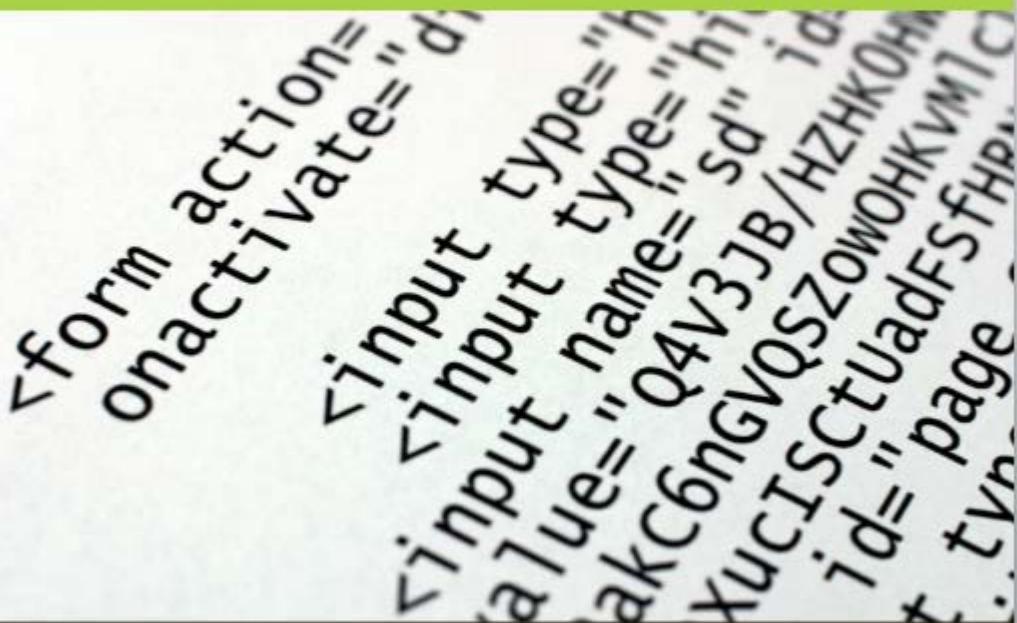


Elaboración de documentos web mediante lenguajes de marcas

UF1841

Xabier Ganzábal García



A large, faint watermark of a document form with various input fields and a submit button.

IFCD0210 Desarrollo de aplicaciones con tecnologías web

MF0491_3>UF1841

Paraninfo

Elaboración de documentos web mediante lenguajes de marcas

UF1841

Xabier Ganzábal García

Paraninfo

Índice

1. Diseño Web	1
1.1. Principios de diseño web	3
1.1.1. Diseño orientado al usuario	3
1.1.2. Diseño orientado a objetivos	3
1.1.3. Diseño orientado a la implementación	4
1.2. El proceso de diseño web	4
1.2.1. Estructura de un sitio web y navegabilidad	4
1.2.2. Estructura y composición de páginas	4
1.2.3. Compatibilidad con navegadores	5
1.2.4. Diferencias entre diseño orientado a presentación e impresión	5
2. Lenguajes de marcado generales	7
2.1. Origen de los lenguajes de marcado generales: SGML y XML	9
2.2. Características generales de los lenguajes de marcado	9
2.3. Estructura general de un documento con lenguaje de marcado	9
2.3.1. Metadatos e instrucciones de proceso	10
2.3.2. Codificación de caracteres. Caracteres especiales (escape)	10
2.3.3. Etiquetas o marcas	11
2.3.4. Elementos	11
2.3.5. Atributos	11
2.3.6. Comentarios	12
2.4. Documentos válidos y bien formados. Esquemas	12
3. Lenguajes de marcado para presentación de páginas web ..	15
3.1. Historia de HTML y XHTML. Diferencias entre versiones	17
3.2. Estructura de un documento	18
3.2.1. Versiones	19
3.2.2. Elementos de la cabecera	20
3.2.3. Elementos del cuerpo del documento	22
3.3. Color	22
3.3.1. Codificación de colores	22
3.3.2. Colores tipo	24
3.3.3. Colores seguros	24
3.4. Texto	24
3.4.1. Encabezados. Jerarquía y estructura del contenido de un documento	26
3.4.2. Párrafos	28
3.4.3. Alineación, espaciado y sangrado de texto	28
3.4.4. Características de letra: tipos, tamaños y colores	28
3.4.5. Separadores de texto	28
3.4.6. Etiquetas específicas para el marcado de texto. Estilos lógicos	28
3.5. Enlaces de hipertexto	30
3.5.1. Estructura de un enlace: la dirección de Internet o URL	30
3.5.2. Estilos de enlaces	31
3.5.3. Diferencias entre enlaces absolutos y relativos	31
3.5.4. Enlaces internos	31

3.5.5. Enlaces especiales: correo electrónico. Enlaces de descarga	32
3.5.6. Atributos específicos: título, destino, atajos de teclado, etc.	33
3.6. Imágenes	34
3.6.1. Formatos de imágenes	34
3.6.2. Características de imágenes: tamaño, título, textos alternativos	34
3.6.3. Enlaces en imágenes	39
3.6.4. Imágenes de fondo	39
3.7. Listas	39
3.7.1. Características	39
3.7.2. Ordenación de listas	40
3.7.3. Anidamiento de listas	41
3.7.4. Otros tipos de listas: listas de definición	41
3.8. Tablas	44
3.8.1. Estructura básica	44
3.8.2. Formato de tablas: bordes, alineación, tamaño, etc.	46
3.8.3. Formato de contenido de celdas. Agrupamiento de filas y columnas	46
3.8.4. Tablas anidadas	50
3.8.5. Buenas prácticas en el uso de tablas	51
3.9. Marcos	52
3.9.1. Creación de marcos	52
3.9.2. Ventajas e inconvenientes en el uso de marcos	55
3.9.3. Soporte de navegadores	56
3.9.4. Formateado de marcos	56
3.9.5. Enlaces entre contenido de marcos	57
3.9.6. Marcos anidados	57
3.9.7. Marcos incrustados (iframes)	59
3.10. Formularios	60
3.10.1. Descripción general y uso de formularios	61
3.10.2. Elementos de un formulario: texto, botones, etc.	63
3.10.3. Procesamiento de formularios	70
3.10.4. Formateado de formularios: atajos de teclado, orden de edición, grupos, etiquetas, etc.	71
3.11. Elementos específicos para tecnologías móviles	72
3.11.1. Selección del lenguaje de marcas para tecnologías móviles	73
3.11.2. Hojas de estilo en dispositivos móviles	73
3.12. Elementos en desuso (<i>deprecated</i>)	73
3.12.1. Texto parpadeante	73
3.12.2. Marquesinas	74
3.12.3. Alineaciones	74
3.12.4. Otros elementos en desuso	74
4. Hojas de estilo web	75
4.1. Tipos de hojas de estilo: estáticas y dinámicas	77
4.2. Elementos y estructura de una hoja de estilo	78
4.2.1. Creación de hojas de estilo	78
4.2.2. Aplicación de estilos	79
4.2.3. Herencia de estilos y aplicación en cascada	86
4.2.4. Formateado de páginas mediante estilos	88
4.2.5. Estructura de páginas mediante estilos	103
4.3. Diseño de estilos para diferentes dispositivos	115
4.4. Buenas prácticas en el uso de hojas de estilo	121

Contenidos

1. Diseño web

- Principios de diseño web:
 - Diseño orientado al usuario.
 - Diseño orientado a objetivos.
 - Diseño orientado a la implementación.
- El proceso de diseño web:
 - Estructura de un sitio web y navegabilidad.
 - Estructura y composición de páginas.
 - Compatibilidad con navegadores.
 - Diferencias entre diseño orientado a presentación e impresión.

2. Lenguajes de marcado generales

- Origen de los lenguajes de marcado generales: SGML y XML.
- Características generales de los lenguajes de marcado.
- Estructura general de un documento con lenguaje de marcado:
 - Metadatos e instrucciones de proceso.
 - Codificación de caracteres. Caracteres especiales (escape).
 - Etiquetas o marcas.
 - Elementos.
 - Atributos.
 - Comentarios.
- Documentos válidos y bien formados. Esquemas.

3. Lenguajes de marcado para presentación de páginas web

- Historia de HTML y XHTML. Diferencias entre versiones.
- Estructura de un documento:
 - Versiones.
 - Elementos de la cabecera.
 - Elementos del cuerpo del documento.
- Color:
 - Codificación de colores.
 - Colores tipo.
 - Colores seguros.

