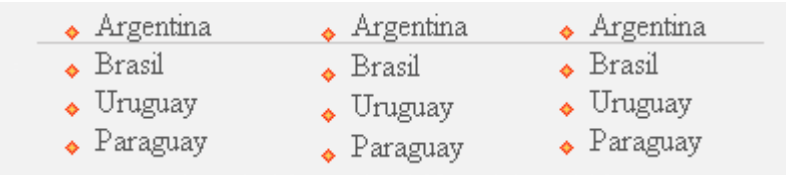


Figura 61.- Viñetas mediante imagen de fondo

En el ejemplo se observa que ahora la diferencia entre navegadores es de apenas 1 pixel, lo que ya es un logro.

La primera línea anula la viñeta de la lista, la segunda indica el la imagen a utilizar, la tercera la posición con respecto al ítem de la lista (a la izquierda y abajo), la cuarta dice que el fondo no se repita y la quinta desplaza la palabra 15 píxeles hacia la derecha para que no tape la imagen. Esto se podría haber escrito en 3 líneas usando el método abreviado de la siguiente manera.

```
li{
    list-style: none;
    background: url('rombo.png') left bottom no-repeat;
    padding-left: 15px;
}
```

15.5.3 Posición de la viñeta o numeración

Con la propiedad `list-style-position` podemos indicar si la viñeta o numeración debería aparecer fuera o dentro de cada elemento. Su sintaxis es

```
ul | ol {     list-style-type: inside | outside }
```

Por defecto es `outside`, lo cual indica que la viñeta o numeración quedan fuera del texto del elemento de lista. Esto provoca que si el texto del elemento ocupa más de una línea en la pantalla del navegador, entonces la viñeta o numeración quedan fuera de éste.

Sin embargo, si establecemos esta propiedad al valor `inside`, entonces la viñeta o numeración aparece dentro del texto del elemento y si ocupa más de una línea entonces a partir de la segunda aparecerán por debajo de la viñeta o numeración.

A continuación presentamos dos ejemplos de como quedaría una misma lista empleando ambos valores en la propiedad `list-style-type`.

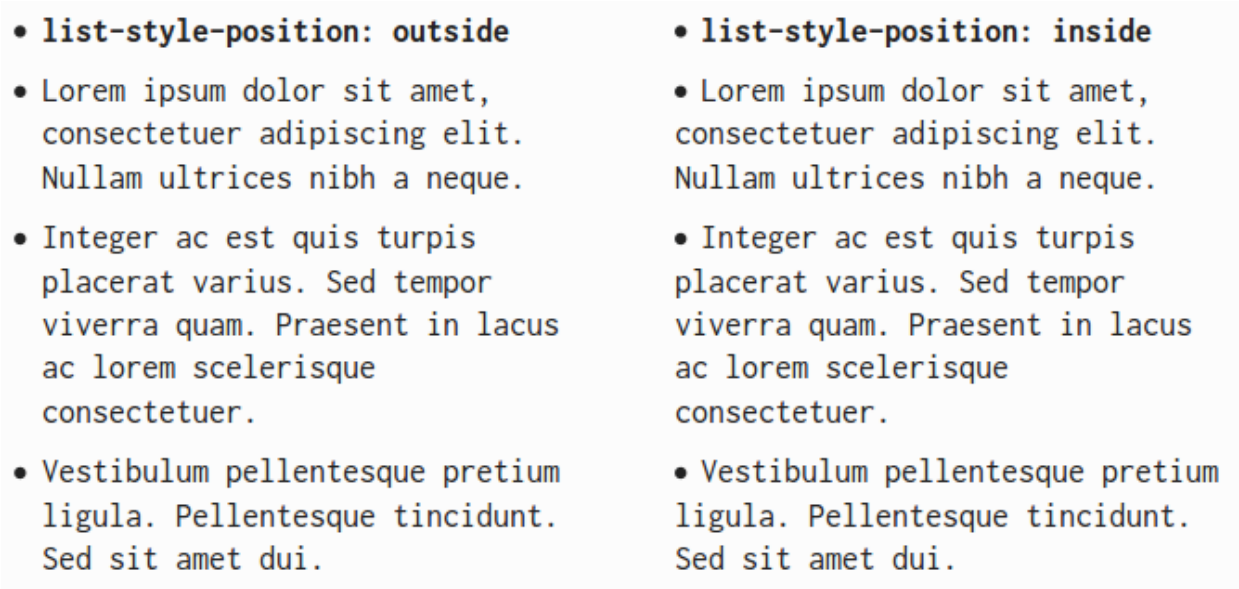


Figura 62.- Posición de la viñeta

15.5.4 Listas multinivel

En las listas multinivel también podemos personalizar las viñetas por niveles. Sin embargo, el selector de la regla se complica algo. Debemos establecer una regla por cada nivel y tenemos dos opciones:

1. Definir una clase para los elementos de un nivel de la lista.
2. Utilizar un selector por contexto del elemento.

En la primera opción simplemente definimos el atributo `class` en cada elemento `li` de cada nivel de la lista. Tiene la ventaja de la simplicidad y que estamos definiendo el formato de un nivel concreto en una lista concreta, lo cual viene muy bien para páginas donde hay varias listas y cada una de ellas tiene un formato diferente.

El segundo es algo más complejo y solo es conveniente usarlo cuando la página tiene solamente una lista o todas las listas que pudieran haber en la página tienen el mismo formato. Consiste en utilizar un selector por contexto de la siguiente manera:

La regla para los elementos del primer nivel de la lista tendría como selector el elemento `ul`

```
ul {
    list-style-image: url(url_imagen);
}
```

Para los elementos del segundo nivel de la lista habría que poner la siguiente regla.

```
ul ul {
    list-style-image: url(url_imagen);
}
```

Como se puede observar el selector `ul ul` indica que el formato se aplicará a un elemento `ul` que se encuentra como descendiente de otro elemento `ul`.

Esta sería la regla para el tercer nivel de la lista

```
ul ul ul{
    list-style-image: url(url_imagen);
}
```

Ahora, la regla anterior se aplicará a todos los elementos `ul` que están dentro de otro `ul` y este a su vez dentro de ontro `ul`. Asi sucesivamente, podemos ir añadiendo `ul` al selector para obtener el formato de los siguientes niveles.

Por ejemplo, supongamos el siguiente formato para una lista

```
ul {
    list-style-image: url('nivel1.png');
}
ul  ul {
    list-style-image: url('nivel2.png');
}
ul  ul  ul {
    list-style-image: url('nivel3.png');
}
```

El resultado sería el siguiente

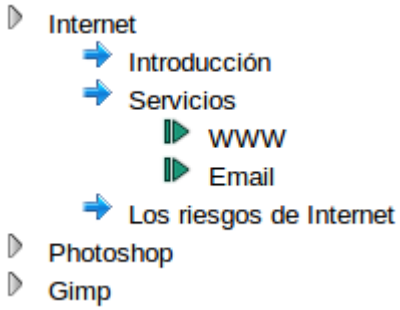


Figura 63.- Lista multinivel personalizada

15.5.5 Contadores

Para las listas multinivel también podemos crear un esquema numerado. Un esquema

numerado sería una lista numerada en la que los elementos se numerán con la numeración del elemento del que subyace, como por ejemplo el índice de este documento.

Para crear un esquema numerado tenemos que utilizar la función `counter()` junto con las propiedades `counter-reset` y `counter-increment`. Además, tendremos que asignar estas propiedades a la pseudoclase `:before` asignando valores a la propiedad `content`.

Veamos por separado cada una de estos conceptos

La propiedad `counter-reset` inicializa el valor de un contador. El nombre del contador es a elección del desarrollador. Por ejemplo, para inicializar un contador llamado `nivel1` tendríamos que escribir la siguiente regla.

```
body {
    counter-reset: nivel1
}
```

La regla anterior hace que el contador `nivel1` se inicialice con el valor 0 al aplicar formato al cuerpo del documento. También podríamos haber indicado un número al lado del nombre del contador para establecer su valor inicial.

La propiedad `counter-increment` se emplea para incrementar un contador. Por ejemplo, para incrementar el contador inicializado antes podríamos escribir lo siguiente

```
counter-increment: nivel1;
```

Si se omite el número de incremento, el contador aumentará en uno. Si queremos que se incremente en otro valor hay que indicarlo al lado del nombre del contador.

La propiedad `content` se emplea para añadir contenido dinámico a un elemento mediante CSS. El valor de esta propiedad se añade al elemento asignada. En nuestro caso lo que haremos es utilizar esta propiedad para asignarle el valor de los contadores. Para ello, la asignaremos a la pseudoclase `:before`, para generar la numeración de los elementos antes del texto de dichos elementos.

Vamos a ver como haríamos un esquema numerado de tres niveles. Para ello necesitaremos tres contadores que inicializaremos en una regla para el elemento `body`.

```
body {
    counter-reset: nivel1 nivel2 nivel3;
}
```

Ahora tenemos que asignar a los elementos del primer nivel la numeración de ese nivel. Al ser listas numeradas vamos a crear la siguiente regla

```
ol li:before {
  counter-increment: nivel1 1;
  counter-reset: nivel2;
  content: counter(nivel1) ".- ";
```

```
}
```

La regla anterior utiliza un selector algo complejo. Indica que antes del elemento `li`, dentro de un elemento `lo`, hay que incrementar el contador del nivel uno, inicializar el contador del nivel dos e insertar antes del texto del documento la numeración del nivel uno y un punto guión como separador de la numeración y el texto del elemento.

La siguiente regla es para los elementos de la lista de nivel 2.

```
ol ol li:before {
  counter-increment: nivel2;
  counter-reset: nivel3;
  content: counter(nivel1) "." counter(nivel2) ".- ";
}
```

Es muy similar a la anterior, pero ahora el contador que se incrementa es el del nivel dos. Hay que inicializar el contador del nivel 3 e insertar la numeración de este nivel con su separador antes del texto de cada elemento de la lista.

Por último, tenemos la regla para los elementos del nivel 3. Sería la siguiente

```
ol ol ol li:before {
  counter-increment: nivel3;
  content: counter(nivel1) "." counter(nivel2) "."
counter(nivel3) ".- ";
}
```

Creo que es lo suficientemente autoexplicativa. Por último y para evitar la numeración habitual de los elementos lo, hay que anularla ya que estamos utilizando la propiedad content en cada elemento li para asignarles dinámicamente esta numeración. Así que tendremos que añadir la siguiente regla.

```
ol {
    list-style-type: none;
}
```

Si insertaramos una lista como la siguiente

```
<h2>Contenidos del curso</h2>
<ol>
    <li>Introducción a la informática
    <li>LibreOffice
    <ol>
        <li>Writer
        <li>Calc
        <li>Impress
    </ol>
    <li>Internet
    <ol>
        <li>Servicio WWW
        <ol>
```

```
            <li>Navegadores web
            <li>Búsquedas en Internet
            <li>Búsquedas en Internet
        </ol>
        <li>Correo electrónico
        <li>Redes sociales
    </ol>
    <li>Amenazas en Internet
</ol>
```

Tendríamos el siguiente resultado

16 Formularios

Un formulario es un documento para introducir unos datos que posteriormente son registrados o procesados. Un formulario web permite al usuario introducir datos los cuales son enviados a un servidor web que se encargará de procesarlos.

En la web hay multitud de sitios web donde el usuario tiene que introducir información: una página de registro en una red social, un mensaje de correo electrónico, una compra de productos en una venta online, etc.

En un formulario web los datos se introducen en uns elementos especiales llamados controles. Existen controles de diferente tipo para adaptarse al tipo de información que hay que introducir en ellos. Por ejemplo, un cuadro de texto servirá para introducir un fragmento de texto tecleado por el usuario, una casilla de verificación se emplea para introducir un dato que solo puede tener dos posibles valores, una lista desplegable consiste en un conjunto de datos de los cuales el usuario tiene que elegir uno.

Los formularios web han sufrido un avance muy importante en la nueva versión de HTML. No solo se han introducido nuevos controles sino también hay nuevos atributos para flexibilizar el uso del formulario por parte del usuario y para facilitar la tarea del desarrollador web. En este capítulo vamos a ver todas estas nuevas características.

16.1 Tipos de controles

Los usuarios interaccionan con los formularios a través de los llamados controles. El *nombre de control* de un control viene dado por su atributo `name`. Cada control tiene tanto un valor inicial como un valor actual, que son ambos cadenas de caracteres. En general, el *valor inicial* de un control puede especificarse con el atributo `value` del elemento de control.

El *valor actual* del control se hace en primer lugar igual al valor inicial. A partir de ese momento, el valor actual del control puede ser modificado a través de la interacción con el usuario y mediante scripts.

El *valor inicial* de un control no cambia. Así, cuando se reinicializa el formulario, el valor actual de cada control se reinicializa a su valor inicial. Si el control no tiene un valor inicial, el efecto de una reinicialización del formulario sobre ese control es indefinido.

Cuando se envía un formulario para su procesamiento, para algunos controles se empareja su nombre con su valor actual, y estas parejas se envían con el formulario. Aquellos controles cuyo par nombre/valor se envían se llaman controles con éxito.

HTML5 define los siguientes tipos de controles:

- ✔ Cuadro de texto.- Disponemos de cuadro de texto de una línea para introducir información textual. Los cuadro de texto pueden ser de los siguientes tipos:
  - ✗ Cuadro de texto normal.
  - ✗ Cuadro de texto para búsqueda
  - ✗ Cuadro de texto para un número de teléfono
  - ✗ Cuadro de texto url
  - ✗ Cuadro de texto email
  - ✗ Cuadro de texto para una contraseña
  - ✗ Cuadro de texto para un número entero
- ✔ Fecha y hora.- Para introducir una fecha mediante un calendario y una hora.
- ✔ Rango de números.- Para introducir un número en coma flotante acotado en un rango.
- ✔ Paleta de colores.- Para seleccionar un color en una paleta de colores.
- ✔ Casilla de verificación.- Para introducir un dato con solo dos posibles valores.
- ✔ Grupo de botones de radio.- Para elegir una opción de entre un conjunto de opciones excluyentes.
- ✔ Botón. Podemos utilizar tres tipos de botones:
  - ✗ Botones de envío de formulario para enviar los datos del formulario.
  - ✗ Botontes de reinicio del formulario para devolver a todos los controles del formulario a su estado inicial.
  - ✗ Botones pulsadores, los cuales solamente realizan una tarea asociada mediante un script.
- ✔ Fichero.- Para subida de ficheros al servidor.
- ✔ Área de texto.- Para introducir un texto largo.
- ✔ Lista desplegable.- Para seleccionar una o varias opciones de entre un conjunto.
- ✔ Lista de datos.- Lista para rellenar cuadros de texto de valor múltiple.
- ✔ Campos ocultos.- Son controles que contienen datos introducidos por el desarrollador web, no por el usuario, el cual no puede verlos.

Los elementos utilizados para crear controles aparecen normalmente dentro de un elemento `form`, pero también pueden aparecer fuera de la declaración de un elemento `form` cuando se utilizan para construir interfaces de usuario. Obsérvese que los controles que estén fuera de un formulario no pueden ser enviados.

16.2 Estructura de un formulario web

Un formulario se representa por el elemento `form`. Entre la etiqueta de apertura y cierre de este elemento hay que introducir todos los controles que forman el formulario. El formulario se envía a una URL que representa una aplicación web encargada de procesar estos datos. Uno de los controles del formulario, generalmente el último, se emplea para enviar los datos al servidor web cuando el usuario ha finalizado de introducir los datos. La sintaxis del elemento `form` es la siguiente

```
<form method="metodo" action="URL>" enctype="tipo_contenido"
accept-charset="lista_codificación_de_caracteres"
accept="lista_de_tipos_de_contenido" id="nombre"
target="destino">

    <fieldset>
        <legend>Título del formulario </legend>
        …
        controles del formulario
        …
    </fieldset>
</form>
```

El elemento `form` contiene todo el formulario. Este incluye uno o varios elementos `fieldset` que se emplean para agrupar los controles de formulario. Un elemento `legend` puede utilizarse para dar un título al formulario. El elemento `form` tiene varios atributos, los más habituales son los siguientes:

Atributo	Descripción
`action=URL`	Este atributo especifica la URL de la aplicación web que recibe los datos del formulario y los procesa.
`method= get | post`	Este atributo especifica qué método HTTP se usará para enviar el conjunto de datos del formulario. Los valores posibles (que no distinguen entre mayúsculas y minúsculas) son "get" (valor por defecto) y "post". ✔ `Get`. Con este método, el conjunto de datos del formulario se agrega al URL especificado por el atributo action (con un signo de interrogación ("?") como separador) y este nuevo URL se envía al programa que procesará los datos del formulario. ✔ `post`. Con este método, el conjunto de datos del formulario se incluye en el cuerpo del formulario y se envía a la aplicación web

`enctype = tipo_de_contenido`	Este atributo especifica el tipo de contenido usado para enviar el formulario al servidor (cuando el valor del atributo method sea `post`). El valor por defecto de este atributo es `application/x-www-form-urlencoded`. El valor `multipart/form-data` debería usarse en combinación con el elemento `<input type="file">`.
`accept-charset = lista_codificación_caracteres`	Este atributo especifica la lista de codificaciones de caracteres para los datos introducidos que son aceptadas por el servidor que procesa este formulario. El valor es una lista de valores de codificaciones de caracteres separadas por espacios y/o comas. El cliente debe interpretar esta lista como una lista o-exclusiva, es decir, el servidor es capaz de aceptar cualquier codificación de caracteres individual por entidad recibida. El valor por defecto de este atributo es la cadena reservada "UNKNOWN" ("desconocido"). Los navegadores pueden interpretar este valor como la codificación de caracteres que fue usada para transmitir el documento que contiene este elemento `form`.
`accept = lista_de_tipos_de_contenido`	Este atributo especifica una lista de tipos de contenido separados por comas que un servidor procesador de formularios manejará correctamente. Los navegadores pueden utilizar esta información para filtrar ficheros no conformes cuando pidan al usuario seleccionar ficheros para enviar al servidor mediante un el elemento `input type="file"`.
`id = nombre`	Este atributo da un identificador al elemento de modo que se pueda hacer referencia a él desde hojas de estilo o scripts.

El elemento `form` actúa como un contenedor de controles. Este debe especificar:

- ✔ La disposición del formulario, dada por los contenidos del elemento.
- ✔ El programa que manejará el formulario completado y enviado (el atributo `action`). El programa receptor debe ser capaz de interpretar las parejas nombre/valor para poder hacer uso de ellas.
- ✔ El método por el cual se enviarán los datos del usuario al servidor (el atributo `method`).
- ✔ Una codificación de caracteres que debe ser aceptada por el servidor para poder manejar este formulario (el atributo `accept-charset`).

Un formulario no solo puede contener controles, sino también texto y otros elementos (párrafos, listas, etc.). Por ejemplo, para nuestro ejemplo podríamos definir un elemento `form` de la siguiente manera.

```
<form action="http://www.pizzaexpress.com/pedidos.php"
```

```
method="post" accept-charset="utf-8">
…
</form>
```

16.2.1 Elemento fieldset

El elemento `fieldset` (grupo de campos) permite a los desarrolladores web agrupar temáticamente controles y rótulos relacionados. Gracias al agrupamiento de controles es más fácil para los usuarios entender su propósito y al mismo tiempo se facilita la navegación. El uso correcto de este elemento hace los documentos más accesibles. Su uso no es obligatorio, aunque si recomendable

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

El elemento `legend` admite el atributo `align` que especifica la posición de la leyenda con respecto al grupo de campos. Valores posibles:

- ✔ `top`: La leyenda está en la parte superior del grupo de campos. Este es el valor por defecto.
- ✔ `bottom`: La leyenda está en la parte inferior del grupo de campos.
- ✔ `left`: La leyenda está a la izquierda del grupo de campos.
- ✔ `right`: La leyenda está a la derecha del grupo de campos.

Siguiendo con el ejemplo, vamos a introducir estos elementos para que el formulario vaya cogiendo forma.

```
<form action="http://www.pizzaexpress.com/pedidos.php"
method="post" accept-charset="utf-8">
    <fieldset>
        <legend>Pide tu pizza express</legend>
        …
    </fieldset>
</form>
```

16.3 Controles de formulario

A continuación vamos a ver los diferentes elementos necesarios para crear los controles dentro de un formulario. El elemento principal es `input` ya que mediante este elemento vamos a poder introducir controles de muchos tipos diferentes. Aunque también tendremos elementos `select`, para una lista desplegable, `textarea` para un área de texto y `button` para un botón.