- **Texto.**
 - Encabezados. Jerarquía y estructura del contenido de un documento.
 - Párrafos.
 - Alineación, espaciado y sangrado de texto.
 - Características de letra: tipos, tamaños y colores.
 - Separadores de texto.
 - Etiquetas específicas para el marcado de texto. Estilos lógicos.
- **Enlaces de hipertexto:**
 - Estructura de un enlace: la dirección de internet o URL.
 - Estilos de enlaces.
 - Diferencias entre enlaces absolutos y relativos.
 - Enlaces internos.
 - Enlaces especiales: correo electrónico. Enlaces de descarga.
 - Atributos específicos: título, destino, atajos de teclado, etc.
- **Imágenes:**
 - Formatos de imágenes.
 - Características de imágenes: tamaño, título, textos alternativos.
 - Enlaces en imágenes.
 - Imágenes de fondo.
- **Listas:**
 - Características.
 - Ordenación de listas.
 - Anidamiento en listas.
 - Otros tipos de listas: listas de definición.
- **Tablas:**
 - Estructura básica.
 - Formato de tablas: bordes, alineación, tamaño, etc.
 - Formato de contenido de celdas.
 - Agrupamiento de filas y columnas.
 - Tablas anidadas.
 - Buenas prácticas en el uso de tablas.

- Marcos (frames):
 - Creación de marcos.
 - Ventajas e inconvenientes en el uso de marcos.
 - Soporte de navegadores.
 - Formateado de marcos.
 - Enlaces entre contenidos de marcos.
 - Marcos anidados.
 - Marcos incrustados (iFrames).
- Formularios:
 - Descripción general y uso de formularios.
 - Elementos de un formulario: texto, botones, etc.
 - Procesamiento de formularios.
 - Formateado de formularios: atajos de teclado, orden de edición, grupos, etiquetas, etc.
- Elementos específicos para tecnologías móviles:
 - Selección del lenguaje de marcas para tecnologías móviles.
 - Hojas de estilo en dispositivos móviles.
- Elementos en desuso (deprecated):
 - Texto parpadeante.
 - Marquesinas.
 - Alineaciones.
 - Otros elementos en desuso.

4. Hojas de estilo web

- Tipos de hojas de estilo: estáticas y dinámicas.
- Elementos y estructura de una hoja de estilo:
 - Creación de hojas de estilo.
 - Aplicación de estilos.
 - Herencia de estilos y aplicación en cascada.
 - Formateado de páginas mediante estilos.
 - Estructura de páginas mediante estilos.
- Diseño de estilos para diferentes dispositivos.
- Buenas prácticas en el uso de hojas de estilo.

1. Diseño web

Contenido

1.1. Principios de diseño web

1.2. El proceso de diseño web

El término diseño web se refiere habitualmente al diseño de la parte del cliente (*front-end*) de un sitio web. Abarca varias disciplinas: diseño gráfico, diseño de interfaces de usuario y conceptos de usabilidad y accesibilidad. Los diseñadores web deben manejar también diferentes tecnologías. Las básicas son HTML, CSS y JavaScript.

1.1. Principios de diseño web

El diseño de un sitio web es fundamental. Además de ser atractivo visualmente debe facilitar las tareas de los usuarios. Una página mal diseñada será difícil de entender y usar y por tanto los usuarios no querrán visitarla. Hay una serie de normas generales que el diseñador web debe tener siempre presente:

- **Usabilidad.** El diseño debe facilitar que el usuario encuentre la información fácil y rápidamente.
- **Consistencia.** Un sitio web suele estar formado varias páginas y secciones. Hay que mantener un diseño consistente entre todas las partes del sitio.
- **Velocidad de carga.** A los usuarios no le gusta esperar, hay que optimizar la velocidad de carga de la página.

1.1.1. Diseño orientado al usuario

En el diseño orientado al usuario, las características y necesidades del mismo se ponen en el centro del proceso de diseño, que se basará en las características de los usuarios, las tareas que quieren realizar y el entorno en que las realizan.

Es un proceso iterativo en que las propuestas de los diseñadores son probadas por usuarios reales. Los diseñadores incorporan las sugerencias de los usuarios a la siguiente versión del diseño.

1.1.2. Diseño orientado a objetivos

El diseño orientado a objetivos pone la atención en el propósito de la web, lo que los propietarios quieren conseguir con ella: puede ser una tienda online, un medio de comunicación o una página para una empresa. Ayuda a decidir qué funcionalidades debe incluir la página.

Las técnicas orientadas a objetivos son más apropiadas para las primeras fases del desarrollo de una web. Las centradas en el usuario se pueden usar más adelante, al detallar la interacción del usuario y evaluar la usabilidad de la página.

1.1.3. Diseño orientado a la implementación

El diseño orientado a implementación se fija en cómo se hace la página. Dentro del diseño web, el W3C (*World Wide Web Consortium*) publica los estándares para HTML, XHTML y CSS. Asegurar que las páginas los cumplen (dentro de lo posible) es una buena práctica que mejora la compatibilidad entre navegadores y facilita el mantenimiento del código desarrollado.

También incluye las cuestiones de accesibilidad, para que los usuarios con limitaciones (visuales, motrices, auditivas) puedan usar la página. Si se desea hacer un sitio web accesible, lo que en España es obligatorio para las páginas de la Administración Pública, hay que seguir las pautas de accesibilidad que publica el W3C dentro de la *Iniciativa para la Accesibilidad Web* (WAI, *Web Accessibility Initiative*).

1.2. El proceso de diseño web

El proceso de diseño de la interfaz suele comenzar con el esquema o plano del sitio web (*wireframe*). Al principio el esquema se centra en la disposición de la información y la navegación por el sitio, dejando de lado aspectos como tipografía, colores o imágenes de fondo. Posteriormente se puede ir refinando para obtener un mayor nivel de detalle.

1.2.1. Estructura de un sitio web y navegabilidad

El diseño de navegación se centra en los elementos que permiten que el usuario se mueva entre las páginas del sitio web. En una página suele haber más de una forma de navegación. Por ejemplo, en un periódico la información se organiza por secciones, pero también es posible ver las últimas noticias escritas o las más leídas.

1.2.2. Estructura y composición de páginas

Uno de los principales objetivos del esquema es el diseño de información. Se trata de cómo se estructura y muestra la información de la página, de manera que sea fácil de entender. La jerarquía visual de los contenidos facilita que el usuario encuentre la información que busca de manera rápida y contribuye a que la información se entienda mejor.

1.2.3. Compatibilidad con navegadores

Sin duda, la principal queja entre los diseñadores web es la cantidad de trabajo extra que deben dedicar a que sus páginas se vean igual en todos los navegadores, o al menos en los más usados (Internet Explorer, Chrome/Chromium, Mozilla Firefox, Opera y Safari). Además, no todos los usuarios tienen los navegadores actualizados, por lo que también hay que ocuparse de las versiones anteriores (para ayudar en esto hay librerías como *Modernizer*). Todos los ejemplos de este libro están probados con Firefox versión 29 a no ser que se especifique lo contrario.

1.2.4. Diferencias entre diseño orientado a presentación e impresión

A veces el usuario quiere imprimir el contenido que está viendo en una página web. Puede ser un artículo de periódico o el resumen de un pedido a una tienda online. Imprimir la página usando la función de imprimir del navegador no suele dar el resultado esperado porque imprimirá toda la página (elementos de navegación, imágenes) y no solo lo que se desea imprimir. Por eso es muy habitual que las páginas incorporen una versión para imprimir del contenido. Algunos de los de los cambios recomendados son:

- Revisar la fuente. Hay fuentes que se ven muy bien en la pantalla pero quedan mal en papel. También hay que revisar colores e imágenes de fondo.
- Quitar los elementos de navegación.
- Eliminar videos y animaciones y reducir el número de imágenes y anuncios.
- Incluir la URL de la página e información sobre autoría y/o copyright.

2. Lenguajes de marcado generales

Contenido

- 2.1. Origen de los lenguajes de marcado generales:
SGML y XML
- 2.2. Características generales de los lenguajes de marcado
- 2.3. Estructura general de un documento con lenguaje de marcado
- 2.4. Documentos válidos y bien formados. Esquemas

Los *lenguajes de marcado generales* son lenguajes que definen reglas para lenguajes de marcas. También se les denomina metalenguajes. Los más conocidos son SGML y XML. Tomando como base estos lenguajes generales se crean lenguajes para usos específicos, también llamados aplicaciones. Por ejemplo, se dice que HTML (hasta la versión 4) es una aplicación de SGML. Es decir, HTML es un lenguaje que sigue las reglas generales del SGML. De la misma manera, XHTML es una aplicación de XML.

2.1. Origen de los lenguajes de marcado generales: SGML y XML

El origen de los lenguajes de marcas está en las anotaciones que se hacían a los textos que se mandaban a la imprenta. Por ejemplo, un subrayado especial para indicar que algo debía imprimirse en negrita o símbolos especiales para delimitar una sección que debía aparecer como pie de página. Los lenguajes de marcas permiten insertar anotaciones dentro de un documento.

El SGML (Standard Generalized Markup Language) es un estándar ISO desde 1986. Es el sucesor de SGL, de IBM. El XML (eXtensible Markup Language, Lenguaje de marcas extensible) es un estándar del W3C. Se desarrolló para solucionar las limitaciones del SGML y al tiempo hacerlo más sencillo y flexible. Además, incluye el SGML como subconjunto, de manera que todo documento SGML es también un documento XML. Tuvo un gran éxito desde su publicación. Actualmente, hay decenas de lenguajes y formatos basados en XML y está muy extendido para la transmisión y almacenamiento de información.

2.2. Características generales de los lenguajes de marcado

Aunque no es necesario que sea así, la opción escogida por la mayoría de lenguajes de marcado es mezclar el contenido y las marcas en el mismo documento. Las marcas se diferencian del resto del contenido mediante caracteres especiales. Los documentos se almacenan en ficheros de texto plano.

2.3. Estructura general de un documento con lenguaje de marcado

En un fichero XML se distinguen dos partes:

- Un prólogo, opcional, con metadatos e instrucciones de proceso.
- Un elemento raíz que contendrá el resto de elementos.

2.3.1. Metadatos e instrucciones de proceso

Las instrucciones de proceso dan información sobre el documento (metadatos) al procesador. Por ejemplo, la primera línea de un documento XML suele ser:

```
<?xml version='1.0' encoding='UTF-8'?>
```

Es una instrucción que indica la versión del lenguaje y la codificación de caracteres utilizadas en el documento. También es posible asociar un fichero XML a un DTD o una hoja de estilo con las instrucciones:

```
<!DOCTYPE personas SYSTEM "personas3.dtd">  
<?xmlstylesheet type="text/xsl" href="incidencias.xslt"?>
```

2.3.2. Codificación de caracteres. Caracteres especiales (escape)

La codificación de caracteres es la manera en que almacenan los caracteres del documento. El procesador debe conocerla o puede que no muestre bien todos los caracteres. Al guardar el documento, hay que asegurarse de que la codificación sea la correcta. La codificación de caracteres en XML suele ser UTF-8 (también es la recomendada para HTML).

Los caracteres que se usan para delimitar las marcas o los valores de los atributos no pueden usarse dentro del documento directamente. Si queremos usarlos hay que *escapar el carácter*. En XML hay tres opciones:

- Entidades predefinidas XML. Hay cinco disponibles.

CARÁCTER	ENTIDAD
>	>
<	<
'	"
'	'
&	&

- Usar el código Unicode del carácter. Por ejemplo, para escribir una A (código 41 en hexadecimal) se podría usar A.

- Utilizar una sección CDATA. El texto dentro de la sección no se interpretará como un lenguaje de marcado. Es la mejor opción si hay que escapar muchos caracteres.

```
<![CDATA[...]]>
```

En HTML y XHTML hay más referencias definidas (ver sección 3.4).

2.3.3. Etiquetas o marcas

En HTML, XML y SGML las etiquetas se escriben entre los símbolos '<' y '>'.

```
<!--etiqueta de apertura-->
<a>
<!--etiqueta de cierre-->
</a>
<!--etiqueta de apertura y cierre-->
<meta/>
```

2.3.4. Elementos

En general, los elementos tienen una etiqueta de apertura y otra de cierre. El texto contenido entre ambas etiquetas es el texto marcado y está afectado por la etiqueta.

En algunos lenguajes, como HTML, se permiten algunos elementos sin etiqueta de cierre. En XML (y por tanto en XHTML), todos los elementos tienen que estar cerrados (en el caso de elementos vacíos se puede abrir y cerrar el elemento con una sola etiqueta como en el ejemplo anterior).

2.3.5. Atributos

Los elementos pueden tener atributos. Se sitúan en la etiqueta de apertura. No importa el orden en que aparezcan. En la mayoría de los casos los atributos aparecen en la forma nombre=valor.

```
<p class='importante'>Este párrafo es importante</p>
```

En XML, el valor de los atributos debe ir siempre entre comillas simples o dobles.

Algunos atributos son *booleanos*. En estos casos basta con poner el nombre del atributo, no hace falta poner el valor. Es el caso del atributo *disabled* en HTML.

```
<form disabled>
...
</form>
```

Atributos en HTML y XHTML

En HTML y XHTML hay atributos globales y específicos. Los globales se pueden usar con cualquier elemento. Se trata de los atributos *accesskey*, *class*, *contenteditable*, *dir*, *draggable*, *dropzone*, *hidden*, *id*, *lang*, *spellcheck*, *style*, *tabindex*, *title*, *translate*. Los atributos específicos solo son válidos para uno o varios elementos.

2.3.6. Comentarios

Los comentarios se usan para explicar el fichero. Pueden aparecer en cualquier lugar, menos dentro de una etiqueta y los procesadores los ignoran. Incluir comentarios es una buena práctica, sobre todo en ficheros complejos. Ya han aparecido en los ejemplos del capítulo.

```
<!--comentario-->
```

2.4. Documentos válidos y bien formados. Esquemas

En XML, se distingue entre documentos bien formados y documentos válidos. Se dice que un documento está bien formado si cumple las reglas de sintaxis de XML. Para que un documento pueda considerarse como XML, debe estar bien formado. Entre otras condiciones, esto implica que:

- Hay un elemento raíz que contiene todos los elementos.
- Todas las etiquetas están cerradas en el orden adecuado.
- En las etiquetas, se diferencia entre mayúsculas y minúsculas. Las etiquetas de apertura y cierre deben coincidir exactamente.
- Los valores de los atributos se escriben entre comillas.

Por ejemplo este fichero si está bien formado

```
<?xml version="1.0" encoding="UTF-8"?>
<trabajadores>
  <trabajador cod="t100">
    <nombreCompleto>Antonio Pérez</nombreCompleto>
    <nacimiento>1970</nacimiento>
    <carnetConducir/>
  </trabajador>
  <trabajador cod="t101">
    <nombreCompleto>Ana Gómez</nombreCompleto>
    <nacimiento>1980</nacimiento>
  </trabajador>
</trabajadores>
```

En cambio, el siguiente fichero no está bien formado. No tiene un solo nodo raíz, y en el segundo trabajador la etiqueta de cierre empieza por una mayúscula.

```
<?xml version="1.0" encoding="UTF-8"?>
<trabajador cod="t100">
  <nombreCompleto>Antonio Pérez</nombreCompleto>
  <nacimiento>1970</nacimiento>
  <carnetConducir/>
<trabajador>
<trabajador cod='t101'>
  <nombreCompleto>Ana Gómez</nombreCompleto>
  <nacimiento>1980</nacimiento>
</Trabajador>
```

Documentos válidos

Una de las novedades del SGML fue introducir el concepto de DTD (*Document Type Definition*). Con el DTD es posible definir el tipo de un documento, es decir: su estructura y qué etiquetas, atributos y valores pueden aparecer. Los documentos que cumplen las reglas establecidas en un DTD (y estén bien formados) se consideran válidos respecto a ese DTD. En XML también se pueden usar los esquemas XML (*XML Schema*) para validar documentos.

3. Lenguajes de marcado para presentación de páginas web

Contenido

- 3.1. Historia de HTML y XHTML. Diferencias entre versiones
- 3.2. Estructura de un documento
- 3.3. Color
- 3.4. Texto
- 3.5. Enlaces de hipertexto
- 3.6. Imágenes
- 3.7. Listas
- 3.8. Tablas
- 3.9. Marcos
- 3.10. Formularios
- 3.11. Elementos específicos para tecnologías móviles
- 3.12. Elementos en desuso (*deprecated*)

El HTML (*Hypertext Markup Language*, Lenguaje de Marcado de Hipertexto) y su lenguaje hermano XHTML (Extensible HTML) son los principales lenguajes de marcado para crear páginas web.