Sin embargo, los controles de un formulario comparten una serie de atributos para definir su comportamiento dentro del formulario. Los principales son:

Atributo	Descripción

`name="variable" id="variable"`	Estos atributos son fundamentales y obligatorio su uso en al menos uno de ellos. Contiene el nombre que recibe la variable en la que se almacena la entrada del formulario o datos que el visitante teclee o marque. Debería usarse `id`, pero `name` se incluye por compatibilidad con versiones anteriores. Para asegurarse utilizar ambos.
`disabled`	Este atributo vacío deshabilita el control para la entrada de datos por parte del usuario. Cuando está establecido, el atributo `disabled` tiene los siguientes efectos sobre un elemento: ✔ No se puede dirigir el foco hacia controles deshabilitados. ✔ En el orden de tabulación, se salta por encima de los controles deshabilitados. ✔ Los controles deshabilitados no pueden tener éxito. El modo en que se representan los elementos deshabilitados depende del navegador . Por ejemplo, algunos navegadores dibujan en gris los objetos de menú deshabilitados, los rótulos de los botones, etc.
`readonly`	Especifica si el control puede ser modificado por el usuario. Cuando está establecido, el atributo `readonly` tiene los siguientes efectos sobre un elemento: ✔ El foco puede dirigirse hacia elementos de sólo lectura, pero éstos no pueden ser modificados por el usuario. ✔ Los elementos de sólo lectura están incluidos en la navegación con tabulador. ✔ Los elementos de sólo lectura pueden tener éxito. El modo en que se representan los elementos de sólo lectura depende del agente de usuario. La única manera de modificar dinámicamente el valor del atributo readonly es mediante un script.
`required`	Atributo vacío que especifica que el usuario tiene que introducir un dato obligatoriamente. De lo contrario no se podrá enviar los datos del fomulario.
`autofocus`	El control que tenga establecido este atributo vacío será el que obtenga el foco automáticamente cuando la página que contiene el formulario se presenta en la pantalla del navegador.
`accesskey="caracter"`	Asigna una tecla de acceso a un elemento. Una tecla de acceso es un carácter único del conjunto de caracteres del documento. Se debería considerar la forma que tienen los usuarios potenciales de interaccionar a la hora de especificar la tecla de acceso. Al pulsar la tecla de acceso asignada a un elemento, el foco se dirige hacia el elemento. La acción que tiene lugar cuando el foco se dirige hacia un elemento depende del elemento. Por

	ejemplo, cuando el usuario activa un campo de texto, éste permite la entrada de texto, etc. La invocación de teclas de acceso depende del sistema subyacente. Por ejemplo, en máquinas que ejecuten MS Windows, normalmente hay que pulsar la tecla `alt` además de la tecla de acceso. En sistemas Apple, normalmente hay que pulsar la tecla `cmd` además de la tecla de acceso. La representación de teclas de acceso depende del navegador. Es recomendable incluir la tecla de acceso en el texto del rótulo o dondequiera que se aplique la tecla de acceso. Los navegadores deberían representar las teclas de acceso de tal modo que se enfatice su papel y se distinga de otros caracteres (p.ej., subrayándola).
`tabindex="nº"`	Este atributo especifica la posición del elemento actual dentro del orden de tabulación del documento actual. Este valor debe ser un número entre 0 y 32767. Los navegadores no deberían tener en cuenta los ceros a la izquierda. El orden de tabulación define el orden en que el foco se dirige hacia los elementos cuando se navega por medio del teclado, generalmente mediante la tecla `Tab`. El orden de tabulación puede incluir elementos anidados en otros elementos. Los navegadores deberían navegar por los elementos a los que puede dirigirse el foco de acuerdo con las siguientes reglas: 1. Navegar en primer lugar por aquellos elementos que soporten el atributo `tabindex` y tengan asignado para éste un valor positivo. La navegación se produce desde el elemento con menor valor de `tabindex` hasta el elemento con el valor más alto. Los valores no necesitan ser secuenciales ni deben comenzar por un valor en particular. Si hay elementos que tengan valores idénticos de `tabindex` debería navegarse por ellos según el orden en que aparezcan en el flujo del código HTML. 2. A continuación se navega por aquellos elementos que no soporten el atributo `tabindex` o por los que soportándolo tengan asignado para él un valor "0". Se navega por estos elementos según el orden en que aparezcan en el flujo deL código HTML. 3. Los elementos que estén deshabilitados no participan en el orden de tabulación.

16.3.1 Etiquetas de control

El elemento `label` representa un título en un control de formulario. Este título puede asociarse a su control mediante el atributo `for` o definiendo el elemento del control dentro del propio elemento `label`. Por ejemplo

```
<label for="nombre">Nombre</label> <input type="text"
name="nombre" size="30"/>
```

Vemos que el atributo for del elemento label contiene el id del cuadro de texto. También podríamos haberlo definido así

```
<label>Nombre <input type="text" name="nombre"
size="30"/></label>
```

16.3.2 Elemento input

El elemento principal para un control de formulario es `input` ya que mediante este elemento vamos a poder introducir controles de muchos tipos diferentes. Evidentemente, para distinguir un tipo de control de otro hay que introducir algo en el elemento que haga esta diferencia. En este caso será el valor del atributo `type`. El valor que tome este atributo nos dará el tipo de control. El elemento `input` se emplea para representar la mayoría de los controles de usuario. Los principales son:

- ✔ Cuadro de texto.
- ✔ Texto de búsqueda.
- ✔ Teléfono.
- ✔ URL.
- ✔ Email.
- ✔ Cuadro de contraseña.
- ✔ Fecha y hora.
- ✔ Número.
- ✔ Rango.
- ✔ Selección de color.
- ✔ Casilla de verificación.
- ✔ Grupo de botones de radio.
- ✔ Botón.
- ✔ Fichero.
- ✔ Imagen.
- ✔ Botón para envío de formulario.
- ✔ Botón para reinicio del formulario.
- ✔ Campo oculto.

Además, un elemento `input` dispone de una serie de atributos, muchos de los cuales son comunes a la mayoría de los tipos de controles. Estos son:

Atributo	Descripción
`id = variable` `name = variable`	Estos atributos son fundamentales y obligatorio su uso en al menos uno de ellos. Contiene el nombre que recibe la variable en la que se almacena la entrada del formulario o datos que el visitante teclee o marque. Debería usarse `id`, pero `name` se incluye por compatibilidad con versiones anteriores. Para asegurarse, es recomendable utilizar ambos.
`value="valor numérico/texto"`	Es un atributo que nos sirve para dar un valor inicial del campo que deseemos. Esto es muy útil si por ejemplo en el caso de controles que tienen un valor por defecto que raramente cambia. Así, si el valor esta preestablecido, en la mayoría de las ocasiones el usuario no tendrá que teclearlo. En el caso de los botones de envío (`<input type="submit">`) y de reinicio del formulario (`<input type="reset">`) sirve para cambiar los mensajes que por defecto muestran estos botones. Modificando este valor para estos botones, podremos poner el texto que deseemos. Hay otros controles en los que el desarrollador deberá darles un valor inicial ya que el usuario lo seleccionará de una lista, un grupo de botones de radio o una casilla de verificación.
`size="valor numérico"`	Define la longitud de la ventana de texto en caracteres.
`maxlength="valor numérico"`	Es la longitud máxima del numero de caracteres que se permite al usuario introducir en el control. El valor de `maxlength` no puede ser menor que `size`, de forma que visualmente no tecleemos hasta el final de la línea del formulario, dando un aspecto más pulido, del formulario.
`checked`	Cuando el atributo `type` tiene el valor `radio` o `checkbox`, este atributo vacío especifica que el botón está marcado (on). Los navegadores no deben tener en cuenta este atributo para otros tipos de control.
`pattern="expresion_regular"`	Especifica una expresión regular a la que se tiene que ajustar el valor introducido en el control. Por ejemplo, si tenemos un cuadro de texto para recoger el código postal del usuario podemos escribir una expresión regular que solamente admita valores numericos de cinco cifras.
`multiple`	Este atributo vacío se establece para indicar si el control admite más de un valor. Por ejemplo, en un cuadro de texto donde introducimos varias direcciones de correo electrónico.
`min="nº"`	En el caso de controles para datos numéricos indican el valor

max=”nº”	mínimo y valor máximo que se admite para un control.
placeholder =”texto”	Este atributo contiene un texto de ayuda que aparece en el control textual para indicar al usuario una pista sobre el tipo de información que debe introducir. Por ejemplo, para un control de tipo fecha puede poner placeholder=”dd/mm/aaaa” para indicar al usuario el formato de fecha que debe introducir y este valor aparecerá en un color gris difuminado. En cuanto el control obtenga el foco el texto de ayuda desaparecerá.

Vamos a ver con algún detalle más los diferentes controles de formulario.

16.3.3 Cuadros de texto

Un elemento input con type=”text” es un cuadro de texto simple. Indica que se trata de una entrada de datos formada por una línea de texto. Por ejemplo

```
<label>
Nombre <input type="text" name="nombre" size="20" maxlength=
"30" tabindex="1" required placeholder="Introduce tu nombre"/>
</label>
<label>
Apellidos <input type="text" name="apellidos" size="40"
maxlength= "60" tabindex="2" required placeholder="Introduce tus
apellidos"/>
</label>
```

Vemos que los cuadro de texto están introducidos dentro de elementos label para asignarles sus etiquetas. Nótese también el uso de los atributos comúnes tanto en los controles de formulario como en el elemento input. El aspecto del formulario con estos dos controles de texto sería el siguiente.

Figura 64.- Controles de texto

El cuadro de texto nombre tiene un tamaño visual de 20 caracteres y admite como máximo 30. También, es obligatorio introducir un dato.

16.3.4 Texto de búsqueda

Este control es muy similar al anterior. Consiste en un cuadro de texto para introducir un criterio de búsqueda. Por ejemplo

```
<label>
Buscar <input type="search" name="buscar" size="60"
placeholder="Introduce un texto a buscar"/>
</label>
```

16.3.5 Teléfono

El control con `type="tel"` representa un cuadro de texto para introducir un número de teléfono. Esto no quiere decir que el control solamente admite números ya que los formatos de número de teléfono son muy variados. Así que este cuadro de texto admite cualquier texto. Si queremos que su valor se ajuste a un formato de número telefónico concreto tendremos que utiliar el atributo `pattern`. Por ejemplo

```
<label>
Teléfono <input type="tel" name="telefono" size="9" maxlength=
"9" tabindex="3" required placeholder="#########" pattern="[0-9]
{9}"/>
</label>
```

Como podemos observar hemos puesto un atributo `pattern` para que solamente admita un número de nueve cifras.

16.3.6 URL

Este control textual se emplea para introducir URL's absolutas. El valor de su atributo `type` es `url`. Veamos el siguiente ejemplo.

```
<label>
Dirección web <input type="url" name="direccion_web" size="50"
maxlength= "256" tabindex="4"/>
</label>
```

Su aspecto visual es similar a un cuadro de texto, pero solamente admite texto con formato de URL absoluta.

16.3.7 Email

Parecido al anterior, un control con `type="email"` se emplea para introducir una dirección de correo electrónico. Es por tanto un control textual, pero el texto introducido debe ajustarse al formato de las direcciones de correo electrónico. Por ejemplo

```
<label>
Email <input type="email" name="email" size="50" maxlength=
"128" tabindex="5"/>
</label>
```

16.3.8 Cuadro de contraseña

Este es un cuadro de texto para introducir una clave. Cuando el usuario teclea texto dentro los caracteres no se visualizan, son sustituidos por asteriscos. El valor del atributo `type` es `password`. Veamos el siguiente ejemplo.

```
<label>
Clave <input type="password" name="clave" size="10" maxlength=
"10" tabindex="6" required/>
</label>
```

16.3.9 Fecha y hora

Para introducir campos de tipo fecha disponemos de `type="date"` y para campos hora `type="time"`. En el momento de escribir este documento Chrome y Safari son los únicos navegadores que reconocen estos controles. Los navegadores que no lo reconozcan presentarán un cuadro de texto simple. Veamos un ejemplo

```
<label>
Fecha de nacimiento <input type="date" name="fecha_nac"
tabindex="7" required/>
</label>
```

Para un campo tipo fecha el resultado podría ser el siguiente:

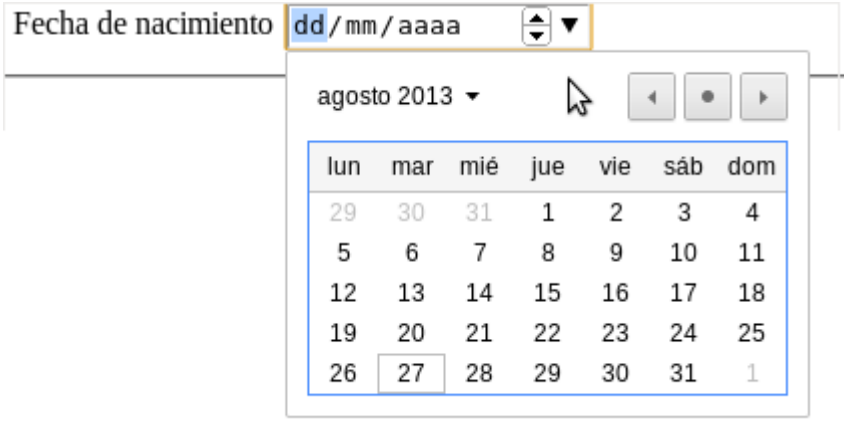


Figura 65.- Campo tipo fecha

Como se puede apreciar, permite desplegar un calendario para seleccionar un día concreto.

Para el campo tipo hora tenemos el siguiente ejemplo

```
<label>
Hora de salida <input type="time" name="hora_salida"/>
</label>
```

Su resultado sería

Figura 66.- Campo tipo hora

Estos tipos de controles evitan que el usuario introduzca datos erróneos en datos con un formato muy específico.

16.3.10 Número

Es un cuadro de texto para introducir un número. Un control con `type="number"` solamente admite números. Veamos el siguiente ejemplo

```
<label>
Edad <input type="number" name="edad" size="2" min="18" max="65"
```

```
tabindex="6" required/>
</label>
```

Como podemos apreciar, hemos incluido atributos `min` y `max` para indicar el rango admisible de números. Su aspecto dependerá del navegador utilizado. En Google Chrome v28 aparecería así.

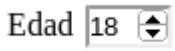


Figura 67.-
Control number

Disponemos de dos flechas para aumentar o reducir el número.

16.3.11 Rango

Un control `type="range"` se emplea para introducir un valor dentro de un rango de valores. Puede parecer igual a un campo tipo `number` con los atributos `min` y `max` establecidos, pero hay una diferencia. En un campo tipo `range` podemos introducir valores en punto flotante, mientras que en campos tipo `number` solamente admite números enteros.

Los navegadores deberían presentar un control de este tipo mediante una barra deslizante, por lo que el usuario puede establecer el valor arrastrando la barra a derecha o izquierda. La variación del valor producido por cada desplazamiento es indicado por el atributo `step`. Por ejemplo, si necesitamos un campo para introducir un porcentaje entre 0 y 100 con dos decimales podríamos poner un control de esta manera.

```
<label>
Porcentaje <input type="range" name="porcentaje" min="0"
max="100" step="0.01" value="0"/>
</label>
```

16.3.12 Color

Un control `type="color"` se emplea para seleccionar un color de una paleta de colores. Veamos el siguiente ejemplo.

```
<label>
Color <input type="color" name="color" value="#FF0000"/>
</label>
```

El dato que el navegador envía al servidor es un valor hexadecimal con el color elegido. Con el atributo `value` podemos establecer un valor por defecto. El resultado es el siguiente en un navegador Chromium.

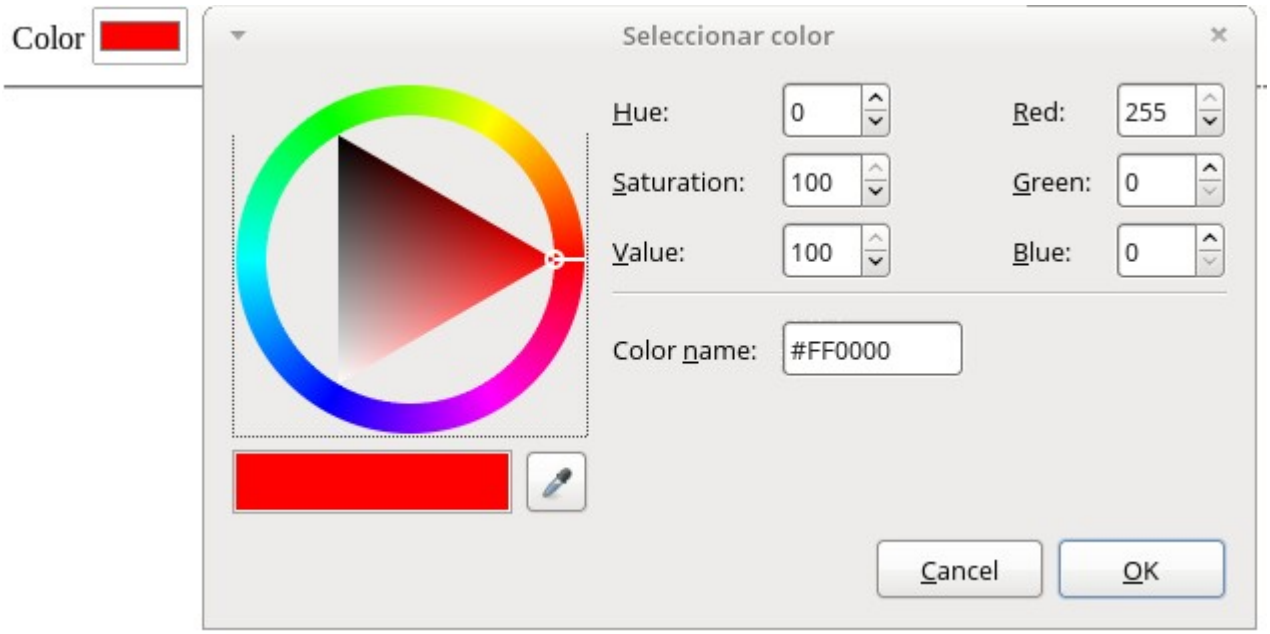


Figura 68.- Control color

16.3.13 Casilla de verificación

Una casilla de verificación es un control para recoger un dato que solamente puede tener dos posibles valores: seleccionado o no seleccionado. El atributo `value` recoge el valor cuando se selecciona, sino es así se devuelve una cadena vacía. Si se incluye el atributo `checked` se indica que inicialmente la casilla aparecerá seleccionada. Por ejemplo

```
<label><input type="checkbox" name="familia_num" value="Si"
>Familia Numerosa</label>
```

El aspecto de la casilla de verificación podría ser el siguiente.

Figura 69.- Casilla de verificación

16.3.14 Grupo de botones de radio

Un grupo de botones de radio es un conjunto de opciones de las cuales hay que elegir solamente una. Cuando el usuario selecciona una de las opciones, las otras quedan sin selección. Para poder tener un grupo de botones de radiohay que incluir un elemento `input` con el atributo `type` establecido al valor `radio` por cada opción del grupo. Todas estos elementos deben tener el mismo valor del atributo `name` e `id`, así, el navegador sabe que para todos los elementos del grupo de botones de radio con la misma variable deberá controlar que solamente uno esté seleccionado. El navegador enviará al servidor el valor del atributo `value` de la opción seleccionada. Por tanto, los valores de este atributo deben ser diferentes.

Si el usuario no selecciona ningún valor, se devuelve una cadena vacía, y se puede usar el atributo `checked` para seleccionar por defecto una de las opciones. Veamos un ejemplo.

```
<fieldset>
<legend>Forma de pago </legend>
<input type="radio" name="forma_pago" value="visa"/> Visa
<input type="radio" name="forma_pago" value="ae"/> American
Express
<input type="radio" name="forma_pago" value="mc"/> MasterCard
<input type="radio" name="forma_pago" value="pp"/> Paypal
</fieldset>
```

En este ejemplo el usuario debe elegir una única forma de pago. El resultado sería el siguiente

Figura 70.- Grupo de botones de radio

16.3.15 Fichero

Hay formularios que se emplean para subir archivos a un servidor. En este caso debemos hacer dos cosas. La primera es codificar el formulario adecuadamente. Aquí hay que establecer el atributo `enctype` del elemento `form` al valor `multipart/form-data`. Posteriormente debemos introducir un elemento `input` con el valor del atributo `type` a `file`, el cual genera un botón *Examinar...* que permitirá buscar un fichero en nuestro ordenador. Dependiendo del navegador puede aparecer un cuadro de texto para escribir directamente el nombre del archivo. Veamos un ejemplo

```
<form method="post" name="subir_archivo"
enctype="multipart/form-data"/>
    Adjuntar fotocopia DNI: <input type="file" name="archivo"/>
    …
</form>
```

Su aspecto puede ser el siguiente

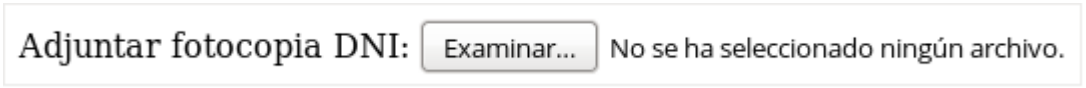


Figura 71.- Campo fichero

16.3.16 Imagen

Este control no es muy habitual usarlo. Consiste en una imagen que al hacer clic sobre ella se envía el formulario al servidor. Hace que el navegador presente una imagen que es sensible al ratón. Lo que el formulario envía al servidor es un registro cuyos campos son el nombre definido para la imagen seguido de los parámetros `.x=n .y=n` donde `n` son los números de las coordenadas `x y` del punto en el que estaba el ratón en el momento del envío. Para indicar el archivo donde está la imagen se utiliza el atributo `src` . Este sería un ejemplo de respuesta:

```
imagen.x=99&imagen.y=15
```

El ejemplo que generaría esta respuesta es

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

Generalmente, para enviar los formularios se utilizan botones de envío.

16.3.17 Área de texto

El elemento `textarea` crea un control de entrada de texto. Los navegadores deberían usar los contenidos de este elemento como valor inicial del control y representar este texto inicialmente. Para indicar el tamaño del área de texto se emplean los siguientes atributos:

`rows = número`

Este atributo especifica el número de líneas de texto visibles. Los usuarios podrán introducir más líneas de texto, por lo que se dispondrán de barras de desplazamiento en el área de texto cuando los contenidos se extiendan más allá del área visible.

`cols = número`

Este atributo especifica la anchura visible en caracteres de anchura media.

El siguiente ejemplo crea un control `textarea` de 5 filas por 80 columnas que contiene inicialmente dos líneas de texto.

```
<label>
Observaciones<br>
<textarea name="observaciones" rows="5" cols="80">
Introduzca aquí cualquier comentario a tener en cuenta.
Si dispone de alguna minusvalía física, indíquela aquí también.
</textarea>
</label>
```

El resultado sería el siguiente

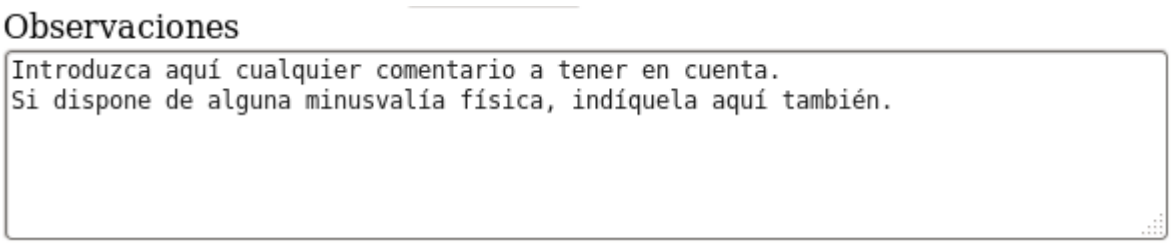


Figura 72.- Área de texto

16.3.18 Lista desplegable

El elemento `select` crea una lista desplegable. Cada opción ofrecida por la lista se representa por un elemento `option`. Un elemento `select` debe contener al menos un elemento `option`.

El elemento `optgroup` permite agrupar opciones lógicamente. Esto es particularmente útil cuando el usuario debe elegir de entre una larga lista de opciones; es más fácil apreciar y recordar grupos de opciones relacionadas que una larga lista de opciones sueltas. Todos

los elementos `optgroup` deben especificarse directamente dentro de un elemento `select`, es decir, no pueden anidarse unos grupos dentro de otros.

Para especificar el tamaño de la lista disponemos del atributo `size`. Si un elemento `select` se presenta como una lista fija con desplazamiento, este atributo especifica el número de filas de la lista que deberían ser visibles al mismo tiempo. Si el valor de este atributo es 1, entonces la lista solamente presenta el valor seleccionado y puede desplegarse para visualizar el resto y seleccionar alguno.

Si el atributo `multiple` está activado, permite selecciones múltiples. Si no está activado, el elemento `select` sólo permite seleccionar un solo elemento de la lista.

16.3.18.1 Opciones de la lista

Cada opción de la lista se representa mediante un elemento `option`. Cuando se represente una opción de un menú, los navegadores deberían usar como opción el valor del atributo `label`. Si este atributo no está especificado, los navegadores deberían usar los contenidos del elemento `option`. Los atributos del elemento `option` son los siguientes:

`selected`

Si está establecido, este atributo vacío especifica que esta opción está preseleccionada.

`value = "valor"`

Este atributo especifica el valor inicial del control. Si este atributo no está establecido, el valor inicial es igual a los contenidos del elemento `option`.

`label = "texto"`

Este atributo permite especificar un rótulo para la opción más corto que el contenido del elemento `option`. Cuando esté especificado, los agentes de usuario deberían usar como rótulo de la opción el valor de este atributo en lugar del contenido del elemento `option`.

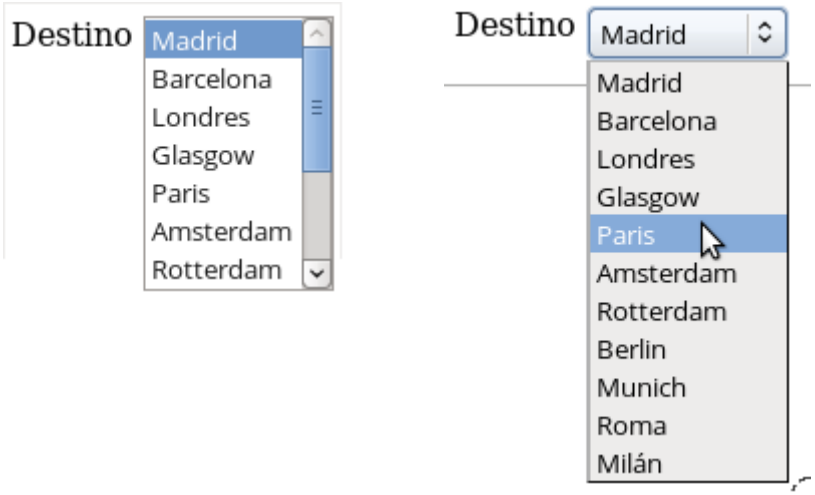
En este ejemplo, creamos una lista que permite al usuario seleccionar un destino para su viaje.

```
<label>Destino
<select size="7" name="destino">
    <option value="MAD">Madrid</option>
    <option value="BCN">Barcelona</option>
    <option value="LHR">Londres</option>
    <option value="GLA">Glasgow</option>
    <option value="CDG">Paris</option>
    <option value="AMS">Amsterdam</option>
    <option value="RTM">Rotterdam</option>
    <option value="SXF">Berlin</option>
    <option value="MUC">Munich</option>
    <option value="FCO">Roma</option>
    <option value="LIN">Milán</option>
```

```
</select>
```

También podemos dejar la lista desplegable usando solamente una tamaño de 1. La definición del elemento `select` sería el siguiente.

```
<select size="1" name="destino">
```



Con las listas desplegables, es obligatorio seleccionar siempre un valor, y si no hay ninguno seleccionado, por defecto el navegadorenviará el primer valor de la lista.

16.3.18.2 Agrupar opciones.

Dentro de una lista podemos utilizar el elemento `optgroup` para agrupar conjuntos de opciones que están lógicamente agrupadas. El atributo `label` del elemento `optgroup` especifica el rótulo de un grupo de opciones.

Si añadimos grupos de opciones se mejora la interacción con el usuario al poder este localizar más rápidamente la opción que busca. El ejemplo anterior quedaría de la siguiente manera.

```
<label>Destino
<select size="12" name="destino">
    <optgroup label="España">
        <option value="MAD">Madrid</option>
        <option value="BCN">Barcelona</option>
    </optgroup>

    <optgroup label="Reino Unido">
        <option value="LHR">Londres</option>
        <option value="GLA">Glasgow</option>
    </optgroup>

    <optgroup label="Francia">
        <option value="CDG">Paris</option>
    </optgroup>
```

```
    <optgroup label="Holanda">
         <option value="AMS">Amsterdam</option>
         <option value="RTM">Rotterdam</option>
    </optgroup>

    <optgroup label="Alemania">
         <option value="SXF">Berlin</option>
         <option value="MUC">Munich</option>
    </option>

    <optgroup label="Italia">
         <option value="FCO">Roma</option>
         <option value="LIN">Milán</option>
    </optgroup>
</select>
```

El resultado podría ser el siguiente

Figura 73.- Grupos de opciones

16.3.19 Envío de formulario

Un elemento input con el atributo type establecido al valor submit es un botón de envío del formulario. El valor del atributo value será el texto que aparezca en el botón. También, podemos asinarle un `id` y en este caso también se enviaría este dato al servidor. Por ejemplo

```
<input type="submit" name="operacion" id="operacion"
value="Buscar vuelos"/>
```

como podemos apreciar, este control no necesita un elemento `label`. El resultado sería el siguiente.

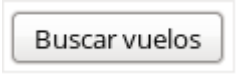


Figura 74.- Botón de envío

16.3.20 Reinicio del formulario

Es un tipo de elemento especial que no recibe datos, sólo ejecuta localmente una acción. Tiene el aspecto de un botón y hace que los campos de entrada y todos los componentes del formulario vuelvan a su estado original con sus valores por defecto. Por tanto carece de atributo `id` y `name`. El atributo opcional `value` contiene la frase que se mostrará en el botón, en lugar de Reiniciar, que es el texto por defecto. Por ejemplo

```
<input type="reset" value="Empezar de nuevo"/>
```

El resultado sería

Figura 75.- Botón reset

16.3.21 Campo oculto

Es simplemente un almacenamiento interno de datos. El visualizador no muestra el contenido, recogido en el atributo `value`, pero sí se lo pasa al programa que recibe los datos del formulario definido en el atributo `action` del elemento `form`. Resulta un componente muy adecuado para guardar información que se mantiene estable en la página o algún dato que el programa necesita y no tiene que suministrarlo el usuario. Por ejemplo

```
<input type="hidden" name="accion" id="accion" value="buscar"/>
```

16.3.22 Elemento button

Los botones creados con el elemento `button` funcionan exactamente igual que los botones creados con el elemento `input`, pero ofrecen posibilidades más ricas de representación: el elemento `button` puede tener contenido.

El siguiente ejemplo crea los botones de envío y de reinicialización con `button` en lugar de con `input`. Los botones contienen imágenes sacadas de elementos <IMG>.

```
<button name="operacion" id="operacion" value="enviar"
type="submit">
Enviar <img src="/iconos/ahivaeso.gif" alt="¡Ahí va eso!"/>
</button>
<button name="reiniciar" type="reset">
Reinicializar <img src="/iconos/ayno.gif" alt="¡Ay, no!"/>
</button>
```

Los atributos de este elemento son los siguientes

Atributo	Descripción
`value = “valor”`	Este atributo asigna al botón su valor inicial.

`type = submit | button | reset`	Este atributo declara el tipo del botón. Valores posibles: ✔ `submit`: Crea un botón de envío. Este es el valor por defecto. ✔ `reset`: Crea un botón de reinicialización. ✔ `button`: Crea un botón pulsador.

16.3.23 Lista de opciones. Elemento datalist

Una lista de datos es un conjunto de opciones que se emplea como sugerencia al usuario a la hora de introducir un dato en un control. Por ejemplo, supongamos que en un cuadro de texto el usuario tiene que introducir su cantante favorito. Pues bien, se le puede mostrar una lista con un conjunto de cantantes y grupos musicales como sugerencia por si alguno de ellos fuera su favorito. Así el usuario lo puede elegir y se asignaría al cuadro de texto. Si ningún elemento de la lista fuera la opción del usuario, entonces tendrá que teclearla.

A la hora de hacer una lista de opciones hay primero que definir un elemento `datalist`, que contendrá elementos `option` por cada opción o sugerencia al usuario. Posteriormente, hay que asignar el elemento `datalist` a un elemento `input` mediante el atributo `list`. Vamos a ver un ejemplo.

```
<label>
    Cantante o grupo favorito:
    <input type="text" name="cantante" id="cantante" size="30"
maxlength="100" list="cantantes"/>
</label>
<datalist id="cantantes">
    <option value="AC/DC">
    <option value="Robert Palmer">
    <option value="Poison">
    <option value="Billy Idol">
    <option value="John Cougar Mellencamp">
</datalist>
```

El resultado sería el siguiente

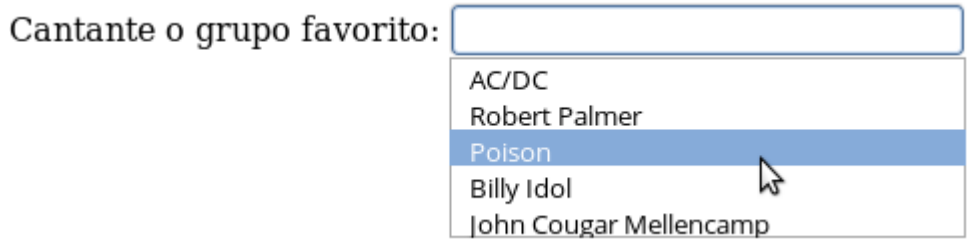


Figura 76.- Lista de opciones

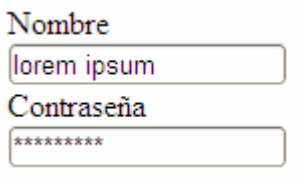
Cuando el usuario hace clic sobre el cuadro de texto se despliega la lista de sugerencias. También si pulsa la tecla flecha abajo. Además, cuando el usuario comienza a teclear le aparecerán las sugerencias que coincidan con el texto tecleado.

17 Formato del formulario

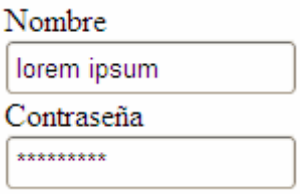
Mediante el empleo de las propiedades de CSS podemos mejorar el aspecto de los formularios. Como la etiqueta `<input>` se emplea para varios tipos de controles, esta sección nos ayudará a introducir los selectores por los atributos del elemento.

17.1 Mejoras en los campos de texto

Por defecto, los campos de texto de los formularios no incluyen ningún espacio de relleno, por lo que el texto introducido por el usuario aparece pegado a los bordes del cuadro de texto. Añadiendo un pequeño relleno a cada elemento `input`, se mejora notablemente el aspecto del formulario. Por ejemplo, si no ponemos ningún relleno, veríamos los campos de texto de un formulario así

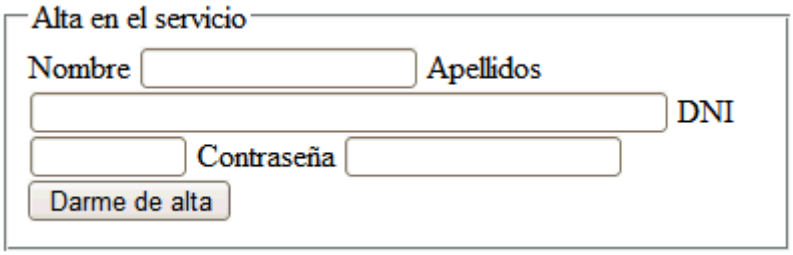


Si añadimos un relleno de 3 píxeles lo veríamos así



17.2 Etiquetas alineadas y formateadas

Los elementos `input` y `label` de los formularios son elementos en línea, por lo que el aspecto que muestran los formularios por defecto, es similar al de la siguiente imagen.



El código HTML del ejemplo anterior es el siguiente:

```
<form>
<fieldset>
  <legend>Alta en el servicio</legend>

  <label for="nombre">Nombre</label>
  <input type="text" id="nombre" />
```

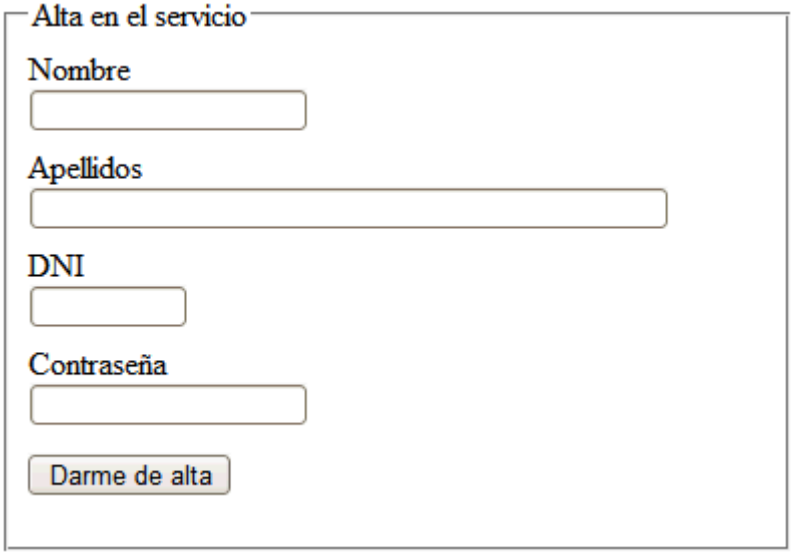
```
  <label for="apellidos">Apellidos</label>
  <input type="text" id="apellidos" size="50" />

  <label for="dni">DNI</label>
  <input type="text" id="dni" size="10" maxlength="9" />

  <label for="contrasena">Contraseña</label>
  <input type="password" id="contrasena" />

  <input class="btn" type="submit" value="Darme de alta" />
</fieldset>
</form>
```

Aprovechando los elementos `label`, se pueden aplicar unos estilos CSS sencillos que permitan mostrar el formulario con el aspecto de la siguiente imagen



En primer lugar, se muestran los elementos `label` como elementos de bloque, para que añadan una separación para cada campo del formulario. Además, se añade un margen superior para no mostrar juntas todas las filas del formulario:

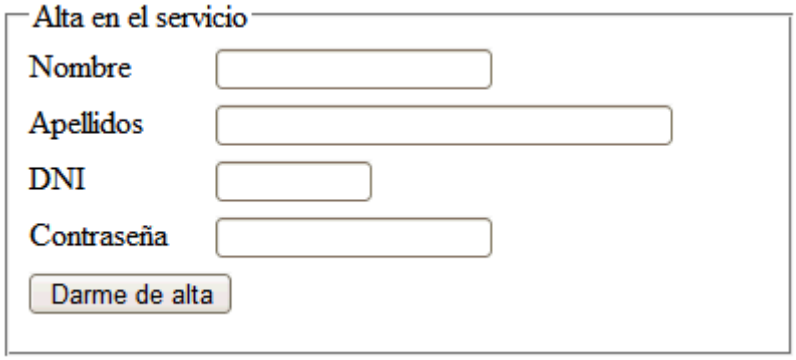
```
label {
  display: block;
  margin: .5em 0 0 0;
}
```

El botón del formulario también se muestra como un elemento de bloque y se le añade un margen para darle el aspecto final deseado. Para ello definimos una clase que asignaríamos al botón

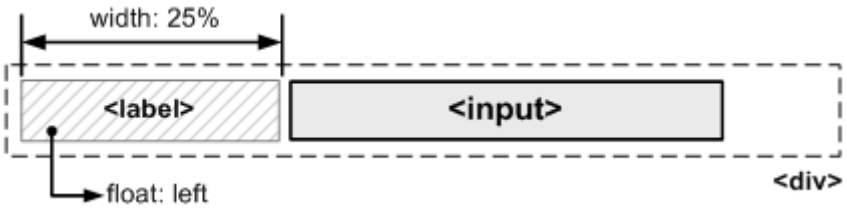
```
.btn {
  display: block;
  margin: 1em 0;
```

```
}
```

En ocasiones, es más útil mostrar todos los campos del formulario con su `label` alineada a la izquierda y el campo del formulario a la derecha de cada `label`, como muestra la siguiente imagen.



Para mostrar un formulario tal y como aparece en la imagen anterior no es necesario crear una tabla y controlar la anchura de sus columnas para conseguir una alineación perfecta. Sin embargo, sí que es necesario añadir un nuevo elemento (por ejemplo un `div`) que encierre a cada uno de los campos del formulario (`label` e `input`). El esquema de la solución propuesta es el siguiente



Por tanto, en el código HTML del formulario anterior se añaden los elementos :

```
<form>
  <fieldset>
  <legend>Alta en el servicio</legend>

  <div>
    <label for="nombre">Nombre</label>
    <input type="text" id="nombre" />
  </div>

  <div>
    <label for="apellidos">Apellidos</label>
    <input type="text" id="apellidos" size="35" />
  </div>
  ...
</fieldset>
</form>
```

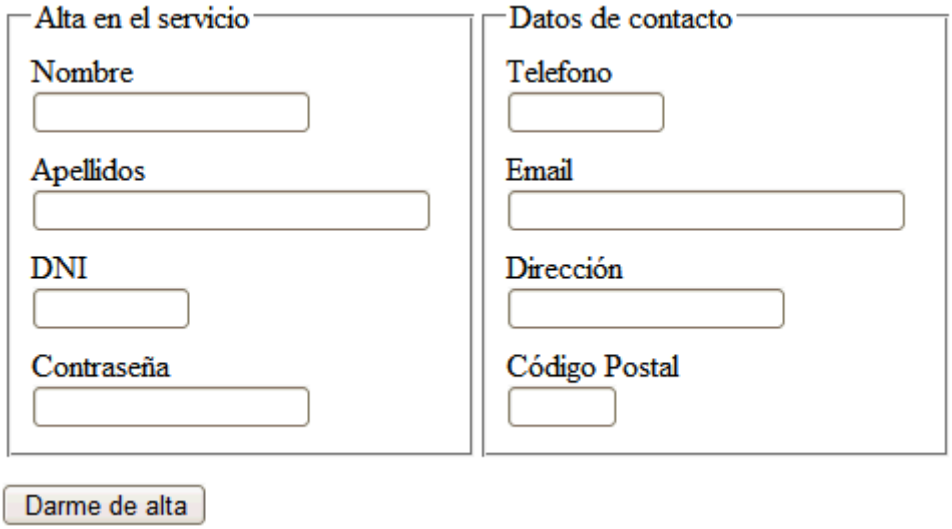
Y en el código CSS se añaden las reglas necesarias para alinear los campos del

formulario:

```
div {
  margin: .4em 0;
}
div label {
  width: 25%;
  float: left;
}
```

17.3 Formulario de varias columnas

Los formularios complejos con decenas de campos pueden ocupar mucho espacio en la ventana del navegador. Además del uso de pestañas para agrupar los campos relacionados en un formulario, también es posible mostrar el formulario a dos columnas, para aprovechar mejor el espacio.



La solución consiste en aplicar la siguiente regla CSS a los `fieldset` del formulario:

```
form fieldset {
  float: left;
  width: 48%;
}
```

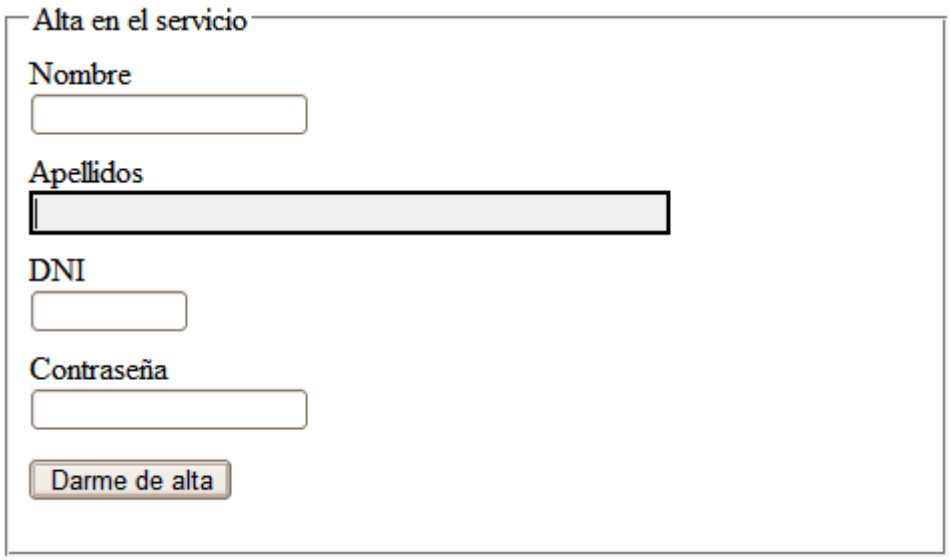
Luego repartimos los campos del formulario entre varios `fieldset`.

```
<form>
  <fieldset>
        ...
  </fieldset>
  <fieldset>
    ...
  </fieldset>
  ...
</form>
```

17.4 Resaltar un campo seleccionado

Una de las mejoras más útiles para los formularios HTML consiste en resaltar de alguna forma especial el campo en el que el usuario está introduciendo datos. Para ello, CSS define la pseudo-clase `:focus`, que permite aplicar estilos especiales al elemento que en ese momento tiene el foco o atención del usuario.