3.1. Historia de HTML y XHTML. Diferencias entre versiones

El HTML es el resultado del trabajo de Tim Berners-Lee durante los años ochenta y principios de los noventa del siglo pasado. La primera mención al HTML es de 1991, y en 1993 Berners-Lee y Dan Connolly publican la primera propuesta de especificación. A partir de 1996 el W3C se encarga de mantener el HTML.

La última versión es HTML5. Lleva desarrollándose varios años y se espera que se convierta en recomendación oficial a lo largo de 2014. Los navegadores no incluyen todavía todas las características, pero van ampliando el soporte rápidamente.

La versión anterior, 4.01, tiene 3 variantes:

- *Strict*. Estricta, no permite los elementos en desuso (*deprecated*). La mayoría son elementos con información sobre presentación.
- *Transitional*. Permite elementos en desuso.
- *Frameset*. Para páginas con marcos (*frames*). Se tratan en el apartado 3.9.

En las últimas versiones el HTML se ha centrado en la semántica. En HTML5 (y ya en la versión estricta de HTML 4.01), casi todas las etiquetas y atributos relacionados con la presentación de los contenidos han desaparecido. Ese tipo de información se deja para las hojas de estilo. El HTML debe centrarse en el contenido, su estructura y su significado, pero no en cómo debe mostrarse. Además, incluye nuevas etiquetas semánticas para organizar el contenido de los documentos.

El XHTML es un lenguaje diferente, basado en HTML. Simplificando, podemos decir que el XHTML es la versión XML del HTML. Las diferencias más importantes son:

- No se permiten elementos sin etiqueta de cierre. Los elementos vacíos se pueden abrir y cerrar en la misma etiqueta.
- El XHTML distingue entre mayúsculas y minúsculas.
- Los valores de los atributos tienen que estar entre comillas, simples o dobles.

Se definieron tres variantes del XHTML 1.0, equivalentes a las tres variantes de HTML 4.01. El XHTML 1.1 es como el XHTML 1.0 Strict pero con soporte para módulos.

Se suponía que la sintaxis más estricta y precisa del XHTML ayudaría a producir páginas libres de errores y por tanto se esperaba que los usuarios de HTML

estuvieran encantados de cambiar a XHTML. Esto no fue así. Aunque el XHTML gustó a algunos desarrolladores se comprobó que amplios sectores, sobre todo los diseñadores web, seguían prefiriendo el HTML. Por eso, cuando llegó el momento de desarrollar un nuevo estándar el W3C se decidió por el HTML5 en lugar del XHTML 2, y no se ha definido un XHTML equivalente. De cualquier manera, es posible escribir HTML5 de manera que sea XML bien formado.

HTML	XHTML
HTML 5	—
HTML 4.01 Strict	XHTML 1.0 Strict
HTML 4.01 Transitional	XHTML 1.0 Transitional
HTML 4.01 Frameset	XHTML 1.0 Frameset

Tabla 3.1. Equivalencia entre versiones de HTML y XHTML.

3.2. Estructura de un documento

El siguiente ejemplo muestra un documento HTML básico.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Primeras pruebas</title>
  </head>
  <body>
    Hola mundo
  </body>
</html>
```

Ejemplo 3.1. Página básica.

La primera línea es la declaración del tipo de documento. No es parte de la página propiamente dicha.

A continuación aparece el elemento `html` que a su vez contiene un elemento `head` y un elemento `body`. Toda la página está contenida dentro del elemento `html`. La última linea del fichero es la etiqueta de cierre del elemento `html`.

En el elemento `<head>` se incluye información sobre la página (autor, juego de caracteres, título) y en `<body>` el contenido propiamente dicho.

Vamos a utilizar el ejemplo 3.1 para introducir terminología habitual en el diseño web. El elemento raíz es `<html>`. Los elementos `<head>` y `<body>` son hijos de `<html>`, y por tanto `<html>` es el padre de ambos. Dos elementos con el mismo parente son hermanos (*siblings*). También es habitual hablar de descendientes (hijos, nietos...) y de antecesores (padres, abuelos...).

3.2.1. Versiones

La primera linea de cualquier documento HTML o XHTML suele ser la declaración del tipo de documento, o `<!DOCTYPE>`. La instrucción DOCTYPE (no es una etiqueta) indica al navegador el lenguaje y versión del documento. La forma en la que el navegador interpreta el documento depende de la declaración del tipo de documento, por lo que un error puede hacer que la página no se muestre correctamente.

La siguiente tabla muestra las declaraciones de tipo de documento correspondientes a las últimas versiones de HTML y XHTML.

Versión	Declaración de DOCTYPE
HTML 5	<code><!DOCTYPE html></code>
HTML 4.01 Strict	<code><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"></code>
HTML 4.01 Transitional	<code><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd" ></code>
HTML 4.01 Frameset	<code><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd" ></code>
XHTML 1.0 Strict	<code><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" ></code>
XHTML 1.0 Transitional	<code><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" ></code>
XHTML 1.0 Frameset	<code><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd" ></code>
XHTML 1.1	<code><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"></code>

Tabla 3.2. DOCTYPE para las diferentes versiones de HTML y XHTML.

3.2.2. Elementos de la cabecera

La cabecera de la página está formada por el elemento `<head>`. Aquí se incluyen metadatos, es decir, información sobre la página y se vinculan otros documentos como hojas de estilo y ficheros JavaScript.

La cabecera puede incluir los elementos: `title`, `link`, `meta`, `base`, `script`, `noscript`, `style` y `template`. Veamos los más importantes:

- El elemento `title` es obligatorio, contiene el título de la página. Los navegadores suelen mostrarlo en la pestaña y en la barra de programas.
- El elemento `meta` se utiliza para representar varios tipos de metadatos. Suelen usarse para especificar la codificación de caracteres o información como el autor, palabras clave o una descripción de la página. Se usa un formato de pares nombre/valor.

```
<meta name='author' content='Xabier Ganzábal'>  
<meta name='description' content='Página con ejemplos básicos de HTML'>
```

En HTML5 está disponible el atributo `charset` para especificar la codificación de caracteres:

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

Es importante incluir la codificación de caracteres del documento, porque si no especifica (o se hace incorrectamente) es posible que algunos caracteres, como las letras acentuadas y la ñ, no se vean bien. El W3C recomienda usar la codificación UTF-8. Al guardar el fichero, además de usar la extensión `.html` hay que asegurarse de que se guarda con la codificación de caracteres indicada.

El elemento `meta` es un elemento vacío. En HTML5 no tiene etiqueta de cierre, en XHTML debe cerrarse adecuadamente.

- El elemento `link` permite vincular hojas de estilo al documento HTML.

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

El atributo `rel` es obligatorio e indica la relación entre la página y el documento vinculado.

El atributo `href` indica la ruta a la hoja de estilo. Es un elemento vacío. En HTML5 no tiene etiqueta de cierre, en XHTML debe cerrarse adecuadamente. Profundizaremos en este elemento en el tema 4, cuando veamos las hojas de estilo.

- Con el elemento `script` se puede vincular un fichero de Javascript, indicando su URL en el atributo `src`.

```
<script type="text/javascript" src="ficherojs.js"></script>
```

También se puede usar para introducir directamente el código de un script. Por ejemplo, si se incluye en la cabecera

```
<script>alert('Hola')</script>
```

el navegador mostrará una alerta con texto 'Hola' al cargar la página.

- El elemento `style` permite añadir reglas de estilo en el propio documento HTML, sin necesidad de usar una hoja de estilos externa e incluirla en el documento con `link`. Es el único elemento presentacional de HTML5, y lo cierto es que no es habitual usarlo en producción, sólo para realizar prototipos rápidos. Lo usaremos en algunos ejemplos.

En el siguiente ejemplo podemos ver cómo queda una página con una cabecera completa,

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo de elementos de la cabecera</title>
    <meta charset='UTF-8'>
    <meta name='author' content='Xabier Ganzabal'>
    <meta name='description' content='Página con ejemplos básicos de HTML'>
    <!--incluir hoja de estilo-->
    <link rel="stylesheet" type="text/css" href="hojaEstilo.css">
    <!--incluir fichero javascript-->
    <script type="text/javascript" src="ficherojs.js"></script>
  </head>
  <body>
    Aquí va el contenido de la página...
  </body>
</html>
```

Ejemplo 3.2. Elementos de la cabecera.

3.2.3. Elementos del cuerpo del documento

El contenido de la página va dentro del elemento `body`. La página acaba con la etiqueta de cierre de este elemento. Es un elemento obligatorio, salvo en las páginas con `frameset` (sección 3.9).

Elementos de bloque y de línea

En HTML los elementos pueden ser de bloque o de línea. Esto afecta a como los muestra el navegador. Cuando se trata de un elemento de bloque se introduce un salto de linea antes y después del mismo. Los elementos de linea se muestran uno al lado del otro mientras quepan en el elemento contenedor.

3.3. Color

En HTML5 los atributos relacionados con el color han desaparecido. Como el resto de propiedades relacionadas con la presentación, actualmente debe especificarse con CSS.

3.3.1. Codificación de colores

Existen varias formas de codificar los colores.

- RGB (Red, Blue, Green). En esta codificación cada color se representa según la cantidad de rojo, azul y verde que contiene utilizando tres números entre 0 y 255.

Por ejemplo, la siguiente regla CSS hace que el color de fondo de la página sea rojo.

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

Alternativamente, se puede usar la notación hexadecimal, sustituyendo cada uno de los tres números por su equivalente en hexadecimal.

```
body { background-color: #ff0000 }
```

Como desventaja, la codificación RGB es poco intuitiva, y es difícil imaginar cual va a ser el color resultante si cambiamos alguno de los valores.

Color	RGB	HSL
Blanco	255, 255, 255	0, 0%, 100%
Negro	0, 0, 0	0, 0%, 0%
Rojo	255, 0, 0	0, 100%, 50%
Verde	0, 255, 0	120, 100%, 50%
Azul	0, 0, 255	240, 100%, 50%
Verde oscuro	0, 128, 0	120, 100%, 25%
Verde claro	128, 255, 128	120, 100%, 75%

Tabla 3.3. Equivalencia entre RGB y HSL.

- **RGBa.** Es similar a la codificación RGB, pero añade un valor de opacidad, el canal alpha. Se utiliza un cuarto número, entre 0 y 1. Si vale 1, es el color RGB normal. Al ir disminuyendo el valor, el color se vuelve más transparente, hasta llegar a 0, totalmente transparente.

```
body { background-color: rgba(255, 0, 0, 0.5) }
```

- **HSL (Hue-Saturation-Lightness).** El primer número, el matiz, es un número entre 0 y 360 que representa el color base. Los otros, saturación y luminosidad, toman un valor entre 0 y 1.

Tiene la ventaja de ser más intuitivo que RGB. Como se puede ver en la tabla 3.3, al variar el último número, la luminosidad, se obtiene el mismo color pero más claro u oscuro.

```
body { background-color: hsl(30, 100%, 20%) }
```

- **HSLa.** Es como HSL pero añadiendo el canal alpha, de manera análoga a la codificación RGBa.

```
body { background-color: hsla(30, 100%, 20%, 0.3) }
```

3.3.2. Colores tipo

Además de las opciones de codificación comentadas en el punto anterior, hay 16 colores que se pueden usar mediante su nombre según la especificación de HTML5: *aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white y yellow*.

Además de estos colores, en CSS3 se definen un total de 147 nombres de colores. Salvo excepciones, funcionan sin problemas en los navegadores más extendidos.

3.3.3. Colores seguros

Anteriormente, era habitual que muchos ordenadores no pudieran mostrar más de 256 colores. Si una página web requería un color no disponible, lo habitual era sustituirlo por el color disponible más parecido. Esto podía causar que muchas páginas no se vieran tal y como el diseñador había pensado.

Para solucionar este problema, se definieron los colores seguros (*web safe colors*), 216 colores estándar que en principio se mostrarían igual en todos los navegadores y monitores.

Actualmente la mayoría de los ordenadores y dispositivos móviles no tienen problemas para mostrar colores de al menos 16 bits, por lo que los colores seguros han dejado de tener utilidad.

3.4. Texto

En HTML los saltos de línea se ignoran y los espacios se colapsan. Por ejemplo, esta página

```
<!doctype html>
<html>
<head>
  <title>Primeras pruebas</title>
  <meta charset='UTF-8'>
</head>
<body>
  Todo en
  una linea
  v los espacios colapsan
</body>
<html>
```

Ejemplo 3.3. Espacios en blanco y saltos de línea.

Se mostraría así:

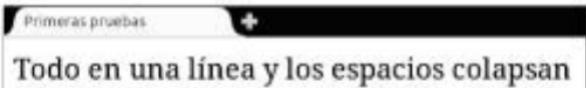


Ilustración 3.1 . Espacios en blanco en HTML.

Cuando se quiere un salto de línea, lo más habitual es usar el elemento `
`, que representa un salto de línea (no tiene etiqueta de cierre). También podemos usar la etiqueta `p`, para párrafos, que se trata en el apartado 3.4.2.

Respecto de los espacios, la primera opción es usar una referencia a carácter. Es posible reescribir el ejemplo anterior como

```
<body>
  Todo en <br>una linea<br>y los nbsp;&ampnbsp&ampnbsp&ampnbspespacios&ampnbsp
  &ampnbsp&ampnbsp&ampnbsp&ampnbspcolapsan
</body>
```

Ejemplo 3.4. Elemento `br` y espacios en blanco.

Otra opción en este caso es usar el elemento `pre`, para texto preformateado.

```
<pre>
  Todo en
  una linea
  y los   espacios           colapsan
</pre>
```

Ejemplo 3.5. Elemento `pre`.

El texto entre las etiquetas de `pre` se mostrará respetando espacios en blanco y saltos de línea.

Referencias a carácter

Es posible indicar un carácter mediante su código Unicode utilizando las referencias a carácter. Para algunos de estos símbolos se ha definido un nombre. Se trata de las entidades predefinidas de HTML.

Carácter	Referencia	Entidad	Carácter	Referencia	Entidad
Espacio	 	 	Salto de linea	
	
á	Á	´	í	á	á
é	É	´	é	é	é
í	Í	í	í	í	í
ó	Ó	ó	ó	ó	ó
ú	Ú	ú	ú	ú	ú
ñ	Ñ	Ñ	ñ	ñ	ñ

Tabla 3.4. Referencias a carácter y entidades predefinidas.

Es habitual usarlas para introducir espacios en blanco o saltos de línea. También se usan para caracteres especiales que no se visualizan correctamente. En el caso de páginas en español se pueden usar para las letras con acentos y la ñ (aunque no es necesario hacerlo si se indica correctamente el *charset* como se indicó en la sección 3.2.2).

3.4.1. Encabezados. Jerarquía y estructura del contenido de un documento

HTML dispone de seis elementos para los encabezados, *h1*, ..., *h6*, ordenados de mayor a menor importancia. Representan el encabezado de la sección en la que aparecen y deben contener contenido introductorio a la misma, por ejemplo, el titular de una noticia. Se muestran en negrita y más grandes que el resto del texto. El tamaño crece con la importancia del encabezado, por lo que el texto marcado como *h1* se mostrará más grande que el marcado con *h6*. Estos elementos no deben usarse para conseguir efectos visuales, sino para marcar texto que realmente sea un encabezado para el contenido que le rodea y establecer una jerarquía entre los contenidos.

Los elementos *h1*, ..., *h6* son elementos de bloque: al mostrarlos, se introduce un salto de linea antes y después del elemento.

Elementos semánticos

Una de las novedades más importantes en HTML5 son los elementos semánticos para organizar el contenido del documento. Se trata de *header*, *footer*, *aside*, *section*, *nav* y *article*. Antes de HTML5 se usaba el elemento *div* en lugar de todos estos elementos semánticos. Actualmente, solo es

apropiado usar *div* cuando no haya una etiqueta más apropiada. Para elegir el tamaño y la posición de estos elementos se utilizan las hojas de estilo.

Es importante señalar que, aunque por su nombre parece que indican la posición en la pantalla, se deben entender de una manera más amplia.

- *header* contiene contenido introductorio para la sección de la página en que aparece. Es habitual que contenga los elementos de encabezado, *h1*, ..., *h6*.
- *aside* debería usarse para contenido relacionado solo parcialmente con el contenido principal. En una revista o periódico, este tipo de información se suele presentar en una barra o cuadro lateral. No tiene por qué aparecer en un lateral de la página.
- *footer*. Contiene información sobre la sección correspondiente, como el autor, o información de *copyright*. No es obligatorio que aparezca en la parte de abajo de la sección, pero es lo más habitual.
- *nav* contiene vínculos, internos o externos. Es habitual que aparezca dentro de los elementos *header* o *aside* y que contenga una lista con los vínculos.



Ilustración 3.2 . Esquema de una página con etiquetas semánticas.

En la sección 4.2.5 se explica cómo usar estas etiquetas para estructurar una página web.