La siguiente imagen muestra un formulario que resalta claramente el campo en el que el usuario está introduciendo la información.



Añadiendo la pseudo-clase `:focus` después del selector normal, el navegador se encarga de aplicar esos estilos cuando el usuario activa el elemento.

```
input:focus {
  border: 2px solid #000;
  background: #F3F3F3;
}
```

Desafortunadamente, la pseudo-clase `:focus` no funciona en navegadores obsoletos como Internet Explorer 6

17.5 Botones con iconos

Un botón de formulario tiene un formato por defecto muy simple y podemos mejorarlo

mucho utilizando propiedades que ya conocemos. Observemos los siguientes botones

El primer botón tiene un icono y una etiqueta. El código CSS sería el siguiente

```
.boton {
    display: block;
    margin: .5em 0 0 0;

    background-image: url('icono1.png');
    background-repeat: no-repeat;
    background-position: center 20%;
    background-color: darkseagreen;
    padding-top: 50px;
    border: 2px outset cyan;
}
```

El segundo botón es el primero pero cuando se introduce el ratón. Hemos cambiado solamente el color de fondo para crear un efecto. Tendríamos que añadir la siguiente regla CSS para que este efecto se viera al introducir el ratón.

```
.boton:hover {
    background-color: lightgreen;
}
```

Como se puede observar, estamos empleando la pseudoclase `:hover` para aplicar un formato diferente al introducir el ratón. También, podemos aumentar el efecto anterior cambiando el icono del botón. Para que el entorno del botón quede mejor con el nuevo icono hemos cambiado también el color de fondo y del borde. La regla anterior se modificaría para quedar así.

```
.boton:hover {
    background-color: skyblue;
    border: 2px outset SteelBlue;
    background-image: url('icono2.png');
}
```

Si queremos, jugamos con las propiedades `padding` y `background-position` para poner el texto del botón a la derecha del icono. El botón tendría el siguiente aspecto

Habría que cambiar la primera regla CSS de la siguiente manera.

```
.boton {
    display: block;
    margin: .5em 0 0 0;

    background-image: url('icono1.png');
    background-repeat: no-repeat;
    background-position: 5% center;
    background-color: darkseagreen;
```

```
    padding-left: 60px;
    height: 60px;
    border: 2px outset cyan;
}
```

Como se puede apreciar, también hemos tenido que darle una altura al botón para evitar que el icono de fondo se vea recortado.

18 Elementos multimedia

Las páginas web se concibieron como documentos multimedia, capaces de mostrar todo tipo de recursos gráficos y de audio. Multimedia en sitios web se refiere al uso de la mezcla de medios (texto, imágenes, video, audio y animación) para comunicar un mensaje.

En la web, el mensaje puede ser mediante una presentación narrativa lineal como en una película que no permite al usuario alterar la narración, o puede ofrecer al usuario interactuar con el medio. La elección de que medios multimedia utilizar siempre estará sujeta a las necesidades de los usuarios del sitio web y como estos mejor satisfacen estas necesidades de información y entrega de contenido.

La historia de multimedia en la web corre en paralelo a los aumentos en la conectividad de Internet y con las mejoras en las tecnologías de codificación y compresión digital de los medios como video, audio y flash. En los primeros años de Internet, durante los años de prevalencia de conexiones conmutadas de 14.4 Kbps y 28.8 Kbps, cualquier intento de multimedia más allá de una imagen animada (GIF animada) fue recibida con frustración por parte de los usuarios de Internet.

Conforme los usuarios de Internet contaban con conexiones más rápidas, vimos la evolución de la multimedia en los sitios web. Los usuarios de Internet en países con mayor número de conexiones de alta velocidad están acostumbrados a la multimedia y se les resulta raro encontrar un sitio web sin este tipo de contenido.

Hoy estamos viviendo un increíble y emocionante auge en el uso de la multimedia en la web por el amplio uso de conexiones de alta velocidad en los paises más desarrollados y por el amplio uso de dispositivos de captura de audio y video en formato digital comprimido con la introducción de estos servicios en los teléfonos móviles.

En un capítulo anterior vimos la introducción de imágenes en nuestras páginas web. En este capítulo veremos como introducir video y audio.

18.1 Multimedia antes de HTML 5

El primer elemento disponible en HTML para los ficheros multimedia fue `embed`. Este elemento, que en realidad nunca llegó a estar incluido oficialmente en el lenguaje por la W3C, todavía funciona en todos los navegadores modernos, pero presenta algunos problemas, que al no estar sujeto a estandarización alguna, ha provocado que cada navegador interprete a su manera las cosas. En la especificación de HTML se recomienda el uso del elemento `object` para estos menesteres.

18.1.1 Elemento embed

Nestcape Navigator implementó la etiqueta `embed` para incorporar ficheros de audio. Es ésta una etiqueta de caracter general, que se usa para la inclusión en las páginas web de todos aquellos archivos ajenos al navegador y que necesitan por lo tanto la ejecución de algún plugin para su interpretación.

Paradójicamente, Internet Explorer asumió después el uso de esta etiqueta para la inclusión de ficheros de audio, para llegar a interpretarla mejor y ampliarla con más atributos y propiedades.

Este elemento nos va a incluir en la página web un panel con controles similares al de cualquier reproductor de audio: un botón Play, para comenzar la reproducción (si no está establecida a automática), un botón Pause, para detenerla momentáneamente y un botón Stop, para detenerla definitivamente (puesta a cero).

El elemento `embed` tiene los siguientes atributos

- ✔ `src="ruta_fichero"`, fija la ruta en la que se encuentra el fichero de audio a reproducir. La ruta puede ser relativa a nuestro sistema de carpetas local, absoluta respecto el sistema de carpetas del servidor web o una URL completa que localice le fichero en Internet.
- ✔ `loop="n | true | false"`, que determina el número de veces que se debe ejecutar el fichero de audio. Los valores admitidos son n (número entero de veces), true (infinitas veces) y false (sólo una vez). Sólo es reconocida por Netscape Navigator.
- ✔ `playcount="n"`, que define el número de veces (n) que se debe ejecutar en fichero de audio en el caso de Internet Explorer.
- ✔ `type="tipo_fichero"`, atributo importante, que declara el tipo de fichero de audio que estamos usando, con lo que el navegador web puede ejecutar el programa o plugin adecuado para la reproducción del fichero. Puede ser audio/midi, audio/wav, etc.
- ✔ `autostart="true/false"`, que determina si el fichero de audio debe empezar a reproducirse por sí sólo al cargarse la página o si por el contrario será preciso la actuación del usuario (o de código de script) para que comience la audición.
- ✔ `pluginspage="URL"`, que establece, en caso de ser necesario un plugin especial para reproducir el fichero, la página web donde se puede descragar el mismo. Sólo se activa en el caso de que el navegador no sea capaz de reproducir el fichero por sí mismo, y es soportada tan sólo por Netscape Navigator.
- ✔ `name="nombre"`, que asigna un nombre identificador (debe ser único en la página) a una etiqueta `embed` determinada, con objeto de ser accedida luego por lenguajes de script.
- ✔ `volume="v"`, que determina el volumen de reproducción del sonido, y que puede

variar entre 0 y 100. Es sólo soportada por Netscape Navigator, que en la consola muestra el valor establecido en su indicador de volumen, siendo su valor por defecto 50. En en caso de Internet Explorer, el valor del volumen por defecto es 50 siendo necesario actuar sobre el control de volumen de la consola para modificarlo.

- ✔ `hidden="true | false"`, que establece si la consola va a ser visible (`false`) o no (`true`). Es éste un aspecto polémico, ya que si ocultamos la consola obligamos al usuario a oir nuestro fichero, sin posibilidad de detenerlo ni de modificar el volumen, y si la mostramos estaremos incrustando en la pantalla un objeto que muchas veces nos romperá el esquema de diseño de nuestra página. Queda determinar su uso en cada caso concreto.
- ✔ `width="w"`, que determina el ancho visible de la consola, en pixels.
- ✔ `height="h"`, que determina el alto visible de la consola, en pixels.
- ✔ `align="top | bottom | center | baseline | left | right | texttop | middle | absmiddle | absbotom"`, análogo al del elemento `img`, define la alineación horizontal o vertical de la consola respecto de los elementos de la página.
- ✔ `hspace="hs"`, que establece la separación horizontal
- ✔ `vspace="vs"`, que establece la separación vertical, en pixels, entre la consola y los elementos de la página que la redean. Análoga a sus equivelentes del elemento <IMG>.

Por ejemplo, para reproducir audio tendríamos el siguiente elemento `embed`.

```
<embed src="minueto.mid" width=160 height=70 autostart="false">
```

18.1.2 El elemento object

Con objeto de normalizar la inclusión de ficheros no nativos en los navegadores web se decidió sustituir los diferentes elementos que realizaban este papel (`applet`, `bgsound`, `embed`, etc.), y que no pertenecían a los estándares web, por un elemento general, que fuera capaz de incrustar en el navegador todo tipo de ficheros. El elemento elegido en el estándar HTML 4.0 fué `object`, al que se dotó de suficientes atributos y flexibilidad para poder realizar correctamente su trabajo. Debido a esto, la propuesta ha sido usar el elemento `object` también para incluir ficheros de audio de todo tipo en las páginas web.

Ahora bien, la aceptación e implementación que el mismo ha tenido varía según el navegador en particular, así como en función del objeto a incrustar. Como regla general, válida no sólo para incrustar ficheros de sonido, sino también para otros tipos, el elemento `object` va a definir un objeto o componente externo encargado de la reproducción del fichero, que en el caso de Internet Explorer suele ser algún tipo de control ActiveX. Mediante `object` se instancia el objeto, se declara su URL y sus principales propiedades generales, y mediante un conjunto de elementos especiales, `param`, se le van pasando los valores que necesita para su correcto funcionamiento o para su configuración deseada.

La sintaxis general de la etiqueta object, para el caso de ficheros de sonido, es del tipo:

```
<objetc atributos>
<param name="nombre" value="valor">
<param name="nombre" value="valor">
...
</object>
```

Los principales atributos de `object`, en referencia a ficheros de audio, son:

`classid="identificador_objeto"`, URL del objeto o componente externo necesario para reproducir el fichero de audio o video.

`type="tipo_fichero"`, atributo importante, que declara el tipo de fichero de audio que estamos usando.

`width="w"`, que determina el ancho visible de la consola, en pixels.

`height="h"`, que determina el alto visible de la consola, en pixels.

`align="top | bottom | center | baseline | left | right | texttop | middle | absmiddle | absbotom"`, define la alineación horizontal o vertical de la consola respecto de los elementos de la página.

`hspace="hs"`, que establece la separación horizontal,

`vspace="vs"`, que establece la separación vertical, en pixels, entre la consola y los elementos de la página que la redean.

`autostart="true | false"`, que determina si el fichero de audio debe empezar a reproducirse por sí sólo al cargarse la página o si por el contrario será preciso la actuación del usuario para que comience la audición.

`standby="mensaje"`, que presenta en pantalla un mensaje al usuario mientras el fichero se carga.

En cuanto a los elementos `param`, los más importantes son:

```
<param name="FileName" value="url_fichero">
```

Determina la URL del fichero de audio a reproducir. No es necesario utilizar sólo ficheros WAV o MID, pudiendo reproducirse también ficheros MP3 o Real Audio.

```
<param name="autostart" value="true | false">
```

Indica al navegador si se debe empezar a reproducir el sonido automáticamente al cargar la página o si por el contrario será preciso que el usuario pulse el botón Play para ello.

Veamos el siguiente ejemplo:

```
<object width="150" height="175" type="audio/midi">
<param name="FileName" value="../sonidos/xfiles.mid">
<param name="autostart" value="true">
</object>
```

18.2 Vídeo en HTML5

La inserción de vídeos en una página web adolece de un problema distinto al de las imágenes. Los diferentes contenedores de vídeo y formatos hacen que, a veces, sea complejo asegurarnos de que toda nuestra audiencia sea capaz de visualizar el contenido que vamos a mostrar.

El contenedor de vídeo es el tipo de archivo que va contener el vídeo, las pistas de audio y otra información necesaria poder mostrar el vídeo correctamente. Hay muchos contenedores diferentes, como *MPEG-4* (archivos .mp4), *Flash* (.flv o .f4v), *Ogg Vorbis* (archivos .ogg), *WebM* (archivos .webm) o *AVI* (extensión .avi).

Cada uno de esos contenedores tiene sus ventajas y sus inconvenientes, es más abierto o presenta las limitaciones propias del software propietario.

Dentro de cada contenedor de vídeo se insertará el vídeo en cuestión. Dado el tamaño que suele ocupar una pequeña porción de vídeo, todos los contenedores almacenan el contenido con algún tipo de compresión. Es decir, el vídeo está codificado con algún método. Esta compresión se realiza mediante un *codec*. Así, en cada contenedor podemos encontrar uno o varios flujos de video y audio, cada uno codificado con un método diferente. Si nuestro ordenador no dispone de los codec adecuados, nos encontraremos con que no somos capaces de visualizar o escuchar el material en cuestión.

En resumen, el vídeo y el audio están codificados con un formato concreto y, a su vez, todo ello está insertado en un formato de archivo contenedor. Dentro de un formato contenedor no siempre nos encontraremos la misma codificación.

Todo esto genera una combinatoria enorme de formatos a los que hay que incorporar la variable del navegador. Cada navegador en la actualidad es capaz de reproducir un número concreto de formatos. A día de hoy, casi todas las nuevas versiones de los navegadores son capaces de reproducir vídeo en formato *WebM*. El formato *Ogg Vorbis*, por ejemplo, una muy buena alternativa por ser libre, en la actualidad no funcionaría en *Internet Explorer* ni en *Safari*.

18.2.1 Formatos de vídeos y soporte del navegador

En la siguiente table se muestran los formatos de vídeo más habituales soportados por los diferentes navegadores

Navegador	MP4	WebM	Ogg
Internet Explorer 9+	SI	NO	NO
Chrome 6+	SI	SI	SI

Firefox 3.6+	NO	SI	SI
Safari 5+	SI	NO	NO
Opera 10.6+	NO	SI	SI

Como podemos observar no hay un formato reconocible por todos los navegadores, así que a la hora de introducir vídeo en nuestra web deberemos indicar varios formatos para asegurarnos de que es visualizado correctamente.

18.2.2 Introducción de video en HTML5

Una de las características más mencionadas de HTML5 fue la capacidad de procesar video. A pesar de la necesidad y demanda del procesamiento de video por parte de los usuarios, los desarrolladores de los navegadores no le dieron la misma importantica. Para estos, introducir soporte de video en los navegadores no es una tarea trivial, pero a pesar de ello HTML5 ha introducido un elemento para reproducir video en una página web. El elemento `video` tiene una sintaxis muy sencilla y solo el atributo `src` es obligatorio.

```
<video src="url_video"></video>
```

El atributo `src` contendrá la URL donde se encuentra el archivo de video, con lo que podremos introducir aquí cualquier vídeo disponible en la red. Este elemento ofrece varios atributos para establecer su comportamiento y configuración. Los atributos `width` y `height`, al igual que en otros elementos HTML ya conocidos, declaran las dimensiones para el elemento o ventana del reproductor. El tamaño del video será automáticamente ajustado para entrar dentro de estos valores, pero no fueron considerados para redimensionar el video sino limitar el área ocupada por el mismo para mantener consistencia en el diseño.

No recomiendo utilizar el atributo `src` para especificar la fuente del video. Este atributo puede ser reemplazado por el elemento `source` y su propio atributo `src` para declarar varias fuentes con diferentes formatos y de esta manera asegurarnos de que el video será visualizado correctamente.

Veamos un ejemplo

```
<video width="420" >
    <source src="mov_bbb.mp4" type="video/mp4">
    <source src="mov_bbb.ogg" type="video/ogg">
    Su navegador no soporta vídeo en HTML5.
</video>
```

Como podemos apreciar, hemos incluido dentro de un elemento `video` dos elementos `source`. Cada uno de ellos hace referencia al mismo vídeo en dos formatos diferentes, mp4 y ogg. El navegador reproducirá el primero que tenga capacidad para ello. Si el navegador no tiene capacidad para reproducir ninguno de los formatos especificados, entonces aparecerá el texto dentro del elemento `video`.

El elemento vídeo tiene los siguientes atributos:

- ✔ `controls`, atributo vacío para que muestre los controles de reproducción de vídeo.

- ✔ `autoplay`, este atributo hace que el vídeo se reproduzca en cuanto esté listo.
- ✔ `preload`, hace que el vídeo se empiece a cargar en cuanto se accede a la página. Por el contrario, indicando `preload="none"`, no se cargará hasta que el usuario haga clic en el vídeo para su reproducción. Esta segunda opción es muy útil, si el vídeo no es el recurso fundamental de la página y vamos a tener personas que no estén interesadas en verlo, ya que nos ahorrará mucho ancho de banda.
- ✔ `loop`, atributo vacío para que el vídeo se reproduzca cíclicamente. Si no indicamos nada, al terminar su reproducción se detendrá.
- ✔ `muted`, el vídeo no tendrá sonido al comenzar.
- ✔ `poster`, atributo vacío para indicar al navegador que muestre una imagen mientras se espera a ser reproducido.

18.3 Audio en HTML5

Aunque los navegadores han sido capaces de interpretar los archivos de audio adecuados desde hace ya algunas versiones, es cierto que la aplicación de sonidos a las páginas web ha estado limitada desde siempre por el ancho de banda necesario en las conexiones a Internet para poder descargar de forma adecuada dichos archivos, debido al tamaño excesivo de los mismos.

Otra de las limitaciones importantes que encontramos a la hora de incluir archivos de sonido en nuestras páginas es la diferentes implementación que hacen de ellos los navegadores web más usados. En efecto, no sólo deberemos usar etiquetas HTML distintas para Internet Explorer que para Netscape Navigator, sino que a veces la forma misma de interpretar el sonido puede diferir de uno a otro navegador.

Por último, hay que destacar que a la hora de incluir archivos de audio en nuestras páginas debemos ser conscientes que muchos de los formatos usados, sobre todo en grabaciones de calidad, precisan un plugin o programa especial para su reproducción en el navegador cliente. Y si es cierto que actualmente hay ciertos plugins se han transformado casi en un estándar en Internet (como el de Real Audio o el de MP3), hay otros posibles que no es normal tener instalados, por lo que si incluimos ficheros de esos tipos obligaremos al usuario a tener que instalarlos, cosa a la que suele ser reacio.

18.3.1 Formatos de audio y soporte del navegador

En la siguiente table se muestran los formatos de audio más habituales soportados por los diferentes navegadores

Navegador	MP3	Wav	Ogg
Internet Explorer 9+	SI	NO	NO
Chrome 6+	SI	SI	SI
Firefox 3.6+	NO	SI	SI
Safari 5+	SI	SI	NO

Opera 10.6+	NO	SI	SI

Como podemos observar no hay un formato reconocible por todos los navegadores, así que a la hora de introducir audio en nuestra web deberemos indicar varios formatos para asegurarnos de que es reproducido correctamente.

18.3.2 Introducción de audio en HTML5

Disponemos del elemento `audio` el cual contiene los mismos atributos que el elemento `video`.