3.4.2. Párrafos

Podemos delimitar párrafos de texto con el elemento `p`. Los párrafos son elementos de bloque. No es raro ver ejemplos en los que no usa etiqueta de cierre. Está permitido si el siguiente elemento también es un párrafo (entre otros) o si no hay más elementos dentro del bloque contenedor. No es apropiado usarlos si sólo queremos dejar un espacio en blanco. Se utilizan en el ejemplo 3.6.

3.4.3. Alineación, espaciado y sangrado de texto

A partir de HTML5 los atributos y elementos relacionados con el formato de texto han desaparecido de la especificación. Actualmente, lo correcto es hacerlo a través de CSS, tal y como se explica en el capítulo 4.

3.4.4. Características de letra: tipos, tamaños y colores

Igual que con el formato de texto, los atributos y elementos relacionados con las fuentes de texto han desaparecido de la especificación. Actualmente, lo correcto es hacerlo a través de CSS, tal y como se explica en el capítulo 4.

3.4.5. Separadores de texto

El elemento `hr` se usa para introducir una separación entre dos bloques de contenido dentro de una misma sección. En los navegadores se muestra como una línea horizontal, que era precisamente la definición del elemento en las versiones anteriores del lenguaje. Antes tenía atributos para configurar aspectos de presentación, pero actualmente lo correcto es hacerlo mediante hojas de estilo.

3.4.6. Etiquetas específicas para el marcado de texto. Estilos lógicos

Las etiquetas de semántica a nivel de texto, o estilos lógicos, se usan para indicar qué tipo de información representa el texto que contienen.

En HTML5 los elementos `b`, `i`, `u` y `small` que antes eran presentacionales (`b` para negrita, `i` para cursiva, `small` para letra pequeña), han sido redefinidos para que ahora sean independientes del medio en que se presente la página.

Elemento	Descripción
<code>small</code>	Texto que normalmente aparece en letra pequeña (copyright, aclaraciones)
<code>span</code>	No aporta significado por sí sola, se usa para añadir atributos o estilo
<code>strong</code>	Contenido importante
<code>em</code>	Enfasis
<code>b</code>	Contenido resaltado, pero no por ser especialmente importante (por ejemplo, la primera frase de un párrafo)
<code>i</code>	Representa una parte de texto cualitativamente diferente del texto que la rodea (por ejemplo, términos en técnicos o en otro idioma)
<code>cite</code>	Referencia a una obra creativa
<code>abbr</code>	Abreviatura o acrónimo
<code>dfn</code>	Representa una definición
<code>mark</code>	El texto marcado es especialmente importante dentro del documento
<code>code</code>	Código de programación, rutas de ficheros o en general textos que un ordenador reconoce

Tabla 3.5. Elementos semánticos a nivel de texto.

El siguiente ejemplo muestra algunos de estos elementos.

```
<!DOCTYPE html>
<html>
<head>
<title>Semántica de texto</title>
<meta charset="UTF-8">
</head>
<body>
    El <abbr>W3C</abbr> se encargade definir las versiones de <ahref="HTML">HTML</ahref>
    <p>Los elementos <strong><strong></strong> y <b><b></b> pueden parecer
    iguales, pero no lo son.</p>
    <p>Lo mismo pasa con el <em>elemento em</em> y el <i>elemento i</i>
    </p>
    <p>Tienen <mark>distintos significados</mark> aunque los navegadores
    los muestren igual.</p>
    <p>Con la etiqueta span podemos añadir atributos y estilo a un trozo de
    texto, por
        ejemplo podemos hacer un <span style="font-size:xx-large">texto muy
        grande</span>
    </p>
    </body>
</html>
```

Ejemplo 3.6. Elementos semánticos a nivel de texto.

A continuación se muestra el resultado en el navegador.

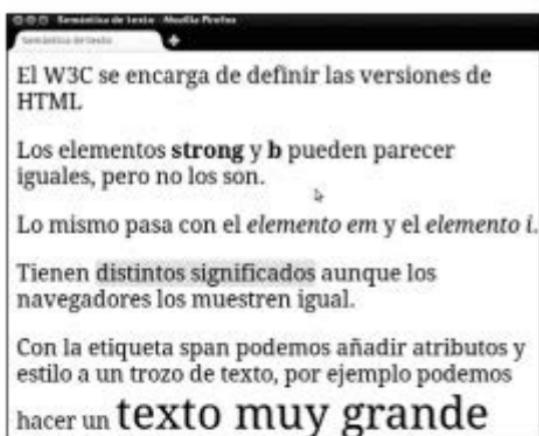


Ilustración 3.3 . Etiquetas de semántica de texto.

3.5. Enlaces de hipertexto

Los enlaces de hipertexto, también llamados hipervínculos o vínculos, son una de las piezas básicas del HTML. En el caso más habitual, el texto marcado como vínculo se mostrará subrayado y en un color diferente (azul o morado, dependiendo de si se ha visitado anteriormente o no), y al pulsar sobre él se cargará una nueva página web.

3.5.1. Estructura de un enlace: la dirección de Internet o URL

Para localizar un recurso en internet se usa su URL (siglas de *uniform resource locator*, localizador de recursos uniforme). Una URL tiene esta estructura:



Ilustración 3.4 . Estructura de una URL.

Protocolo: los habituales en Internet son http y https para conexiones seguras.

Dominio/servidor: nombre del servidor o IP. La URL de la ilustración 3.4 también podría escribirse así <http://81.89.32.200/publicaciones/novedades.html>.

Puerto: opcional, si no se especifica se usa el puerto por defecto para el protocolo. Para http, es el 80. La URL de la ilustración 3.4 también podría escribirse así <http://www.boe.es:80/publicaciones/novedades.html>.

Ruta: la ruta del recurso solicitado dentro del servidor.

3.5.2. Estilos de enlaces

Es habitual que los navegadores muestren en colores diferentes los vínculos ya visitados. Se puede modificar el estilo por defecto de los enlaces mediante las hojas de estilo, como veremos en la sección 4.2.2.

3.5.3. Diferencias entre enlaces absolutos y relativos

En un enlace relativo la URL es relativa a la página que contiene el vínculo. Por ejemplo, si la URL es `href='imagenes.html'`, el fichero imágenes.html se encuentra en el mismo directorio que la página actual. Al seguir un vínculo con `href='imagenes.html'` desde la página www.ejemplo.es/index.html se cargaría www.ejemplo.es/imagenes.html.

Una URL absoluta incluye un dominio.

```
*URL absoluta*
<a href="http://www.servidor.es/inicio.html">Visitar contenido principal</a>
*URL relativa*
<a href="inicio.html">Visitar contenido principal</a>
```

Ejemplo 3.7. URL absoluta y relativa

3.5.4. Enlaces internos

Es posible crear enlaces a un elemento determinado de la página. El enlace usa el atributo global `id` del elemento en cuestión. Por ejemplo, si en una página existe un párrafo como este

```
<p id="principal">Párrafo principal</p>
```

Podemos crear un enlace al párrafo desde la misma página usando

```
<a href="#principal">Visitar contenido principal</a>
```

También se pueden usar desde otra página, como parte de una URL absoluta o relativa

```
<a href="http://www.servidor.es/inicio.html#principal">Visitar contenido principal</a>
```

```
<a href="inicio.html#principal">Visitar contenido principal</a>
```

3.5.5. Enlaces especiales: correo electrónico. Enlaces de descarga

El comportamiento habitual de un enlace es cargar una nueva página en el navegador, pero hay otras opciones.

Los enlaces para enviar un correo electrónico son bastante habituales, aunque por desgracia no suelen resultar útiles. Al pulsar uno de estos enlaces lo normal es que se abra el cliente de correo predeterminado del sistema, si lo hay. En sistemas Windows suele ser Outlook. Si el usuario tiene una cuenta configurada en el cliente podrá mandar un correo, pero actualmente la mayoría de los usuarios usan un correo web.

```
<a href="mailto:ejemplo@noexiste.com">Enviar correo</a>
```

Otro enlace habitual es el de descarga, para almacenar un archivo en el ordenador cliente en lugar de visualizarlo en el navegador. Basta con usar el atributo booleano *download*.

```
<a download href="historico.zip">Descargar archivo histórico</a>
```

3.5.6. Atributos específicos: título, destino, atajos de teclado, etc.

Con el atributo `target` se especifica dónde debe abrirse el enlace: en la misma ventana (por defecto), en una nueva o en otro marco. El caso de los marcos se trata en el apartado 3.9.

```
<a href="www.servidor.es/inicio.html target=_blank">Visitar contenido principal</a>
<a href="inicio.html" >Visitar contenido principal</a>
<a href="inicio.html" target=_self >Visitar contenido principal</a>
```

Ejemplo 3.8. Atributos del elemento a.