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Su navegador no soporta el elemento audio
</audio>
```

Al igual que ocurría con el vídeo, los navegadores no tienen capacidad para reproducir todos los formatos de audio. MP3 está bajo licencia comercial, por lo que no es soportado por navegadores como Firefox u Opera. Vorbis (el codificador de audio del contenedor OGG) es soportado por esos navegadores, pero no por Safari e Internet Explorer. Por esta razón, nuevamente debemos aprovechar el elemento `source` para proveer al menos dos formatos entre los cuales el navegador pueda elegir.

18.3.3 Incluir un video de Youtube

Una vez que hemos visto el método más complejo, pero que nos da más autonomía, analizaremos la alternativa de insertar un vídeo que previamente hayamos ubicado en una página web de vídeos como Youtube.

Insertar un recurso externo en nuestra página web tiene muchas ventajas:

- ✔ No ocuparemos el espacio de nuestro servidor, ni consumiremos nuestro ancho de banda; el recurso estará alojado en todo momento en el otro servidor.
- ✔ Estos servidores especializados suelen proporcionar más velocidad de descarga que el nuestro propio, por lo que en el caso de vídeos se reproducirán de forma más fluida.
- ✔ Cualquier cambio o actualización en el recurso se mostrará inmediatamente en nuestra página, sin necesidad de cambios.

Esto último es el único inconveniente de este método: que en cualquier momento, de forma unilateral, el propietario del material puede optar por cambiarlo o retirarlo, con lo que perderíamos el acceso. No obstante, si somos los propietarios, esto deja de ser un inconveniente.

Técnicamente lo que hacemos es similar a crear una pequeña ventana dentro de nuestra página web, para que en su interior se muestre un material que está alojado en otro servidor. Es el otro servidor el que decide qué método empleará, por lo que no nos debemos preocupar, ya que sólo tendremos que copiar y pegar el código HTML que genere

el servidor.

Para incrustar un video de Youtube seguir estos pasos:

1. Acceder a Youtube y seleccionar un vídeo.
2. Hacer clic en el enlace *Compartir*. La dirección que aparece se puede emplear en diferentes sitios, como blogs, herramientas sociales,etc.
3. En nuestro ejemplo hacer clic en el enlace *Insertar* que aparece más abajo. Copiar el pequeño código HTML que se despliega; será similar a esto que se recoge en la figura:

Figura 77.- Compartir un vídeo Youtube

4. Un poco más abajo podemos personalizar las dimensiones de la ventana y otros parámetros. Tras realizar los cambios, copiar el código HTML.
5. Editar la página web y pegar el código.
6. Al probar la página web en un navegador, obtendremos toda la potencia de un servidor como YouTube recogida en nuestra página web.

19 Mapas

Los mapas de imágenes permiten especificar regiones en una imagen u objeto y asignar una acción específica a cada región (p.ej., abrir un documento, ejecutar un programa,

etc.). Cuando la región es activada por el usuario, se ejecuta la acción.

Un mapa de imágenes se crea asociando un objeto con una especificación de las áreas geométricas sensibles del objeto.

Hay dos tipos de mapas de imágenes:

- ✔ En el lado del cliente ➔ Cuando un usuario activa una región de un mapa de imágenes en el lado del cliente con un ratón, las coordenadas en píxeles son interpretadas por el agente de usuario. El agente de usuario selecciona el vínculo especificado por la región activada y lo sigue. Esta sección se dedica a este tipo de mapa.
- ✔ En el lado del servidor ➔ Cuando un usuario activa una región de un mapa de imágenes en el lado del servidor, las coordenadas en píxeles son enviadas al agente del lado del servidor especificado por el atributo `href` del elemento A. El agente del servidor interpreta las coordenadas y realiza alguna acción.

Se prefieren los mapas de imágenes en el cliente que los mapas de imágenes en el servidor por dos razones: son accesibles a las personas que utilizan navegadores no gráficos y permiten saber en todo momento si el apuntador está sobre una región activa o no.

Los mapas son imágenes activas asociadas a enlaces (o acciones). Cuando el usuario hace un clic sobre un punto del mapa, recibe como respuesta del servidor una página determinada (ya sea mapa en modo local o en modo servidor).

19.1 Mapas en modo cliente. Elementos <map> y <area>

El elemento `map` especifica un mapa de imágenes en el lado del cliente que puede ser asociado con otros elementos (`img`, `object`, o `input`). Un mapa de imágenes se asocia a un elemento a través del atributo `usemap` del elemento.

Dentro de un elemento `map` se incluyen uno o más elementos `area`. Estos elementos no tienen contenido, sino que especifican las regiones geométricas del mapa de imágenes y los vínculos asociados con cada región. Obsérvese que en general los navegadores no representan los elementos `area`. Por tanto, se debe proporcionar texto alternativo para cada `area` con el atributo `alt`.

Si dos o más regiones se superponen, tiene prioridad la región definida por el elemento que aparece antes en el documento.

Los navegadores deberían ofrecer alternativas textuales a los mapas de imágenes gráficos para los casos en que los gráficos no estén disponibles o en que el usuario no pueda acceder a ellos. Por ejemplo, los navegadores pueden usar el texto `alt` para crear vínculos textuales en lugar de un mapa de imágenes gráfico. Estos vínculos pueden ser activados de diferentes maneras (con el teclado, activación por voz, etc.).

19.2 Crear un mapa

La sintaxis que soporta esta forma de trabajar se basa en la inclusión del atributo `usemap` en el elemento `img` que indica que se trata de una imagen que se va a emplear como un mapa cliente.

```
<img src="/images/mapa.gif" usemap="#mapa">
```

Esta directiva carga la imagen `mapa.gif`, y le asigna las áreas de mapa, definidas en el mismo documento HTML, en la parte denominada `#mapa`.

La definición de mapa cliente está limitada por la directiva `map`. En su atributo `name` se inserta el nombre de la definición (en nuestro caso `#mapa`). Dentro de esta sección de código, se incluyen las directivas `area` que definen cada una de las áreas de la imagen.

El elemento `area` tiene los siguientes atributos:

`shape = default | rect | circle | poly`

Este tributo especifica la forma de una región. Valores posibles:

- ✔ default: Especifica la región completa.
- ✔ rect: Define una región rectangular.
- ✔ circle: Define una región circular.
- ✔ poly: Define una región poligonal.

`coords = coordenadas`

Este atributo especifica la posición y la forma en la pantalla. Las coordenadas van separadas por comas sin espacio entre ellas, de la figura que incluye el atributo `shape` o área que define. El número y orden de estos valores depende de la forma que está siendo definida. Combinaciones posibles:

- `rect: x-izquierda, y-superior, x-derecha, y-inferior`. Area rectangular. Se especifican las coordenadas de las esquinas superior izquierda y la inferior derecha, definiéndose así el área rectangular
- `circle: x-centro, y-centro, radio`. Área circular. Se debe especificar en primer lugar las coordenadas del centro y continuación el radio en un valor de puntos. Cuando el valor del radio sea un valor porcentual, los navegadores deberían calcular el valor final del radio basándose en la anchura y altura del objeto asociado. El radio debería ser el menor valor de los dos.
- `poly: x1, y1, x2, y2, ..., xN, yN`. Área poligonal. Se debe especificar las coordenadas de todos los vértices del polígono. El primer par de coordenadas x, y debería ser igual al último para cerrar el polígono. Cuando estas coordendas no sean iguales, los navegadores deberían añadir un par adicional de coordenadas para cerrar el polígono. Las coordenadas son relativas a la esquina superior izquierda del objeto. Todos los valores son longitudes. Todos los valores van separados por comas.

`nohref`

Si está establecido, este atributo booleano especifica que una región no tiene asociado ningún vínculo.

`href="url"`

Indica la URL a la que se accede si se pulsa sobre la zona delimitada por el área indicada, con el ratón.

`target="destino"`

Indica el nombre del marco en el que se cargaría la página web definida en el atributo href.

Es importante (aunque en absoluto necesario), definir una última área que abarque la totalidad del gráfico que empleamos como imagen mapa, para direccionar a una URL "por defecto", con el objeto de contemplar el caso de que se pulse sobre un área no definida en el código. Esto como ya he dicho no es necesario, ya que si una determinada área de la imagen no quedara definida, como no lo está, al pinchar sobre ella, no pasaría nada.

Normalmente se suele definir el URL "por defecto", sólo en caso de un mapa interactivo, con excesivas áreas definidas, y muy próximas entre sí, para indicar que la zona pinchada, no es la que visualmente se pretendía indicar... un ejemplo sería una imagen del mundo en la que Europa se viera muy pequeña, y al querer pinchar sobre las Islas Baleares, y debido al pequeño tamaño de la imagen, pinchamos en Córcega...

Para este ejemplo emplearemos elemento `area` para definir las zonas sensibles. En este caso se supondrá una casa compuesta de un cuadrado, con un triángulo encima, un rectángulo vertical de puerta, y con un árbol al lado de la casa.

```
<img src="/images/casa.gif" usemap="#casa">
<map name="casa">
<!-- Define el área del tejado por medio de un triángulo -->
<area shape="poly" coords="2,62,57,62,28,1" href="tejado.htm">

<!-- Define el área de la puerta por medio de un rectángulo-->
<area shape="rect" coords="21,101,35,138" href="puerta.htm">

<!-- Define el área de la casa por medio de un triángulo -->
<area shape="rect" coords="2,64,57,138" href="casa.htm">

<!-- Define el área de la copa del árbol por medio de un circulo
-->
<area shape="circle" coords="80,76,21" href="arbol.htm">

<!-- Define el área del tronco del árbol por medio de un
rectángulo -->
<area shape="rect" coords="78,98,85,138" href="tronco.htm">

<!-- Esta última área, es el área por defecto si se pincha en
una zona que no sea un área definida, y envía a nuestro
visitante a la pagina "sin_enlace.htm" -->
<area shape="rect" coords="0,0,96,138" href="sin_enlace.htm">
</map>
```

20 Transiciones, transformaciones y animaciones

Una de las novedades de CSS3 que más dinamismo aporta a los elementos de una página web son las transiciones y las transformaciones. Estas nos permitan crear

animaciones en los elementos de la página web con lo que se podría llegar a sustituir otras tecnologías utilizadas hasta la fecha para estos mismos efectos, como Flash.

En este capítulo vamos a ver como crear estos efectos que añaden más atractivo a una página web al dotar de movimiento a los elementos que la componen.

20.1 Transformaciones

Una transformación es un efecto que permite a los elementos de una página web cambiar de tamaño, forma y posición. Con la versión 3 de CSS podemos hacer transformaciones en 2D y 3D. Aquí nos centraremos en las primeras.

Las transformaciones 2D requieren IE10, Firefox, y Opera. Chrome y Safari necesitan el prefijo -webkit- e IE9 el prefijo -ms-. Mediante el uso de una serie de métodos podemos realizar las siguientes transformaciones

20.1.1 Mover un elemento

El método `translate(x,y)` mueve un elemento de su actual posición a una diferente. La posición final es establecida por los valores de `x` e `y`. Por ejemplo, supongamos que tenemos el siguiente elemento en una página web.

```
<div id="muevete">
Hola, ahora estoy quietecita, pero si introduces el ratón aquí
verás como me muevo.
</div>
```

Este elemento tiene el siguiente formato CSS

```
#muevete {
    position: absolute;
    width: 200px;
    height: 200px;
    top: 50px;
    left: 100px;
    background-color: cyan;
}
#muevete:hover {
    transform: translate(100px,25px);
}
```

En este ejemplo hemos creado una capa simple a la que hemos añadido una transformación cuando el ratón se introduce en ella (pseudoclase `:hover`). La capa se desplazará 100 píxeles en el eje x y 25 píxeles en el eje y desde su posición inicial. Al sacar el ratón regresará al lugar donde se encontraba inicialmente.

También disponemos de los métodos `translateX(n)` y `translateY(n)` para mover un elemento únicamente en el eje x o y respectivamente. El parámetro `n` indica el desplazamiento.

20.1.2 Rotar un elemento

El método `rotate(ángulo)` define el giro de un elemento. El ángulo es en grados. Si es positivo el giro se produce en el sentido de las agujas del reloj, si es negativo, en sentido contrario. Veamos un ejemplo.

```
<div id="gira">
Hola, ahora estoy quietecita, pero si introduces el ratón aquí
verás como giro un poquito. Y si sacas el ratón recupero mi
forma inicial.
</div>
```

Esta capa tiene el siguiente formato CSS

```
#gira {
    position: absolute;
    width: 200px;
    height: 200px;
    top: 50px;
    left: 100px;
    background-color: cyan;
}

#gira:hover {
    transform: rotate(25deg);
}
```

20.1.3 Escalar un elemento

Escalar un elemento consiste en cambiar su tamaño. El método `scale(x,y)` modifica el tamaño de un elemento cambiando el ancho y alto. El parámetro `x` corresponde al aumento o disminución del ancho y el parámetro `y` al de la altura. Estos valores indican el aumento o disminución relativo al tamaño actual en tantos por uno. Por ejemplo, si queremos reducir alguna dimensión a la mitad tendremos que indicar 0.5 y si queremos el doble tendremos que poner 2. Si solamente queremos cambiar una dimensión también disponemos de los métodos `scaleX(n)` y `scaleY(n)`. Veamos un ejemplo.

```
<div id="aumenta">
Hola, ahora estoy quietecita, pero si introduces el ratón aquí
verás como crezco. Y si sacas el ratón recupero mi forma
inicial.
</div>
```

A esta capa le damos el siguiente formato CSS

```
#aumenta {
    position: absolute;
    width: 200px;
    height: 200px;
    top: 50px;
    left: 100px;
```

```
    background-color: cyan;
}

#aumenta:hover {
    transform: scale(2,1.25);
}
```

En el ejemplo anterior hemos aumentado el ancho el doble y un 25% la altura.

20.1.4 Giro en dos ejes

El método `skew(x,y)` hace al elemento girar en el eje X y el eje Y. Los parámetros son ángulos. Si son positivos el giro se produce en el sentido de las agujas del reloj y si es negativo en sentido contrario. También tenemos los métodos `skewX(x)` y `skewY(y)` para provocar el giro en un solo eje. Veamos otro ejemplo.

```
<div id="girados">
Hola, ahora estoy quietecita, pero si introduces el ratón aquí
verás como me deformo. Y si sacas el ratón recupero mi forma
inicial.
</div>
```

De nuevo, el formato CSS para esta capa

```
#girados {
    position: absolute;
    width: 200px;
    height: 200px;
    top: 50px;
    left: 100px;
    background-color: cyan;
}

#girados:hover {
    transform: skew(20deg,40deg);
}
```

20.1.5 Cambiándolo todo

El método matrix() permite combinar todas las transformaciones 2D anteriores a la vez. Recibe seis parámetros conteniendo las funciones matemáticas que permiten escalar, mover y girar en dos ejes.

- ✔ `matrix (1, 0, 0, 1, x, y)` equivale a translate x e y
- ✔ `matrix (x, 0, 0, y, 0, 0)` equivale a scale x e y
- ✔ `matrix (1, y, x, 1, 0, 0)` equivale a skew x e y

Veamos un ejemplo

```
<div id="todo2D">
```

```
Hola, ahora estoy muy normal. Introduce el ratón y tendré cuatro
transformaciones simultáneas.
</div>
```

El código CSS es el siguiente

```
#todo2D {
    position: absolute;
    width: 200px;
    height: 200px;
    top: 50px;
    left: 100px;
    background-color: cyan;
}

#todo2D:hover {
    transform: matrix(1.5, 0.1, 0.2, 1.5, 200, 10);
}
```

En el ejemplo anterior hemos aumentado la capa un 50% en alto y ancho. La hemos movido 200 píxeles en el eje X y 10 en el eje Y. Por último la hemos girado 0.1 en el eje X y 0.2 en el eje Y.

20.2 Transiciones

Una transición establece un cambio progresivo de una o varias propiedades CSS de un elemento durante un periodo de tiempo determinado. Disponemos de cuatro propiedades para definir una transición y de una propiedad abreviada. Vamos a verla deternidamente.

Con la propiedad `transition-property` definimos el nombre de una propiedad CSS para la cual se define la transición, es decir, la transición comenzará cuando el valor de esta propiedad cambie. Admite los siguientes valores:

- ✔ `all`, indica que la transición se aplicará a cualquier propiedad. Es el valor por defecto.
- ✔ `none`, la transición no se aplicará a ninguna propiedad.
- ✔ *`propiedades_CSS`*, lista de propiedades CSS separadas por coma a las que se le aplica la transición.

La propiedad `transition-duration` establece el tiempo en el que se demorará la transición. Las unidades válidas son segundos (s), minutos (m), milisegundos (ms). Valores negativos son inválidos, se computan como 0 y este significa que no se produce transición. Si son varias las propiedades indicadas en la propiedad `transition-property` se pueden declarar un tiempo para cada una.

La propiedad `transition-timing-function` determina cómo se desarrolla la transición en el tiempo estipulado. Admite los siguientes valores:

- ✔ `linear`, la transición tendrá la misma velocidad de principio a fin.

- ✔ `ease`, la transición comienza lenta, luego acelera y finalmente acaba lenta. Es el valor por defecto.
- ✔ `ease-in`, la transición comienza lenta.
- ✔ `ease-out`, la transición acaba lenta.
- ✔ `ease-in-out`, la transición comienza y acaba lenta.
- ✔ `steps(n)`, la transición no es lineal, sino a saltos. El valor de n especifica cuantos saltos se producen dentro del periodo de tiempo especificado.
- ✔ steps(n, start), la transición es a saltos los cuales se realizan al inicio de cada fracción de tiempo. Por ejemplo, si la propiedad `transition-duration` está establecida a `4s` e indicamos 5 saltos, entonces habrá un salto cada 4/5=0.8 segundos y al principio de cada fracción.
- ✔ `steps(end)`, igual que la anterior, pero la transición ocurre al final de cada fracción de tiempo.

La propiedad `transition-delay` indica un retraso en el comienzo de la transición. Por defecto no espera nada, la transición comienza en cuanto la propiedad cambia. Un valor negativo actúa al contrario, adelanta la transición al punto que le correspondería estar al haber trascurrido dicho tiempo. Por ejemplo, si definimos un cambio de anchura de 0% al 100% en 4 segundos, y la propiedad `transition-delay` está definida con el valor `-2s` fuerza a comenzar de inmediato la transición comenzando en el 50% de la anchura y el resto lo completará con el tiempo definido en la propiedad `transition-duration`.

Por último, disponemos de la propiedad abreviada `transition` para indicar las cuatro propiedades anteriores en el siguiente orden: `property duration timing-function delay;`

Veamos un ejemplo. Supongamos que tenemos una capa sencilla como la siguiente

```
<div id="ejemplo1">
Transición linear
</div>
```

Esta capa tiene el siguiente formato CSS.

```
#ejemplo1 {
    background-color: #FFFFAA;
    color: darkgray;
    width: 10em;
    text-align:right;
    padding: 5pt;

    transition-property: width,background-color,color;
    transition-duration: 4s;
    transition-timing-function: ease;
}
```

```
#ejemplo1:hover {
    background-color: #2222AA;
    color: bisque;
    width: 100%;
}
```

Si lo probamos veremos que cambia la anchura, el color de fondo y el color de primer plano. Podemos apreciar las diferentes funciones que determinan la transición variando la propiedad `transition-timing-function`.

Podemos modificar este ejemplo para ver como podemos aplicar diferentes tiempos y funciones de transición a diferentes propiedades. Solamente tenemos que cambiar el valor de estas propiedades.

```
transition-property: width,background-color,color;
transition-duration: 2s, 3s, 4s;
transition-timing-function: steps(5), ease, linear;
```

Vemos que hay tres propiedades a cambiar y cada una tiene su duración y función.

Por último, también podemos realizar transiciones con las transformaciones que vimos en el apartado anterior. En el siguiente ejemplo cargamos una imagen

```
<img id="gira_imagen" src="logo.png"/>
```

Ahora definimos estas propiedades CSS para esta imagen

```
#gira_imagen {
    margin-top: 5pt;
    transition: transform ease-in 3s;
}

#gira_imagen:hover {
    transform: rotate(720deg);
}
```

Como podemos observar le hemos aplicado una transformación de giro en la que va a dar dos vueltas completas en 3 segundos.

La siguiente tabla muestra las propiedades a las que se les puede aplicar transiciones.

Propiedad	Tipo
background-color	color
background-position	percentage, length
border-bottom-color	color
border-bottom-width	length
border-left-color	color
border-left-width	length

border-right-color	color
border-right-width	length
border-spacing	length
border-top-color	color
border-top-width	length
bottom	length, percentage
clip	rectangle
color	color
crop	rectangle
font-size	length, percentage
font-weight	font weight
height	length, percentage
left	length, percentage
letter-spacing	length
line-height	number, length, percentage
margin-bottom	length
margin-left	length
margin-right	length
margin-top	length
max-height	length, percentage
max-width	length, percentage
min-height	length, percentage
min-width	length, percentage
opacity	number
outline-color	color
outline-offset	integer
outline-width	length
padding-bottom	length
padding-left	length
padding-right	length
padding-top	length
right	length, percentage
text-indent	length, percentage
text-shadow	shadow
top	length, percentage
vertical-align	length, percentage
visibility	visibility
width	length, percentage

word-spacing	length, percentage
z-index	integer

20.3 Animaciones

La versión 3 de CSS incluye animaciones gráficas que pueden sustituir a las imágenes animadas o las películas Flash. Una animación es un efecto que permite a un elemento cambiar gradualmente desde un estilo a otro. Puedes cambiar tantos estilos como quieras y tantas veces como se quiera.

Las animaciones consisten en dos componentes: una hoja de estilo describiendo la animación y un conjunto de fotogramas clave que indican el estado de comienzo y de fin del estilo de la animación, junto con los estados intermedios entre el principio y el final.

Hay tres ventajas a la hora de usar animaciones CSS en lugar de las tradicionales técnicas de animación:

- ✔ Las animaciones simples son muy fáciles de crear y utilizar. No es necesario conocer ningún lenguaje de script.
- ✔ Las animaciones se ejecutan bien incluso en ordenadores con una carga moderada.
- ✔ Dejar al navegador el control de la secuencia de la animación le permite optimizar el rendimiento y la eficiencia. Por ejemplo, reduciendo la frecuencia de actualización de animaciones ejecutándose en pestañas que no están actualmente visibles.

IE10, Firefox y Opera soportan las animaciones. Chrome y Safari requiere el prefijo -webkit- en las propiedades de la animación.

20.3.1 La regla @keyframes

La regla @keyframes se emplea para definir la animación. Está formado por la palabra clave `@keyframes` seguida con un espacio en blanco del nombre que hayamos elegido para esa animacióne. A continuación apertura y cierre de llaves `{}` y dentro el conjunto de reglas que queramos declarar. Sería algo así

```
@keyframes aparece {
}
```

El nombre de la animación `aparece` identifica a la animación y se empleará para aplicarla al elemento que deseemos.

Las reglas que se definen dentro de `@keyframes` también tienen su sintaxis característica. Cada frame o paso está compuesta por un valor en porcentaje y encerradas entre llaves los pares `propiedad:valor` que deseemos. Cada frame especifica cuando ocurre el cambio. 0% es el principio de la animación y 100% es el final. Las palabras clave `from` y `to` equivalen 0% y 100%. Para un mejor soporte del navegador, deberías siempre definir tanto el 0% como el 100%. Veamos un ejemplo.

```
@keyframes aparece {
```

```
    0% {
        left: 0%;
        opacity: 0;
    }

    100% {
        left: 100%;
        opacity: 1;
    }
}
```

En el ejemplo hemos continuado con la definición de la animación `aparece`. En esta hemos definido dos frames. El primero es el inicio de la animación e indica que el elemento se encuentra en el extremo izquierdo de la pantalla y no se ve por tener la propiedad `opacity` a 0.

El segundo frame situa al elemento en el extremo derecho de la pantalla y se ve completamente al tener la propiedad `opacity` a 1.

Por tanto la animación anterior hará que el elemento se mueva desde la izquierda a la derecha y vaya apareciendo poco a poco. La animación ya está definida. Ahora solo queda asignarla a algún elemento. Por ejemplo a una imagen.

```
<img id="imagen_animada" src="logo.png"/>
```

Su formato CSS es el siguiente

```
#imagen_animada {
    position: absolute;
    animation: aparece 3s infinite;
}
```

Como se puede apreciar en el ejemplo hemos asignado la animación `aparece` al elemento `img` con `id=imagen_animada`. Para ello hemos utilizado la propiedad abreviada `animation` donde le indicamos el nombre de la animación, la duración y que la reproduzca indefinidamente.

En el ejemplo anterior solamente hemos definido un frame al principio y otro al final. Podemos hacer algún cambio y definir un frame en medio para que la imagen vaya de un extremo a otro y regrese, mientras aparece y desaparece. Quedaría así

```
@keyframes aparece {
    from {
        left: 0%;
        opacity: 0;
    }

    50% {
        left: 100%;
        opacity: 1;
    }
```

```
    to {
        left: 0%;
        opacity: 0;
    }
}
```

Ahora hemos usado las palabra clave `from` y `to` para marcar el principio y el final. El frame del 50% en medio hará que la animación vaya desde el estado de `from` a la mitad y desde la mitad al final.

Para asignar una animación a un elemento disponemos de las siguientes propiedades.

Propiedad	Tipo
`animation-name`	Nombre de la animación
`animation-duration`	Duración de la animación en segundos (s) o milisegundos (ms).
`animation-timing-function`	Describe como progresará la animación. Se admiten los mismos valores que para la propiedad transition-timing-function
`animation-delay`	Especifica cuando comenzará la animación
`animation-iteration-count`	Especifica cuantas veces se reproduce la animación. Por defecto 1. Puedes indicar la palabra clave `infinite` para indicar que se reproduzca indefinidamente.
`animation-direction`	Define si hay alternancia o no en el sentido en que se desarrollan cada ciclo de la animación. Así, si su valor es `alternate` los ciclos impares se desarrollan en el sentido indicado y los pares lo hacen en sentido contrario.
`animation-play-state`	Indica si la animación se está reproduciendo o en pausa.
`animation-fill-mode`	Especifica como una animación debería aplicar estilos al elemento asignado antes y después de la animación. Los posibles valores son: ✔ `none`, no se aplica ningún estilo antes ni después de la animación. ✔ `forwards`, el elemento retendrá el estilo del último frame de la animación. Este frame depende de los valores de las propiedades `animation-direction` y `animation-iteration-count`. ✔ `backwards`, el elemento retendrá el estilo del primer frame de la animación cuando esta comienza y durante el tiempo establecido en la propiedad `animation-delay`. Este frame depende de los valores de las propiedades

	animation-direction. ✔ both,

La siguiente table esquematiza cuál es el último frame que se aplica cuando el valor de la propiedad `animation-fill-mode` esta establecida al valor `forwards`.

animation-direction	animation-iteration-count	Último frame
normal	Par o impar	100% o to
reverse	Par o impar	0% o from
alternate	par	0% o from
alternate	impar	100% o to
alternate-reverse	par	100% o to
alternate-reverse	impar	0% o from

La siguiente tabla esquematiza cuál es el primer frame que se aplica cuando el valor de la propiedad `animation-fill-mode` esta establecida al valor `backwards`.

animation-direction	Primer frame
normal o alternate	0% o from
reverse o alternate-reverse	100% o to

La propiedad abreviada animation permite establecer estas propiedades según la siguiente sintaxis.

```
animation:name duration timing-function delay iteration-count
direction;
```

A partir de ahora se recomienda ir poco a poco. Crear animaciones sencillas e ir complicándolas conforme vayamos cogiendo práctica.

21 Marcos

Los marcos son divisiones que se efectúan en la pantalla del navegador. En cada una de estas divisiones podemos cargar un documento web independiente. Los marcos es una técnica para subdividir la pantalla del navegador en diferentes ventanas. Cada una de estas ventanas se podrá manipular por separado, permitiéndonos mostrar en cada una de ellas una página web diferente. Esto es muy útil para, por ejemplo, mostrar permanentemente en una ventana un índice de contenidos de nuestra página, y en otra ventana mostrar el contenido seleccionado.

Los marcos están desaconsejados por el W3C ya que han quedado obsoletos. Se han incluido en HTML5 por compatibilidad con las aplicaciones desarrolladas con versiones anteriores, pero en un futuro es posible que no estén contemplados.

Emplear marcos es incompatible con el elemento `body`, por lo que como se verá más adelante, esta directiva debe de no considerarse en la página que define los marcos, de lo contrario no funcionará.

Si el navegador no puede mostrar marcos o está configurado para no mostrarlos, representará los contenidos del elemento `noframes`.

Aquí tenemos un ejemplo de documento simple con marcos:

```
<!DOCTYPE html>
<html lang=”es”>
    <head>
         <title>Un documento simple con marcos</title>
    </head>
    <frameset cols="20%, 80%">
         <frameset rows="100, 200">
              <frame src="contenidos_del_marco1.html">
              <frame src="contenidos_del_marco2.html">
         </frameset>
         <frame src="contenidos_del_marco3.html">
         <noframes>
         <p>Este conjunto de marcos contiene:
         Marcos en documentos HTML
         <ul>
              <li><A href="contenidos_del_marco1.html">Marco
1</a>
              <li><a href=”contenidos_del_marco2.html”>Marco
2</a>
              <li><A href="contenidos_del_marco3.html">Marco
3</a>
         </ul>
         </noframes>
    </frameset>
</html>
```

21.1 Disposición de los marcos

Los documentos HTML que describen una disposición de marcos (llamados documentos con marcos) tienen una estructura diferente a la de los documentos HTML sin marcos. Un documento normal tiene un elemento `head` y un elemento `body`. Un documento con marcos tiene un elemento `head`, y un `frameset` en lugar del `body`.

El elemento `frameset` de un documento especifica la disposición de las vistas en la ventana principal del navegador. Además, `frameset` puede contener un elemento `noframes` que proporcione contenido alternativo para los navegadores que no soporten marcos o que estén configurados para no mostrar marcos.

Los elementos que normalmente podrían colocarse en el elemento `body` no deben aparecer antes del primer elemento `frameset` o no será tenido en cuenta.

21.1.1 Elemento frameset

El elemento `frameset` tiene los siguientes atributos

Atributo	Descripción
`rows = lista de multilongitudes`	Este atributo especifica la disposición de los marcos horizontales. Es una lista de longitudes en píxeles, porcentajes o longitudes relativas, separadas por comas. El valor por defecto es 100%, que significa una fila.
`cols = lista de multilongitudes`	Este atributo especifica la disposición de los marcos verticales. Es una lista de longitudes en píxeles, porcentajes o longitudes relativas, separadas por comas. El valor por defecto es 100%, que significa una columna.

El elemento `frameset` especifica la organización de la ventana principal del usuario en términos de subespacios rectangulares.

Cuando se establece el atributo `rows` (filas) se define el número de subespacios horizontales. Cuando se establece el atributo `cols` (columnas) se define el número de subespacios verticales. Ambos atributos se pueden especificar simultáneamente para crear una cuadrícula.

Si no se establece el atributo `rows`, cada columna se extiende a lo largo de toda la longitud de la página. Si no se establece el atributo `cols`, cada fila se extiende a lo largo de toda la anchura de la página. Si no se establece ninguno de los dos atributos, el marco tiene exactamente el mismo tamaño que la página.

Los marcos se crean de izquierda a derecha para las columnas y de arriba a abajo para las filas. Cuando se especifican ambos atributos, las vistas se crean de izquierda a derecha en la fila superior, de izquierda a derecha en la segunda fila, etc.

En este primer ejemplo dividimos la pantalla verticalmente en dos (es decir, creamos una mitad superior y una mitad inferior).

```
<frameset rows="50%, 50%">
...el resto de la definición...
</frameset>
```

El siguiente ejemplo crea tres columnas: la segunda tiene una anchura fija de 250 píxeles (lo cual es útil, por ejemplo, para incluir una imagen de tamaño conocido). La primera recibe el 25% del espacio restante, y la tercera el 75% del espacio restante.

```
<frameset cols="1*,250,3*">
...el resto de la definición...
</frameset>
```

El siguiente ejemplo crea una cuadrícula de 2x3 subespacios.

```
<frameset rows="30%,70%" cols="33%,34%,33%">
```

```
...el resto de la definición...
</frameset>
```

Para el siguiente ejemplo, supongamos que la ventana del navegador tiene actualmente una altura de 1000 píxeles. Para la primera vista se asigna el 30% de la altura total (300 píxeles). Para la segunda vista se especifica que tenga una altura de exactamente 400 píxeles. Esto deja 300 píxeles para repartir entre los otros dos marcos. La altura del cuarto marco se ha especificado como "2*", de modo que es el doble de alto que el tercer marco, cuya altura es sólo "*" (equivalente a 1*). Por tanto el tercer marco será de 100 píxeles de alto y el cuarto tendrá una altura de 200 píxeles.

```
<frameset rows="30%,400,*,2*">
...el resto de la definición...
</frameset>
```

Las longitudes absolutas que no sumen el 100% del espacio real disponible deberían ser ajustadas por los navegadores.

Cuando sobre espacio, el espacio sobrante debería repartirse proporcionalmente entre cada vista. Cuando falte espacio, debería reducirse cada vista en función de la relación entre el espacio especificado y el espacio total.

21.1.2 Contenido de un marco. Elemento frame

Mientras que el elemento `frameset` se emplea para dividir la pantalla en marcos, el elemento `frame` define el contenido y la apariencia de un marco. Dispone los siguientes atributos:

Atributo	Descripción
`name = “nombre”`	Este atributo asigna un nombre al marco actual. Este nombre puede utilizarse como el destino de enlaces o formularios
`longdesc = “uri”`	Este atributo especifica un vínculo a una descripción larga del marco. Esta descripción debería complementar la descripción corta proporcionada por el atributo `title`, y puede ser particularmente útil para navegadores no visuales.
`src = “uri”`	Este atributo especifica la localización de los contenidos iniciales que contendrá el marco.
`noresize`	Si está presente, este atributo vacío le dice al navegador que la ventana del marco no debe ser redimensionable.
`scrolling = auto | yes | no`	Este atributo especifica información sobre la barra de desplazamiento de la ventana del marco. Valores posibles: ✔ `auto`: Este valor le dice al agente de usuario que proporcione mecanismos de desplazamiento en la ventana del marco cuando sea necesario. Este es el valor por

	defecto. ✔ yes: Este valor le dice al agente de usuario que siempre proporcione mecanismos de desplazamiento en la ventana del marco. ✔ no: Este valor le dice al agente de usuario que nunca proporcione mecanismos de desplazamiento en la ventana del marco.
frameborder = 1 | 0	Este atributo proporciona información al navegador sobre el borde del marco. Valores posibles: ✔ 1: Este valor le dice al navegador que dibuje un separador entre este marco y todos los marcos adyacentes. Este es el valor por defecto. ✔ 0: Este valor le dice al navegador que no dibuje un separador entre este marco y todos los marcos adyacentes. Obsérvese que aún se puede dibujar un separador junto a este marco si así se especifica para otros marcos.
marginwidth = "píxeles"	Este atributo especifica la cantidad de espacio que debe dejarse entre los contenidos del marco en sus márgenes izquierdo y derecho. El valor debe ser mayor o igual que cero (píxeles). El valor por defecto depende del navegador.
marginheight = píxeles	Este atributo especifica la cantidad de espacio que debe dejarse entre los contenidos del marco en sus márgenes superior e inferior. El valor debe ser mayor o igual que cero (píxeles). El valor por defecto depende al agente de usuario.

El atributo src especifica el documento web que contendrá el marco. El siguiente ejemplo de documento HTML dispone de cuatro marcos:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Un documento con marcos</title>
    </head>
    <frameset cols="33%,33%,33%">
        <frameset rows="*,200">
            <frame src="contenidos_del_marco1.html">
            <frame src="contenidos_del_marco2.gif">
        </frameset>
        <frame src="contenidos_del_marco3.html">
        <frame src="contenidos_del_marco4.html">
    </frameset>
</html>
```

El navegador cargará cada documento web en una vista separada.

El siguiente ejemplo ilustra el uso de los atributos decorativos de frame. Especificamos que el marco 1 no permitirá barras de desplazamiento. El marco 2 dejará

espacio en blanco alrededor de sus contenidos (inicialmente, un fichero de imagen) y el marco no será redimensionable. No se dibujará ningún borde entre los marcos 3 y 4. Se dibujarán los bordes (por defecto) entre los marcos 1, 2 y 3.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Un documento con marcos</title>
    </head>
    <frameset cols="33%,33%,33%">
        <frameset rows="*,200">
            <frame src="contenidos_del_marco1.html"
scrolling="no">
            <frame src="contenidos_del_marco2.gif"
marginwidth="10" marginheight="15" noresize>
        </frameset>
        <frame src="contenidos_del_marco3.html"
frameborder="0">
        <frame src="contenidos_del_marco4.html"
frameborder="0">
    </frameset>
</HTML>
```

El atributo `target = "marco-destino"` especifica el nombre de un marco en el que debe abrirse un documento. Al asignar un nombre a un marco por medio del atributo name, los desarrolladores web pueden referirse a él como el "destino" de los vínculos definidos por otros elementos. Se pueden establecer el atributo `target` para los elementos que creen vínculos con el elemento `a`, para los mapas de imágenes `area`, y para los formularios `form`.

Además de asignar un nombre de marco, podemos asignar al atributo `target` alguno de los siguientes valores especiales

- ✔ `target="_blank"`, abre una nueva ventana del navegador y muestra el Hiperenlace en ella.
- ✔ `target="_self"`, se muestra en la subventana activa.
- ✔ `target="_parent"`, se muestra en el `frameset` definido anteriormente al actual. Si no hay ningún `frameset` anterior se muestra a pantalla completa suprimiendo todas las subventanas de la pantalla.
- ✔ `target="_top"`, suprime todas las subventanas de la pantalla y se muestra a pantalla completa.

Este ejemplo ilustra cómo es posible mediante la especificación de un destino la modificación dinámica de los contenidos de un marco. Primero definimos un grupo de marcos en el documento `frameset.html`, mostrado a continuación:

```
<!DOCTYPE html>
<html>
```

```
    <head>
        <title>Un documento con marcos</title>
    </head>
    <frameset rows="50%,50%">
        <frame name="fijo" src="inicial_fijo.html">
        <frame name="dinamico" src="inicial_dinamico.html">
    </frameset>
</html>
```

Después, en `inicial_dinamico.html`, hacemos un vínculo al marco llamado `"dinamico"`.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Un documento con vínculos con destinos
específicos</title>
    </head>
    <body>
        ...comienzo del documento...
        <p>Puede avanzar ahora a la <a href="diapo2.html"
target="dinamico" >diapositiva 2.</a>
        ...más documento...
        </p>
        <p>Lo está haciendo muy bien. Vaya ahora a la <a
href="diapo3.html" target="dinamico">diapositiva 3.</a></p>
    </body>
</html>
```

Si se activa cualquiera de los vínculos se abre un nuevo documento en el marco llamado `"dinamico"`, mientras que el otro marco, `"fijo"`, mantiene sus contenidos iniciales.

La definición de un grupo de marcos nunca cambia, pero los contenidos de uno de sus marcos sí pueden cambiar. Una vez que los contenidos de un marco cambian, la definición del grupo de marcos deja de reflejar el estado actual de sus marcos.

Los grupos de marcos hacen más difícil para los usuarios la navegación hacia adelante y hacia atrás por el historial del navegador.

21.2 Marcos anidados

Los grupos de marcos pueden anidarse hasta cualquier nivel. En este caso de lo que se trata es de hacer `frames` anidados o lo que es lo mismo, `frames` en el interior de un `frame`(s) ya definido con anterioridad, es decir una subdivisión en el interior de un `frame`. En el siguiente ejemplo, el `frameset` exterior divide el espacio disponible en tres columnas iguales. A continuación el `frameset` interior divide la segunda área en dos filas de alturas diferentes.

```
<!DOCTYPE html>
<html>
```

```
    <head>
        <title>Un documento con marcos</title>
    </head>
    <frameset cols="33%, 33%, 34%">
        ...contenidos del primer marco...
        <frameset rows="40%, 50%">
            ...contenidos del segundo marco, primera fila...
            ...contenidos del segundo marco, segunda fila...
        </frameset>
        ...contenidos del tercer marco...
    </frameset>
</html>
```

En este ejemplo hemos definido en primer lugar, un `frameset` que nos divide la pantalla del navegador en 2 columnas verticales de 100 pixeles la de la izquierda (en la pantalla), y el resto del área para el visualizador. Dentro de la misma, tal y como se ve en el ejemplo, hemos definido una subdivisión de la primera columna (la de la izquierda), en 2 filas (en el interior de la primera columna), que nos da un cuadro superior de 200 pixeles fijos y el inferior del resto de la pantalla del visualizador. Vemos de este modo lo que es un frame anidado, al mismo tiempo que ya hemos visto como subdividir una columna de las 2 que hemos definido.

```
<!DOCTYPE html>
<html>
    <head>
        <title> Título de la Página </title>
    </head>
    <frameset cols=“100,*”>
        <frameset rows=“200,*”>
        ...
        </frameset>
    </frameset>
</html>
```

Veamos otro ejemplo; supongamos que queremos hacer una subdivisión más o menos simétrica en 4 frames de la pantalla de nuestro visualizador, para lo que primero definimos las 2 columnas verticales:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Un documento con marcos</title>
    </head>
    <frameset cols=“50%,*”>
        ....
    </frameset>
</html>
```

Así hemos definido 2 columnas de la primera de ellas del 50% de la pantalla, y el resto que también es 50%, para la otra. Definamos ahora dentro de cada una de las columnas 2 filas que hagan que se nos muestre una pantalla dividida en 4 cuadros simétricos. Para ello

pondríamos:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Un documento con marcos</title>
    </head>
    <frameset cols="50%,*">
        <frameset rows="50%,*">
            ...
        </frameset>
    </frameset>
</html>
```

Dentro de la primera de las columnas, lo que nos dividiría ésta en 2 subdivisiones verticales. Ahora surge la duda de como dividir la segunda de nuestras columnas en otras 2 filas, pues de la misma forma que la anterior.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Un documento con marcos</title>
    </head>
    <frameset cols="50%,*">
        <frameset rows="50%,*">
            ...
        </frameset>
        <frameset rows="50%,*">
            ...
        </frameset>

    </frameset>
</html>
```

Debemos de considerar, que cada `frame` debe contener un documento HTML en su interior, con lo que a mayor número de divisiones, mayor número de documentos HTML a cargar, por lo que tendremos que considerar que llevará un mayor tiempo de carga de la "página total".

Otra forma de definir el ancho, alto o espaciado de las frames, es mediante múltiplos, tal y como se ve en `"valor*"`. El valor de este campo es opcional, un sólo asterisco implica una vista de "tamaño relativo", lo que se interpreta como una petición de darle a la vista todo el espacio que quede libre. Si hay varias vistas de tamaño relativo, el espacio libre se divide entre ellas. Si hay un valor delante del asterisco, la vista que lo tenga toma más espacio relativo, por ejemplo "2*,*" daría 2/3 del espacio a la primera vista y un tercio a la segunda.

21.3 Contenido alternativo

Se debería proporcionar contenido alternativo para aquellos navegadores que no soporten marcos o que estén configurados para no mostrar marcos.

Los navegadores que no soportan la característica de marcos, no mostrarán nada de

lo indicado con estos elementos. Es por ello que existe un elemento llamado `noframes`. Todo los indicado entre esta directiva será lo que muestren los navegadores sin capacidad para visualizar marcos. Los navegadores que soporten marcos obviaran el contenido incluido en `noframes`.

Anteriormente hemos mencionado que el elemento `frameset` era incompatible con el elemento `body`, y como veremos ahora no es del todo cierta. El único requisito que debe de cumplir para que no sea totalmente incompatible, es que la definición de marcos debe de ir antes de la definición del cuerpo del documento o `body`. Veamos un ejemplo:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Un documento con marcos</title>
    </head>
    <frameset cols="50%,*">
        <frame src="marco1.html">
        <frame src="marco2.html">
        <noframes>
            <body>
                ...
            </body>
        </noframes>
    </frameset>
</html>
```

Tal y como vemos en este ejemplo, se definen antes los cuadros de las ventanas (o frames) con el elemento `frameset` y el elemento `noframes` para aquellos navegadores que no soporten marcos, va incluida el elemento `body` entre las que indicaremos el contenido que queremos mostrar, dado que si el navegador accede a la segunda opción definida en `noframes`, indica que no soporta los frames, mientras que si sí los soporta, esta segunda opción de `noframes`, será obviada por el mismo.

Por ello podemos y la recomendación es hacer siempre uso del elemento `noframes`, para poder dar la opción al visitante, de que pese a que no tenga el navegador apropiado, poder ver la alternativa a la página con marcos que hemos confeccionado.

21.4 Marcos en línea

El elemento `iframe` permite insertar un marco dentro de un bloque de texto. Insertar un marco en línea dentro de una sección de texto es muy similar a insertar un objeto mediante un elemento `object`: ambos permiten insertar un documento HTML en medio de otro, ambos pueden alinearse con el texto circundante, etc.

La información a insertar en línea se designa mediante el atributo `src` de este elemento. Los contenidos del elemento `iframe`, por su parte, sólo deberían ser mostrados por los navegadores que no soporten marcos o que estén configurados para no mostrar marcos.

El elemento `iframe` dispone de los siguientes atributos:

Atributo	Descripción
longdesc=uri	Este atributo especifica un vínculo a una descripción larga del marco. Esta descripción debería servir como complemento de la descripción corta que proporciona el atributo title, y es particularmente útil para los navegadores no visuales.
name="nombre"	Este atributo asigna un nombre al marco actual. Este nombre puede utilizarse como el destino de vínculos subsiguientes.
width="longitud"	La anchura del marco en línea.
height="longitud"	La altura del marco en línea.