La URL que queremos vincular se indica en `href`, como hemos ido viendo en los ejemplos del capítulo.

El atributo `rel` indica la relación con el elemento vinculado. Los valores admitidos se indican en la tabla 3.6.

Atributo	Valores posibles	Descripción
<code>href</code>	Una URL relativa o absoluta	URL del recurso enlazado
<code>target</code>	<code>_blank, _self, _top, _parent</code> id de un frame o iframe	Dónde se abrirá el enlace
<code>rel</code>	<code>alternate, autor, bookmark, help, license, next, nofollow, noreferrer, prefetch, prev, search, tag</code>	Relación entre el documento y el documento enlazado
<code>hreflang</code>	Identificador del idioma	Idioma del documento enlazado
<code>type</code>	Un tipo MIME válido	Tipo del documento enlazado
<code>download</code>	Atributo booleano	Si se usa, el recurso enlazado se <u>descarga</u>

Tabla 3.6. Atributos del elemento a.

Además de los atributos específicos, no es raro usar los atributos globales `title` y `accesskey` con los vínculos, para asociar títulos y atajos de teclado respectivamente.

```
<a href="inicio.html" accesskey='v' title='Principal'>Visitar contenido principal</a>
```

Ejemplo 3.9. Enlace con título y atajo de teclado.

3.6. Imágenes

Las imágenes se incluyen con el elemento `img`. Las imágenes no se insertan dentro del documento HTML como se insertarían en un documento de texto, sino que quedan vinculadas al mismo mediante su URL. Al encontrar un elemento `img`, el navegador solicitará la imagen al servidor correspondiente.

3.6.1. Formatos de imágenes

Se admiten imágenes en formato GIF, JPEG y PNG. También se pueden incluir ficheros PDF de una sola página, ficheros XML con SVG y animaciones (GIF animados y AJPEG).

3.6.2. Características de imágenes: tamaño, título, textos alternativos

El elemento `img` tiene, entre otros, estos atributos

Atributo	Descripción	Valor por defecto	Obligatoria
<code>src</code>	URL del fichero	No tiene	Sí
<code>alt</code>	Texto alternativo	No tiene	Sí
<code>height</code>	Alto de la imagen	El de la imagen	No
<code>width</code>	Ancho de la imagen	El de la imagen	No

Tabla 3.7. Atributos del elemento `img`.

El elemento tiene dos atributos obligatorios, `src` y `alt`. Por ejemplo:

```
<img src='arbol.jpg' alt='arbol de hojas rojas'>
```

Este elemento no tiene etiqueta de cierre.

El atributo `src` indica donde se encuentra la imagen. Es una URL, relativa o absoluta, al fichero que se desea incluir en la página. Aunque en principio podemos vincular cualquier imagen disponible en Internet usando su URL, muchos servidores impiden que se acceda a sus imágenes desde otros para no perder ancho de banda.

El atributo `alt` se usa para establecer un texto alternativo. Si el navegador, por cualquier motivo, no puede cargar la imagen muestra en su lugar este texto. Además, este atributo se usa en los lectores de pantalla. Por ejemplo, si el

usuario que está visitando la página no puede ver, recibirá el atributo `alt` como descripción de la imagen. Si no ponemos atributo `alt`, el navegador mostrará la imagen igualmente, pero no hay que olvidar que este atributo es obligatorio.

Los atributos `width` y `height` indican el ancho y el alto de la imagen, respectivamente. Si se escribe un número sin más, se interpreta que el tamaño se expresa en pixeles. Si no se especifican ancho y el alto, se toma el tamaño original de la imagen. Además, el navegador reserva el hueco adecuado para la imagen mientras recibe el fichero. Si solo se especifica uno, el navegador calcula el otro para mantener la proporción original de la imagen.

```
<img src='arbol.jpg' alt='arbol de hojas rojas' height='200' width='200'>
```

Este es uno de los pocos casos en que siguen incluyendo información sobre cómo debe presentarse el elemento en HTML5, en lugar de indicarlo a las hojas de estilo.

También se puede añadir un título a una imagen usando el atributo `title`, un atributo global que podemos usar con cualquier elemento HTML. Debe proporcionar información adicional sobre el elemento de manera concisa. La mayoría de los navegadores lo muestran en una pequeña caja de texto cuando el ratón pasa por encima.

El favicon

Hay un tipo de imagen muy habitual en las páginas web: el `favicon`. Es una imagen que identifica a la web a la manera de un logo. También se conoce como ícono de favoritos o ícono de la página. Por lo general es una imagen pequeña, habitualmente de 16 por 16 pixeles.

Existen algunas diferencias en los navegadores respecto a los formatos admitidos y la manera de representarlo. Los formatos más aceptados son ICO, PNG y GIF. Algunos navegadores admiten también animaciones. También varía como los usan los navegadores, pero por lo general se usa en la barra de direcciones, como ícono para los marcadores y como símbolo de las pestañas.

Para incluirlo, se usa la etiqueta `link` en la cabecera de la página.

```
<link rel="shortcut icon" href="favicon.ico">
```

En el atributo `rel` se puede poner “`shortcut icon`” o “`icon`” indistintamente.

```
<link rel="icon" href="favicon.ico" type="image/x-icon" />
```

El valor del atributo `type` indica el formato de la imagen, para GIF y PNG sería

```
<link rel="icon" href="favicon.gif" type="image/gif" />
<link rel="icon" href="favicon.png" type="image/png" />
```

El siguiente ejemplo muestran el elemento `img` y sus atributos, y además incluye un `favicon`.

```
<!DOCTYPE html>
<html>
  <head>
    <!--el favicon se incluye en la sección de cabecera-->
    <link rel="icon" href="favicon.ico" type="image/x-icon" />
    <meta charset='UTF-8'>
    <title>Imágenes</title>
  </head>
  <body>
    <p>Si no hay atributos height y width, coge los valores de la imagen</p>
    <img src='paisaje.jpg' alt='vistas al mar'>
    <p>Si la imagen no existe, se muestra el texto alternativo</p>
    <img src='paisajeInexistente.jpg' alt='vistas al mar'>
    <p>También se puede agrandar o reducir, y añadir un título, que se
       mostrará al pasar el ratón</p>
    <img src='paisaje.jpg' alt='vistas al mar' height='187' width='250'
         title='Vistas desde el restaurante'>
  </body>
</html>
```

Ejemplo 3.10. Elemento `img` y `favicon`.

Este es el resultado en el navegador



Ilustración 3.5 . Etiqueta img.

Audio y video

También es posible utilizar ficheros de audio y video en una página web. A lo largo de las versiones de HTML han existido varios elementos que por una razón u otra no han funcionado bien. A partir de HTML5 tenemos los elementos `audio` y `video`. Aun así, no todos los navegadores soportan los mismos formatos de audio y video. Por ejemplo, para asegurarnos de que un video se vea bien en la mayoría de los navegadores habrá que usar dos formatos para el mismo video, `.mp4` y `.ogg`. Tendremos que tener los dos ficheros disponibles en el servidor y

usar dos elementos `source` dentro del elemento `video`. El navegador utilizará la primera opción con un formato que entienda.

```
<video controls loop>
  <source src="fichero.mp4" type="video/mp4">
  <source src="fichero.ogg" type="video/ogg">
</video>
```

Ejemplo 3.11. Elemento video.

Lo mismo se puede decir del audio, hay que tener al menos dos versiones (`.mp3` y `.ogg`) de cada fichero.

```
<audio controls preload="none" autoplay>
  <source src="fichero.ogg" type="audio/ogg">
  <source src="fichero.mp3" type="audio/mpeg">
</audio>
```

Ejemplo 3.12. Elemento audio.

Atributo	Valores posibles	Descripción
<code>src</code>	Una URL.	Ruta al fichero con el audio/video
<code>controls</code>	Atributo booleano	Muestra controles para manejar el audio/video
<code>loop</code>	Atributo booleano	Reproduce el audio/video en continuo
<code>autoplay</code>	Atributo booleano	Empieza a reproducir el audio/video cuando se carga la página
<code>preload</code>	none; no carga el elemento hasta que se inicia la reproducción metadata; carga metadatos auto; carga el fichero con la página	Elige si el fichero se carga a la vez que la página o solo cuando vaya a reproducirse
<code>mute</code>	Atributo booleano	Solo para video. Si está presente el video se reproduce sin sonido
<code>poster</code>	Una URL.	Solo para video. Imagen que se muestra dentro del reproductor cuando el video no se está reproduciendo
<code>height</code>	Longitud, en cualquier unidad de medida CSS válida	Solo para video. Altura del reproductor de video
<code>width</code>	Longitud, en cualquier unidad de medida CSS válida	Solo para video. Anchura del reproductor de video

Tabla 3.8. Atributos de audio y video.