Para aquellos navegadores que soporten marcos, el siguiente ejemplo colocará un marco en línea rodeado por un borde en medio del texto.

```
<iframe src="marco_en_linea.html" width="400" height="500"
scrolling="auto" frameborder="1">
[Su navegador no soporta marcos o está actualmente configurado
para no mostrar marcos. Sin embargo, puede visitar
<a href="marco_en_linea.html">el documento relacionado.</a>]
</iframe>
```

Los marcos en línea no pueden ser redimensionados (y por lo tanto no tienen un atributo `noresize`). Obsérvese en el ejemplo que si el navegador no soporta marcos, el contenido del elemento `iframe` será lo que se presente por pantalla.

22 Publicación web

Publicar los contenidos en la web es relativamente fácil, lo difícil es decidir qué se publica, la estructura que tendrá y su desarrollo. El objetivo en si no es poner contenido en Internet, sino que sea lo suficientemente atractivo para que los usuarios visiten nuestro sitio. Un sitio web profesional improvisado está condenado al fracaso, mientras que un sitio web previamente diseñado con cuidado y esmero tendrá más posibilidades de éxito.

El desarrollo web consiste en un proceso cuyo objetivo es construir una aplicación web que cumpla unos requisitos establecidos de antemano. Un sitio web se implementa en forma de aplicación web o aplicación que se ejecuta en un servidor web y que cualquier usuario con acceso a Internet puede utilizar. Para construir la aplicación web se cuentan con una serie de herramientas y técnicas que permiten cumplir los objetivos del sitio partiendo de cero.

Antes de comenzar a escribir código del sitio que pretendemos construir es necesario establecer y definir claramente el trabajo a realizar. Esta definición se divide en dos fases:

- ✔ Análisis.- Consiste en estudiar lo que se pretende publicar, es decir, establecer los objetivos y contenidos que tendrá nuestro sitio web.
- ✔ Diseño.- Partiendo de la información recopilada en el apartado anterior, se define la estructura del sitio web que permita una implementación sencilla de la misma.

Posteriormente, el desarrollador podrá construir el sitio web utilizando la información del diseño, para lo cual utilizará una serie de herramientas habituales en programación web: Editores de texto, aplicaciones multimedia, navegadores de Internet, servidores Web, etc. Esta tarea de desarrollo web se divide en dos fases:

- ✔ Desarrollo.- En esta fase se escribe el código HTML, CSS, Javascript, etc. de los componentes web que forman el sitio.
- ✔ Prueba.- Se comprueba que el código escrito en nuestras páginas web aparecen tal y como deseamos. Estas pruebas deberán abarcar diferentes formatos de pantalla y navegadores para asegurarnos que la mayoría de los usuarios podrán ver de igual forma los contenidos independientemente del ordenador y/o navegador que utilicen.

Al finalizar las fases anterioers tendremos nuestro sitio construido y listo para publicarse. Todo el trabajo anterior lo habremos hecho en nuestro ordenador y ahora habrá que publicarlo en Internet almacenándolo en un servidor web. Esto implica las siguientes fases:

- ✔ Publicación.- Finalmente, una vez el sitio está definido y hecho pasaremos a su publicación en un servidor de Internet para ponerlo a disposición de los usuarios. Esta tarea no es trivial ya que dispondremos de muchas opciones en el mercado para publicar nuestro sitio web, cada una con sus ventajas e incovenientes. Habrá que hacer un estudio previo de los diferentes hostings para decidir cuál es la mejor opción.
- ✔ Evaluación.- No pensemos que lo que hemos hecho está perfecto. Las páginas web que forman nuestro sitio tendrá errores y habrá que corregirlos. También, es posible que los contenidos no sean los adecuados. Para saber si realmente lo publicado es lo que nos planteamos al principio hay que hacer una evaluación del mismo tomando como guia los requisitos que recabamos en la fase de análisis.
- ✔ Mantenimiento.- Aquí no acaba el trabajo. Un sitio web es un ente vivo. Está en constante evolución añadiendo, eliminando o alterando la información que contiene. Por tanto habrá un trabajo posterior de mantenimiento del sitio para adaptarlo a nuevas necesidades o ampliarlo. Este trabajo se prolongará hasta que el sitio deje de publicarse e implica todas las fases anteriores.

22.1 Construcción de un sitio web

Una página web la vemos en nuestro navegador, o cliente web, y parece una sola entidad, pero no es así, está compuesta por multitud de diferentes ficheros, como son las imágenes, los posibles vídeos y lo más importante, el código fuente que dice donde colocar cada texto, cada imagen o cada video y la forma que tendrán estos al ser colocados en la página. No es problema que las webs estén compuestas por tantos elementos, ya que rápidamente veremos que su organización es fácil y que no se nos van a perder o escapar ninguno.

Como hemos podido imaginar y a modo de resumen, para publicar en Internet necesitaremos construir unos documentos hipertexto, o hipermedia, con sus

correspondientes archivos de imagen o video y colocarlos en unos ordenadores que son servidores de páginas web. Sin embargo, hay que definir muy bien que vamos a construir y que elementos lo formarán. Para ello se hace imprescindible un trabajo previo que se detalla a continuación.

22.1.1 Análisis del sitio web

Para definir lo que pretendemos publicar y sus objetivos haremos lo siguiente:

22.1.1.1 Elegir un tema

Este es el primer paso y tiene una gran importancia ya que de las conclusiones que obtengamos determinará todo el desarrollo posterior. ¿Sobre qué vamos a publicar? Generalmente tendemos a seleccionar el tema en función de nuestras apetencias personales y no es un mal comienzo, pero deberemos tener presente que el destinatario final de nuestro sitio web es el usuario de Internet. Por tanto deberemos de seleccionar un tema que también sea de su interés.

22.1.1.2 Definición de los objetivos

Una vez elegido el tema haremos un análisis. El análisis consiste en la definición de los objetivos que se pretenden alcanzar con el sitio y la recopilación de los requisitos que tiene que cumplir, los cuales se utilizarán para realizar una evaluación del resultado final. Además se tendrá que especificar cuál es el perfil del usuario que utilizará el portal, ya que determinará el desarrollo del mismo. Finalmente, se concretará el contenido del portal que permitirá alcanzar dichos objetivos.

Esta es una etapa crucial pues en ella debemos definir:

1. Objetivos a cubrir con la construcción del sitio web.
2. Finalidad o propósito de nuestro sitio web.

22.1.1.3 Definición del destinatario

Las personas tenemos diferentes formas de percepción, comunicación y aprendizaje y cada uno nos desenvolvemos en un entorno social determinado. Luego para diseñar el sitío, debemos conocer e identificar perfectamente al público objetivo al que va dirigida la misma.

Hay sitios web que atraen a cientos o miles de visitantes diariamente y hacen su dinero principalmente por la venta de espacios publicitarios (banners). Sin embargo, hay una cosa que debemos tener clara: la cantidad de gente que visite nuestra web site no es tan importante como la calidad de la experiencia que esos visitantes obtengan. Es decir, no debemos pensar en la web como un medio tradicional de transmisión de mensajes publicitarios (medios de comunicación), sino como un medio personal, especializado, individual, interactivo y selectivo.

Deberemos encontrar nuestro público dentro de Internet, conseguir que su atención se dirija a nuestras páginas. Ahora la pregunta será: ¿Cómo podemos alcanzar este objetivo?

1. Nuestra web debe estar diseñada de forma atractiva.

2. Debemos contar con sistemas que faciliten la navegación entre las diferentes secciones.
3. Incorporando aplicaciones que ofrezcan un valor añadido al visitante.

22.1.1.4 Análisis del contenido

Necesitamos realizar una descripción esquemática de los contenidos para poder organizar internamente la información, determinar formatos y, si es necesario, utilizar bases de datos, como contenedores de información. En esta fase y en función de lo visto anteriormente, debemos determinar qué medios vamos a utilizar, por ejemplo:

- ✔ Imágenes fotográficas.
- ✔ Gráficos.
- ✔ Vídeos.
- ✔ Animaciones.
- ✔ Sonidos.
- ✔ Músicas.
- ✔ Textos.

22.1.1.5 Evaluación de la competencia

Un paso importante antes de desarrollar un sitio web, es evaluar las páginas de la competencia. Si conocemos su URL, debemos visitarla desde la óptica de un usuario potencial y posteriormente trasladar los resultados a nuestra propia web, con el fin de mejorar las carencias y deficiencias de la competencia. Para ello nos haremos las siguientes preguntas:

- ✓ ¿Es un diseño vendedor?
- ✓ ¿Tiene facilidad de navegación?
- ✓ Calidad de los contenidos.
- ✓ ¿Es una web intuitiva y fácil de utilizar?
- ✓ ¿Cuál es la velocidad de carga de las páginas?
- ✓ ¿Cuál es el nivel y calidad del diseño gráfico?

Además, podemos analizar el código fuente de las páginas y ver si está correctamente estructurado el código HTML. Otra cosa muy útil que podemos hacer es realizar una búsqueda en dos o más motores de búsqueda para determinar lo grande que es su presencia en la red.

22.1.2 Diseño

Un sitio web está formado por múltiples componentes: páginas web, hojas de estilos,

imágenes, video, audio, etc. Por tanto, habrá que decidir qué componentes tendrá y la función de cada uno de ellos.

En esta fase, definimos además el tipo de navegación, la interacción con el usuario y su comportamiento general. El resultado guiarán la programación del sitio, convirtiendo esto en una tarea trivial y permitiendo al programador centrarse en aspectos más técnicos del desarrollo, como la programación, las herramientas utilizadas, etc. Este resultado presentará la estructura del portal, es decir, las partes de que se componen y la relación que hay entre ellas.

La mejor forma de presentar esta estructura es mediante un mapa de navegación el cual muestra los componentes de nuestro sitio (páginas web, hojas de estilo, etc) y los vínculos que enlazan unos con otros.

22.1.3 Desarrollo

En esta fase creamos nuestros componentes web: páginas, hojas de estilo, etc. Además recopilaremos otros componentes como imágenes, audio, video, etc.

Para la creación de las páginas web utilizamos un editor de texto. Tenemos muchos para elegir y lo haremos en función de nuestras necesidades. En el mercado existen muchos editores de código HTML que incluyen funciones como autocompletado, coloración de código inteligente, etc.

Una vez que se ha iniciado la recopilación de material, es necesario organizarlo adecuadamente. Un sitio web que cuente con quince páginas, con varias imágenes y textos cada una, puede generar una gran cantidad de archivos. Si cada archivo está en una carpeta distinta y sigue un tipo de denominación diferente, el desarrollador puede perder mucho tiempo localizándolos. Por ello, es preferible seguir algunas convenciones a la hora de almacenar y nombrar los distintos materiales que formarán el sitio web.

22.1.3.1 Nomenclatura de archivos

A la hora de recopilar los elementos necesarios o incluso, cuando ya se empieza a desarrollar las páginas web, es importante prestar atención a los nombres que se da a los archivos. Esto evitará problemas de desorganización, errores en la página porque no se encuentra un archivo, etc. Por ello, es conveniente seguir las siguientes indicaciones:

- ✔ Evitar el uso de caracteres especiales.- Hay ciertos caracteres (acentos, cedillas, letra eñe, espacios en blanco, etc.) que pueden ocasionar problemas de interpretación por parte del navegador web. Por ello, es preferible evitar su uso a la hora de nombrar archivos, siendo mejor llamar a una carpeta *imagenes_en_color* que *imágenes en color*, con acento y espacios en blanco.
- ✔ Utilizar las minúsculas.- Algunos sistemas operativos como Linux, diferencian entre los nombres escritos en mayúsculas y minúsculas, por lo que la carpeta *Imágenes* no será la misma que *imágenes*. Para evitar problemas, la mayoría de desarrolladores prefieren nombrar todos los archivos y carpetas con minúsculas, reservando el uso de las mayúsculas para las instrucciones de código.

- ✔ La página index.- Cuando se inserta una URL con directorios en la barra de direcciones pero no se especifica el archivo (www.pagina. com/carpeta1), la mayoría de servidores buscarán en esa dirección una página llamada *index* (que llevará la extensión acorde al lenguaje en la que está escrita: index.html, index.php, etc.). Por ello es conveniente llamar así a la página inicial del sitio web, evitando que el servidor no sepa qué página cargar.

22.1.3.2 Organización del almacenamiento

Uno de los mayores problemas que se pueden dar a la hora de desarrollar páginas web es que no se carguen los elementos de las mismas. En muchos casos, esto se debe a que dicho elemento (por ejemplo, una imagen) «ha cambiado» de lugar o no se encuentra donde dice el documento web. Para evitarlo, el desarrollador debe ser consistente y cuidadoso a la hora de organizar los elementos de la página web, facilitando así que las referencias a la URL o la dirección de un elemento sean siempre correctas. Un buen método de organización es crear una carpeta raíz en la cual estarán las páginas web que se vayan desarrollando y diferentes carpetas por cada tipo de elemento que lleve cada página: imágenes, vídeos, sonidos, animaciones, etc. Si el número de elementos fuese muy grande, probablemente sería necesario crear subcarpetas por categorías de elementos (imagenes/fotos, imagenes/dibujos, etc.).

Por otra parte, habrá que prestar atención a cómo se indica al servidor la ruta *o path* que debe seguir para encontrar un elemento de la página web:

- ✔ Direccionamiento absoluto.- Consiste en escribir la dirección completa del elemento. Por ejemplo: *http://nombre_dominio/mi_sitio_web/imagenes/imagen1.jpg*. El direccionamiento absoluto puede provocar problemas si no se es cuidadoso. Las direcciones que se empleen mientras se desarrolla la página generalmente harán referencia al ordenador personal del desarrollador *(mis documentos / Imagenes / archivo).* Al ser subidas al servidor, las páginas seguirán (si no han sido modificadas) haciendo referencia a dichas carpetas o directorios y, por tanto, no podrán ser encontradas por los servidores de las páginas.

- ✔ Direccionamiento relativo.- Se escribe la dirección relativa del elemento respecto a la página que lo llama. Por esta razón siempre es recomendable añadir direccionamiento relativo en las referencias a los elementos de una página web (salvo que sean enlaces a otras páginas que no pertenecen al mismo sitio web), ya que así, si las páginas se cambiasen de servidor, no habría que efectuar cambios para que todo siguiese funcionando.

22.1.4 Pruebas

Cuanto más tiempo dediquemos a probar nuestro sitio web menos errores tendrá. Las pruebas de las páginas web deberán ser lo más extensas posibles y que abarquen a todos los elementos de nuestro sitio web. Tenemos que probarlo todo: contenido y formato, enlaces, carga de imágenes, etc.

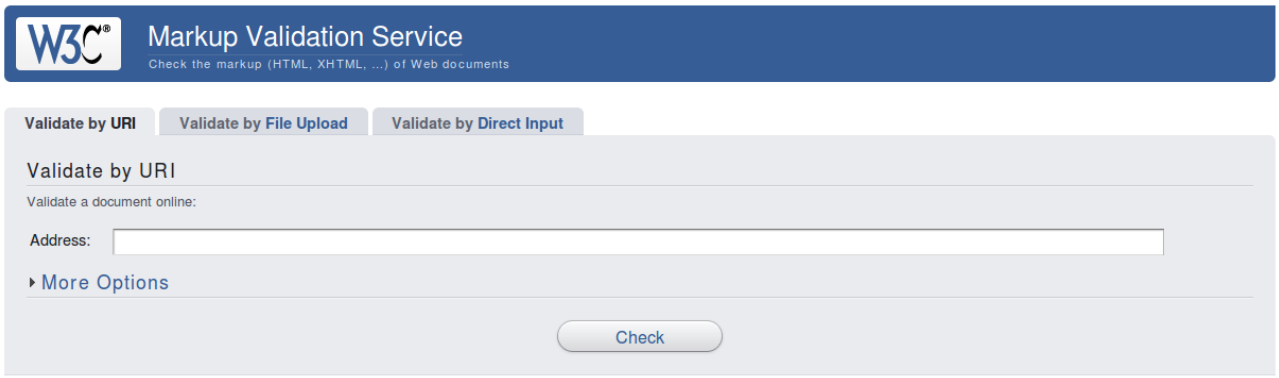
Además, deberemos probar nuestro sitio web en varias plataformas, si es posible, y

con varios navegadores. Un sitio web puede variar su aspecto si lo probamos en ordenador con Windows y un navegador Internet Explorar a otro con Linux y un navegador Chromium.

22.1.4.1 Validar un sitio web

Es de sobra conocida que nuestras páginas web pueden contener errores que el navegador *"arregla"* al presentarla en pantalla. Sin embargo, resulta conveniente que nuestras páginas estén libres de errores para aumentar la probabilidad de que puedan verse correctamente en cualquier navegador.

W3C es un consorcio internacional que produce recomendaciones para el servicio www en Internet. Está dirigido por Tim Berners-Lee, el creador del servicio www. Su sede principal se encuentra en el MIT (Massachussets Institute Technological), aunque dispone de subsedes en Francia y Japón.



El W3C ofrece a los desarrolladores web un servicio de validación para verificar que sus páginas web se han escrito sin errores. Podemos utilizar este servicio para comprobar si nuestras páginas web están correctamente escritas. Estas páginas que queremos validar pueden estar ya publicadas en la red o no, en cuyo caso estarían en nuestro PC y las tendríamos que subir al validador para su comprobación. También, podemos incluir código HTML en el propio validador para su comprobación.

Después de crear un página web, utilizaremos el servicio de validación http://validator.w3.org para comprobar si el código HTML cumple con el estádar del W3C para el tipo de documento que hemos hecho (HTML, XHTML, HTML5). Para utilizar el servicio de validación, seguiremos los siguientes pasos:

- ✔ Iniciamos un navegador e intruducimos la dirección http://validator.w3.org
- ✔ Si queremos validar un página que aún no hayamos publicado en el servidor web, hacemos clic en *Examinar* dentro de *Validate by file upload*. Esto nos permite cargar un archivo de nuestra máquina local.
- ✔ Hacemos clic en Check

Una vez hayamos completado estos pasos el explorador web nos enviará la página web que hayamos introducido o seleccionado al sitio web del w3c. El programa de validación del sitio examinará el documento y enviará al explorador web una página de

resultados.

Debemos de tener en cuenta que el documento HTML o XHTML es válido (es decir, cumple el estándar) sólo cuando no hay errores. Por lo tanto, si vemos errores, los corregimos en el documento web y repetimos el procedimiento de validación. El programa de validación no sólo ofrece una lista de errores, también ofrece sugerencias de corrección que tiene que realizar para que el código cumpla el estándar.

Cuando tengamos una web sin errores podemos incluir un logotipo del W3C el cual da fe de que nuestra página ha sido validada por el W3C.

22.2 Dominio y hospedaje

22.2.1 Selección de un servidor

Para publicar una página en Internet debemos colocarla en un servidor. Los servidores son ordenadores conectados permanentemente a la Red que envían las páginas cuando los exploradores la piden. Antes de nada vamos a diferenciar entre dos formas de alojar una web, en un servidor gratuito o en un servidor de pago. Los servidores de pago suelen tener mayores prestaciones y recursos disponibles que los gratuitos.

22.2.1.1 Cómo elegir un alojamiento gratuito

Los servidores de hosting gratuitos está en desuso. Todos los sitios más importantes en Internet que ofrecían espacio gratuito han ido desapareciendo gradualmente. Ahora lo que se ha popularizado son los sitios gratuitos para hacer blogs, donde ya ofrecen todas las herramientas listas para usar por cualquier persona sin experiencia con las que administrar webs de las que llaman 2.0.

El abaratamiento de los servicios de alojamiento de webs también ha sido un factor importante para que, cualquiera que quiera hacer una web personal o profesional, pueda contar con su propio dominio.

Por suerte, existe una gran variedad de sitios que ofrecen a sus visitantes la oportunidad de publicar sus paginas personales o de empresa de manera gratuita y muchas de estas opciones tienen muy buena calidad.

En un principio, el mejor espacio para publicar una web que podemos elegir puede ser el proporcionado por nuestro proveedor de acceso a Internet. Muchas veces nuestro proveedor, a la vez que ofrece la conexión a Internet, también ofrece un espacio para publicar, por el mismo precio que el acceso. Esta opción tiene sus ventajas:

- ✔ Suelen estar libres de publicidad
- ✔ En muchos casos, existen menos limitaciones por el tipo de contenidos
- ✔ Casi siempre tienen buena velocidad.

El único inconveniente a destacar, aparte de los problemas puntuales que tenga cada proveedor de acceso, es que abandonemos algún día el proveedor y nos retiren las páginas publicadas.

En el caso de que nuestro proveedor no ofrezca servicio de publicación de webs o no nos guste el servicio que ofrece, tenemos la opción de publicar las páginas en alguna web que ofrezca espacio gratuito. Este tipo de servicio es muy popular, exisiten muchas ofertas en Internet, algunas de ellas mejores de las proporcionadas por los proveedores de acceso, incluso por los proveedores de alojamiento. En este caso deberemos utilizar los siguientes criterios para elegir un alojamiento gratuíto:

- ✔ Recursos permitidos.- Existen muchos sitios donde publicar una página, se diferencian fácilmente por los recursos que ponen a nuestra disposición para construirla. Cuando hablamos de recursos nos referimos a la cantidad de herramientas y programas que nos dejan utilizar en nuestra web. Ejemplos de recursos son: estadísticas, contadores, direcciones web cortas, mail gratuito, CGI, bases de datos... Los recursos se hacen necesarios cuando construimos una web avanzada, con secciones dinámicas como un buscador, un portal, o en general, una página que se alimenta de cualquier base de datos o programa para mostrar su contenido. En la gama de recursos que ofrecen estos sitios gratuitos es muy variada. Algunos sitios casi no ofrecen ningún recurso, simplemente podemos publicar la página, y en otros sitios ofrecen más servicios que en algunos proveedores de alojamiento de pago.

- ✔ Velocidad del servidor.- Para elegir un servicio suficientemente veloz tenemos la oportunidad de probar la rapidez con que bajan las páginas de alguna web que esté alojada en ese servidor. Es decir, visitamos una web alojada en ese servidor y vemos lo rápido que llega, comparando nosotros mismos. En un último caso podemos registrarnos nosotros mismos y probar la velocidad del espacio con nuestros propios contenidos. No pasa nada por registrarse en un sitio, probar cómo funciona y si no nos gusta simplemente nos vamos a otro, es gratis.

- ✔ Situación geográfica.- Un dato a considerar es que, dependiendo de dónde está situado físicamente el servidor que alojará nuestras páginas, la velocidad será distinta para nosotros. Es decir, un español ve más rápido un servidor español, pero un americano navegará con más velocidad si el servidor está en América. Esto, se lleva al límite cuando es nuestro proveedor de acceso el que nos ofrece el espacio, en este caso, seremos los que navegen más rápido por esa web, por estar tan proximos los dos sitios, tu acceso a Internet y el servidor de tu página.

- ✔ Publicidad.- También se pueden clasificar los sitios para publicar en Internet por la publicidad que insertan. Los sitios de espacio gratuito suelen subvencionarse mediante la inserción de anuncios en las páginas de los usuarios del servicio. Los tipos de publidad que podemos encontrar son los siguientes:

  - ✗ Publicidad en un frame inferior.- Es tal vez la más agresiva de las publicidades que pueden hacer, pues el banner permanece siempre visible en la página y es un poco feo para las páginas.

  - ✗ Publicidad dentro de la página, utilizando el banner superior. Es el método de inserción de banners más habitual. Como desventaja tiene que el banner se exibe en la propia página, modificando nuestra web.

- ✗ Publicidad en una ventana aparte. Esta publicidad aparece en una ventanita aparte del explorador. Como ventaja está que no utiliza el espacio de nuestra página. Como desventaja contamos que estas ventanas son muy molestas para muchos usuarios.
- ✗ Publicidad en la propia página, en un grafiquito flotando en la propia página. Es muy interesante, pues ocupa muy poco espacio comparado con otros mecanismos y no molesta en exceso.
- ✗ Sin publicidad, sin duda el mejor de los casos. Lo que no nos asegura que el sitio no ponga publicidad algún día y deje de ser tan interesante. Actualmente, muy pocos sitios ofrecen espacio gratuito sin colocar algún tipo de publicidad.

✔ Subida de archivos.- Existen diferentes maneras de subir los archivos. Las más habituales son utilizar FTP o un formulario web. En la descripción de características del alojamiento suelen indicar acceso por FTP cuando esta permitido realizar este tipo de actualizaciones. Para los principiantes será más cómodo subir las páginas por medio de un formulario de una web, pero para las personas experimentadas, a la larga, les será más sencillo y tendrá menos trabajo el realizar las actulizaciones del sitio via FTP. Como ventaja de subir archivos con un formulario tenemos que no será necesario disponer de un programa de FTP ni tampoco saber utilizarlo. Como ventaja del FTP destacamos la rapidez de subida de archivos, ya que podemos seleccionar varios archivos de una sola vez y no tenemos que esperar a que se cargue la página web para que tengamos acceso al formulario, subir un directorio en una sola acción y otra mejora importante, que consiste en que podremos utilizar el programa con el que diseñamos las páginas web (tipo Dreamweaver o Homesite) para acceder al servidor y actualizar los contenidos automáticamente.

22.2.1.2 Cómo elegir un alojamiento de pago

Si el alojamiento gratuito no satisface nuestras expectativas, tendremos que alojar nuestro sitio web en un servidor de pago. La primera diferencia con respecto al alojamiento gratuito es que dispondremos de un dominio propio el cual le dará más personalidad al sitio, se asociará mejor con el nombre de nuestro sitio, a la vez que se hace más accesible la página. Con el reducido precio que tienen actualmente los dominios sería muy indicado adquirir uno para nuestro sitio. En muchos casos al contratar un espacio para alojar la web nos van a ofrecer directamente nuestro propio dominio, si no es así, lo podemos solicitar expresamente. Lo más habitual es que en el proveedor donde alojemos la página se encargue también del registro del dominio. Para seleccionar uno seguiremos los siguientes criterios:

✔ Definir los recursos.- Para encontrar un alojamiento para nuestra página primero nos tenemos que plantear los objetivos que queremos cumplir y los recursos que necesitaremos. Según sea el proyecto que vamos a publicar necesitaremos más o menos personalidad propia o determinados recursos como acceso a bases de datos , estadísticas, etc. Aparte del dominio existen otros recursos que podemos necesitar. Deberemos definir los recursos a utilizar y buscar un proveedor que los acepte todos. Si estamos proyectando una página sencilla, probablemente no te preocupará el

número de recursos avanzados que posea tu servidor, pero siempre será interesante evaluar otros recursos más básicos como:

- ✗ Programación en lenguajes específicos .
- ✗ Acceso a bases de datos
- ✗ Administración de las DNS...
- ✗ El número de direcciones de correo que te van a proporcionar
- ✗ Los CGIs que tenga instalados, como contadores, envío de formularios por correo electrónico, etc.
- ✗ Estadísticas...

✔ Velocidad.- Es una pena que no exista un método realmente fiable para controlar uno de los aspectos que más preocupa a todos los que buscamos un alojamiento. Para conocer la calidad el servicio que nos va a ofrecer el proveedor deberíamos preguntarles a ellos sobre sus líneas y la capacidad que tienen de transferencia con Internet. Pero no solo eso sino también preguntarles por la saturación que tienen sus líneas. Pueden ser muy potentes pero estar utilizadas intensamente. Como los proveedores no van a revelar estos detalles tal como a nosotros nos gustaría, lo más probable es que solo nos quede la posibilidad de probar la velocidad por la práctica. Si navegamos por el sitio de la empresa que estamos evaluando y comprobamos la cantidad de información (Kb) que recibimos por segundo podremos saber cómo va de rápido la página. Otro factor que también condiciona la velocidad es la cercanía del servidor. Ten en cuenta que si estamos en España, un servidor situado en nuestro país será más rápido para nosotros, en la misma condición de líneas, que uno que esté en Estados Unidos. Cuanto más lejos tenga que viajar la información, tarda más. Pero un detalle, si un usuario de Colombia visita una página que está en un servidor español, para él tu servidor será más lento que uno de su país.

✔ Trasnferencia.- Otro aspecto que tenemos que evaluar a la hora de contratar un espacio para nuestra web es la cantidad de megas de transferencia mensual que el proveedor permite realizar desde nuestro dominio hacia fuera. Es decir, las páginas web que manda el servidor tienen un peso en Kbytes y el proveedor los va contando, cuando pasamos el límite que le han asignando al dominio nos cobran el exceso según un precio. Así pues, a la hora de contratar un dominio merece la pena enterarse sobre este aspecto y evaluar también las condiciones que nos ofrecen. Si encontramos un servidor que ofrece transferencia ilimitada no nos tiene que decir tampoco mucho de él, el control de la transferencia les permite ajustar su servicio a unos niveles de calidad superiores. Además, en la mayoría de los casos no nos tendrá que preocupar el superar las tasas de transferencia pues las megas que transfiere una web normal nunca superará los niveles propuestos, a no ser que se trate de un portal o una página muy visitada.

✔ Precio.- El último factor a considerar, y el más fácil de comparar, es el precio. Lo importante es comparar la calidad del servicio y el precio del mismo. Habrá realizar esta comparación sabiendo los datos que tenemos que evaluar.

- ✔ Soporte al cliente.- Este aspecto es muy importante pero desgraciadamente el más difícil de conocer. Las empresas de hosting disponen de un servicio de soporte al cliente para resolver sus dudas y solucionar problemas. Si el soporte al cliente no es bueno, puede ser una pesadilla intentar solucionar algún problema, mientras que un buen soporte al cliente siempre tendrá disponibilidad para resolver problemas lo antes posible. Cuando estemos seleccionando un hosting es conveniente consultar en Internet como es el soporte al cliente de los posibles candidatos para que no haya sorpresas después de contratarlo.

22.2.1.3 Búsqueda de los mejores hostings

Existen en Internet portales que nos ofrecen información y rankings sobre las mejores empresas de alojamiento web en la red. Por ejemplo, los siguiente sitios nos ofrecen información sobre las mejores empresas de hosting:

- ✔ www.top10bestwebsitehosting.com
- ✔ www.hosting-review.com
- ✔ mejorhosting.org

Sin embargo, resulta conveniente realizar una búsqueda en Internet sobre los ranking de empresas de alojamiento ya que estas se actualizan constantemente.

22.2.1.4 Alojamiento en un servidor propio

Esta opción solamente está al alcance de personas con conocimientos en informática necesarios para ello. Consiste en tener en casa un servidor web donde habría que instalar:

1. Una conexión a Internet mediante una línea ADSL o por cable. Deberemos tener cuidado a la hora de elegir una porque necesitaremos tener un buen ancho de banda en subida, en lugar de en bajada, ya que nuestro PC va a actuar como servidor, no como cliente. No es necesario tener una dirección IP asignada estáticamente.
2. Un PC que haría de servidor permanentemente conectado a Internet y encendido las 24 horas del día los 365 días del año para ofrecer total disponibilidad a los usuarios. Si nuestro sitio no va a tener inicialmente muchas visitas podemos utilizar un PC doméstico. Sin embargo, debemos tener presente que al estar encendido todo el tiempo consumirá mucho y también generará ruido y calor. Para reducir su consumo y eliminar ruido y calor podemos adquirir un MiniPC sin ventilador por un bajo coste y con características técnicas avanzadas.
3. Instalar y configurar un sistema operativo en el PC. Puede ser cualquiera: Windows, Linux, Mac OSC, etc
4. Instalar y configurar un servicio web. Existen varias opciones en el mercado y dependiendo del sistema operativo elegido nos convendrá uno u otro. Si tenemos Windows podremos instalar Internet Information Services de Microsoft o también Apache Web Server del proyecto Apache. Si nuestro servidor tiene instalado Linux, entonces la mejor opción es Apache.

5. Instalar y configurar los servicios añadidos de nuestro servidor web, como base de datos, transferencia de ficheros por FTP, acceso remoto para configuraciones, etc.

6. Contratar un dominio y configurar el servicio DNS para que el nombre elegido apunte a nuestro servidor web. Lo más seguro es que cuando contratemos nuestro nombre de dominio la empresa donde lo hagamos nos gestione el DNS mediante DynDNS o algún otro método.

No es una tarea fácil ni trivial montar un servidor web en casa y conlleva mucho trabajo, incluso después de su puesta en funcionamiento, pero resulta una solución mucho más flexible a la hora de instalar y configurar servicios añadidos a nuestros sitios web.

22.3 Normativa

¿Es posible incluir desde el punto de vista legal cualquier elemento en una página web? La respuesta es, obviamente, no. Aunque Internet tiene un espíritu libre y no está totalmente regulado, hay muchas leyes que son trasladables del «mundo real» al virtual. Estas leyes hacen especial referencia a la propiedad intelectual de los elementos y a la posibilidad de publicar datos personales o sensibles.

Toda creación o invento tiene un autor que es el propietario intelectual de la obra. Poco importa que sea una obra literaria sin ninguna calidad artística, un descubrimiento científico sin resonancia alguna o un invento inservible: sigue teniendo un autor y éste es acreedor a unos derechos, los de la propiedad intelectual, que protegen a su creación de la modificación, reproducción o comercialización por parte de terceros sin su permiso. Dichos derechos existen desde el mismo momento de la creación de la obra: es decir, no es necesario inscribirla en un registro, firmarla, poner símbolos de copyright o patente, etc. (todo esto en realidad sólo tiene importancia a efectos de reclamaciones o para publicitar la autoría de la obra).

Todo ello implica que cualquier elemento de Internet, desde un diseño web hasta una imagen, pasando por cualquier texto, tiene un autor y, por tanto, su uso está sometido a ciertas restricciones. No es posible descargarse una imagen y utilizarla en nuestra propia página web, ni cortar y pegar un texto. Es necesario pedir permiso (preferiblemente por escrito) o asegurarse de que éste ha sido concedido de antemano mediante una licencia Creative Commons (http://creativecommons.org), GNU GPL (Licencia pública general de GNU: www.gnu.org) o similares.

Por supuesto, el espíritu libre de Internet ha facilitado iniciativas para la libre distribución de imágenes, textos, piezas musicales, etc., y existen numerosas páginas web que ponen a disposición de los desarrolladores multitud de recursos gratuitos, especialmente imágenes.

Otra forma de obtener recursos sin necesidad de pedir permiso por escrito es comprando creaciones libres de royalties (royalty free), es decir, liberadas del pago de canones por su utilización. Una vez que se compra el CD con las imágenes, sonidos, canciones, vídeos, etc., se pueden utilizar todas las veces que se quiera y generalmente, como se quiera (los autores pueden poner limitaciones de uso, por ejemplo, para publicidad

de productos como el tabaco, pero suelen ser escasas).

Por otra parte, no sólo lo que puede ser considerado como una creación está protegido por la ley: también lo están los datos personales o los considerados como «sensibles» (ideología, sexualidad, afiliación política o sindical, etc.). Estos últimos no pueden estar siquiera recogidos o almacenados en ficheros privados. Esta protección afecta también a Internet ya que el SPAM o los pop-up publicitarios se consideran comunicaciones.

22.3.1 Prácticas poco éticas

Existen una serie de prácticas específicas de la World Wide Web que, aunque no están totalmente reguladas y, por tanto, no pue#den ser calificadas de ilegales, pueden ser consideradas como poco éticas. Un claro ejemplo es el llamado hotlinking (enlace caliente), consistente en incluir en una página web imágenes o vídeos que en realidad están alojados en otro servidor. Esto libera recursos propios (ancho de banda), «robándoselos» a un tercero, que es quien deberá pagar o sufrir el exceso de tráfico generado por la visualización o descarga de dichos elementos.

22.3.2 Creative Commons

Creative Commons (CC) es una organización no gubernamental sin ánimo de lucro que desarrolla planes para ayudar a reducir las barreras legales de la creatividad, por medio de nueva legislación y nuevas tecnologías. Fue fundada por Lawrence Lessig, profesor de derecho en la Universidad de Stanford y especialista en ciberderecho, que la presidió hasta marzo de 2008.

22.3.2.1 Licencias Creative Commons

Las licencias Creative Commons o CC están inspiradas en la licencia GPL (*General Public License*) de la *Free Software Foundation*. No son, sin embargo, un tipo de licenciamiento de software. La idea principal es posibilitar un modelo legal ayudado por herramientas informáticas, para así facilitar la distribución y el uso de contenidos.

Existe una serie de licencias Creative Commons, cada una con diferentes configuraciones o principios, como el derecho del autor original a dar libertad para citar su obra, reproducirla, crear obras derivadas, ofrecerla públicamente y con diferentes restricciones, como no permitir el uso comercial o respetar la autoría original.

Una de las licencias que ofrecía Creative Commons es la que llevaba por nombre "*Developing Nations*" (Naciones en Desarrollo), la cual permitía que los derechos de autor y regalías por el uso de las obras se cobraran sólo en los países desarrollados del primer mundo, mientras que se ofrecían de forma abierta en los países en vías de desarrollo. Esta licencia ha sido retirada por problemas comerciales.

Aunque originalmente fueron redactadas en inglés, las licencias han sido adaptadas a varias legislaciones en otros países del mundo. Entre otros idiomas, han sido traducidas al español, al portugués, al gallego, al euskera y al catalán a través del proyecto Creative Commons International. Existen varios países de habla hispana que están involucrados en este proceso: España, Chile, Guatemala, Argentina, México, Perú, Colombia, Puerto Rico y

Ecuador ya tienen las licencias traducidas y en funcionamiento, en tanto que Venezuela se encuentra en proceso de traducción e implementación de las mismas. Brasil también tiene las licencias traducidas y adaptadas a su legislación.

Poner vuestras obras bajo una licencia Creative Commons no significa que no tengan copyright. Este tipo de licencias ofrecen algunos derechos a terceras personas bajo ciertas condiciones. ¿Qué condiciones? Esta web os ofrece escoger o unir las condiciones de la siguiente lista. Hay un total de seis licencias Creative Commons para escoger:

Licencia	Descripción
Reconocimiento (Attribution)	En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.
No Comercial (Non commercial)	La explotación de la obra queda limitada a usos no comerciales.
Sin obras derivadas (No Derivate Works)	La autorización para explotar la obra no incluye la transformación para crear una obra derivada.
Compartir Igual (Share alike)	La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Con estas cuatro condiciones combinadas se pueden generar las seis licencias que se pueden escoger:

Licencia	Denominación	Descripción
	Reconocimiento (By)	Se permite cualquier explotación de la obra, incluyendo una finalidad comercial, así como la creación de obras derivadas, la distribución de las cuales también está permitida sin ninguna restricción.
	Reconocimiento - NoComercial (by-nc)	Se permite la generación de obras derivadas siempre que no se haga un uso comercial. Tampoco se puede utilizar la obra original con finalidades comerciales.
	Reconocimiento - NoComercial - CompartirIgual (by-nc-sa)	No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.
	Reconocimiento - NoComercial - SinObraDerivada (by-nc-nd)	No se permite un uso comercial de la obra original ni la generación de obras derivadas.

	Reconocimiento - CompartirIgual (by-sa):	Se permite el uso comercial de la obra y de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.
	Reconocimiento - SinObraDerivada (by-nd)	Se permite el uso comercial de la obra pero no la generación de obras derivadas.

22.3.2.2 Publicar una obra bajo licencia Creative Commons

En la web de Creative Commons podemos conseguir una licencia para las obras que hemos creado y publicarlas bajo las condiciones que expresa la licencia escogida. Primero tendremos que leer atentamente las condiciones de cada licencia y elegir la más adecuada. Cuando hayas hecho la elección dispondremos de tres formas para expresarla en nuestra web:

- ✔ Commons Deed: Es un resumen fácilmente comprensible del texto legal con los iconos relevantes.
- ✔ Legal Code: El código legal completo en el que se basa la licencia que has escogido.
- ✔ Digital Code: El código digital, que puede leer la máquina y que sirve para que los motores de búsqueda y otras aplicaciones identifiquen tu trabajo y sus condiciones de uso.

Una vez escogida la forma de expresar la licencia en nuestra obra habrá que incluir el botón *Creative Commons "Algunos derechos reservados"* en la web, cerca de vuestra obra. Este botón enlaza con el Commons Deed, de forma que todos puedan estar informados de les condiciones de la licencia. Si encuentras que tu licencia ha sido violada, entonces tendrás las bases para poder defender tus derechos.

Para generar la forma de expresión de nuestra licencia podemos utilizar un formulario que en la web de Creative Commons pone a disposición de todos los usuarios que quieran liberar sus obras bajo alguna de las licencias antes descritas. Para ello podemos seguir los siguientes pasos:

1. Entrar en www.creativecommons.org y seleccionar la opción de menú *Licences – Choose a licence*.

2. Rellenar el formulario indicando los derechos que concedemos a la obra que hemos creado

3. Rellenaremos también el formulario con los datos del autor, ya que es muy posible que escojamos una licencia con reconocimiento de la autoría.

4. Una vez completados los datos disponemos en la parte derecha del código HTML que nos incluirá la licencia en nuestra página web. Este fragmento de código HTML debe ser insertado en nuestra página web.

Este es el aspecto que tomaría nuestra web con la inclusión de la licencia Creative Commons

Como podemos observar, en la parte inferior se ha incluido la licencia Creative Commons con enlaces al autor de la obra y al detalle de la licencia.

23 Bibliografía

EGUILUZ, J, *Libros web - CSS Avanzado,* [Acceso 25 agosto 2013]. Disponible en <http://librosweb.es/css_avanzado/>

W3C, *HTML 5.1 Nightly – Editor's Draft 30 August 2013,* [Acceso 30 de agosto 2013]. Disponible en <http://www.w3.org/html/wg/drafts/html/master/>

W3C, *Cascading Style Sheet Level 2 Revision (CSS 2.1) Specification,* [Acceso 20 de agosto 2013]. Disponible en <http://www.w3.org/TR/2011/REC-CSS2-20110607/>

GAUCHAT, J.D., *El gran libro de HTML5, CSS3 y Javascript*. 2012 Marcombo Ediciones Técnicas, ISBN 978-84-267-1770-2.

W3SCHOOLS, *HTML5* [Acceso 22 de agosto 2013]. Disponible en <http://www.w3schools.com/html/html5_intro.asp>

W3SCHOOLS, *CSS3 Tutorial* [Acceso 24 de agosto 2013]. Disponible en <http://www.w3schools.com/css3/>

ALVAREZ, M.A., *Degradados con CSS3*, Abril 2011 [Acceso 19 de agosto 2013]. Disponible en <http://www.desarrolloweb.com/articulos/degradado-css-3.html> y <http://www.desarrolloweb.com/articulos/degradados-radiales-css3.html>

GONZALEZ, L., *Imagen digital: conceptos básicos*. [Acceso 21 de agosto 2013]. Disponible en <http://platea.pntic.mec.es/~lgonzale/tic/imagen/conceptos.html>

BUCHANAN, B., *Estilos de las tablas*, Septiembre 2008 [Acceso 26 de agosto 2013]. Disponible en <http://mosaic.uoc.edu/ac/le/es/m6/ud7/index.html>

POZO, J.R., *HTML con clase*, Mayo 2005 [Acceso 28 de agosto 2013]. Disponible en <http://html.conclase.net>

JMIUR, *Transformaciones con CSS*, Marzo 2011 [Acceso 29 de agosto 2013]. Disponible en <http://vagabundia.blogspot.com/2011/03/transformaciones-con-css.html>

EPUÑAN, J., *Animaciones y transformaciones con CSS3*. Enero 2011 [Acceso 29 de agosto 2013]. Disponible en <http://www.csslab.cl/2011/01/31/animaciones-y-transformaciones-con-css3/>

W3C, *CSS Transitions - W3C Working Draft 3 April 2012*. Abril 2012 [Acceso 29 de agosto 2013]. Disponible en <http://www.w3.org/TR/2012/WD-css3-transitions-20120403/>