3.6.3. Enlaces en imágenes

Incluir un enlace en una imagen es sencillo, basta crear un elemento a que contenga el elemento img.

```
<a href='http://www.w3.org'><img src='planta.jpg' alt='una planta'></a>
```

3.6.4. Imágenes de fondo

En la versión 4.01 Transitional es válido usar el atributo *background* en el elemento *body*. Contiene la URL de la imagen que se quiere poner como fondo

```
<body background='fondo.jpg'>
```

Al tratarse de información de presentación, este atributo ya no está disponible en HTML5. Actualmente, las imágenes de fondo deberían incluirse a través de las hojas de estilo.

3.7. Listas

En HTML hay tres tipos de listas: ordenadas, no ordenadas y de definición. En cualquier caso, dentro de una lista hay varios elementos. Para las listas de definición cada elemento tiene dos partes: término y descripción. En la sección 4.2 veremos varias propiedades de estilo para mejorar la presentación.

3.7.1. Características

Las listas se crean habitualmente con los elementos *ul* (*unordered list*) para listas no ordenadas y *ol* (*ordered list*) para listas ordenadas. En las listas ordenadas, el marcador que acompaña a cada elemento es un número o una letra, mientras que en las no ordenadas es simplemente un punto.

En los dos casos, para cada elemento de la lista se usa la etiqueta *li* (*list item*, elemento de lista). No es necesario cerrar el elemento *li* si detrás de él viene otro igual o se cierra la lista.

El elemento *ul* no tienen atributos.

```
<ul>
    <li>Patatas
    <li>Pera
    <li>Leche
</ul>
```

Ejemplo 3.13. Lista no ordenada.

3.7.2. Ordenación de listas

Para listas ordenadas, se usa el elemento `ol`. Sus atributos son:

Atributo	Valores posibles	Descripción
<code>start</code>	Un número entero	Primer número de la lista
<code>type</code>	l - números decimales a - orden alfabético A - orden alfabético en mayúsculas i - números romanos I - números romanos en mayúsculas	Tipo de marcador de la lista
<code>reversed</code>	Atributo booleano	Si está presente, ordena la lista de mayor a menor

Tabla 3. 9. Atributos para listas ol.

El atributo `start` se usa para indicar el primer número de la lista. Con el atributo `type` se elige el tipo de marcador queremos poner antes de cada elemento de la lista. El atributo `reversed` es un atributo booleano. Si está presente la lista se mostrará de mayor a menor.

```
<ol>
    <li>Peras</li>
    <li>Manzanas</li>
    <li>Limones</li>
</ol>
```

Ejemplo 3.14. Lista ordenada.

```
<ol start="3" type="I">
<li>Peras</li>
<li>Manzanas</li>
<li>Limonas</li>
</ol>
```

Ejemplo 3.15. Lista ordenada empezando por el tres y con números romanos.

3.7.3. Anidamiento de listas

Es posible anidar listas poniendo un elemento `ol` o `ul` dentro de una lista. Se puede poner en lugar o dentro de un elemento `li`.

```
<ul>
<li>Ensalada</li>
<ul>
<li>Caprese</li>
<li>César</li>
</ul>
</ul>
```

Ejemplo 3.16. Lista anidada.

3.7.4. Otros tipos de listas: listas de definición

Las listas de definición incluyen grupos término-definición, como las entradas de un diccionario.

```
<dl>
<dt>Tomillo</dt>
<dd>Planta olorosa de la familia de las Labiadas</dd>
</dl>
```

Ejemplo 3.17. Lista de definición básica

El elemento `dl` se usa para definir las listas. En lugar de elementos `li`, dentro de las listas de definición se usan los elementos `dt` y `dd` para los términos y las definiciones, respectivamente. Estos dos elementos deben cerrarse siempre.

Puede haber términos con más de una definición y varios términos que comparten definición

```
<dl>
  <dt>Tomate</dt>
  <dd>Fruto de la tomatera</dd>
  <dd>Juego de naipes</dd>
  <dt>Perro</dt>
  <dt>Can</dt>
  <dd>Animal de compañía</dd>
</dl>
```

Ejemplo 3.18. Lista de definición.

A continuación, un ejemplo con varios tipos de lista en una página completa.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Listas</title>
  </head>
  <body>
    <p>Lista no ordenada</p>
    <ul>
      <li>Patatas</li>
      <li>Peras</li>
      <li>Leche</li>
    </ul>
    <p>Lista ordenada</p>
    <ol>
      <li>Peras</li>
      <li>Manzanas</li>
    </ol>
    <p>Lista ordenada alfabéticamente del revés</p>
    <ol type="a" reversed>
      <li>Peras
      <li>Manzanas
      <li>Limonas
      <li>Fresas
    </ol>
    <p>Lista anidada</p>
    <ul>
      <li>Ensalada</li>
      <ul>
        <li>Caprese</li>
        <li>César</li>
      </ul>
    </ul>
    <p>Lista de definición</p>
    <dl>
      <dt>Tomillo</dt>
      <dd>Planta olorosa de la familia de las Labiadas</dd>
      <dt>Tomate</dt>
      <dd>Fruto de la tomatera</dd>
      <dd>Juego de naipes</dd>
    </dl>
  </body>
</html>
```

Ejemplo 3.19. Página con varios tipos de lista.

El resultado en el navegador se puede ver en la ilustración 3.6.

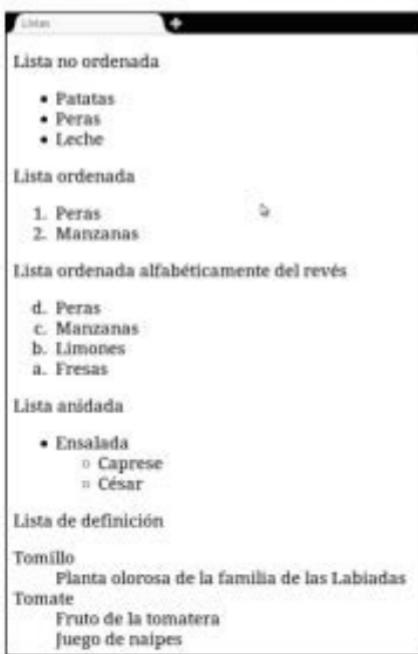


Ilustración 3.6 . Listas ordenadas, no ordenadas y de definición.

3.8. Tablas

Las tablas permiten agrupar datos en filas y columnas de celdas. Si están bien diseñadas pueden ser muy útiles para resumir información. Durante muchos años también se usaron para crear la estructura de las páginas, una práctica abandonada y mal vista actualmente.

3.8.1. Estructura básica

Una tabla está formada por filas, que a su vez están formadas por celdas de datos. Para crear una tabla se usan (entre otros) los siguientes elementos HTML:

- *table*, para crear la tabla.
- *tr*, para crear las filas.

- *td*, para crear las celdas de datos: pueden contener texto, imágenes, listas, otras tablas...
- *th*, para las celdas cabecera; aparecerán en un formato diferente, normalmente en negrita y centrados.
- *caption* permite añadir una leyenda a la tabla. Se trata de un texto que explica el contenido de la tabla y que se mostrará centrado sobre la misma.

Con estos elementos ya podemos crear una tabla básica.

```
<!DOCTYPE html>
<html>
<head>
<title>Tablas</title>
<meta charset='UTF-8'>
<style>table,td,th,tr{border: 1px black solid}</style>
</head>
<body>
<table>
<caption>Tabla de películas, directores y guionistas</caption>
<tr>
<th>Película</th>
<th>Director</th>
</tr>
<tr>
<td>Los otros</td>
<td>Alejandro Amenábar</td>
</tr>
<tr>
<td>Los pájaros</td>
<td>Alfred Hitchcock</td>
</tr>
</table>
</body>
</html>
```

Ejemplo 3.10. Tabla básica con caption.

El ejemplo incluye información de estilo con la etiqueta `style` en la cabecera (`head`) para mostrar la tabla con bordes, como es habitual. Las mismas reglas de estilo se aplican a todos los ejemplos de tablas de esta sección. En el capítulo 4 se tratan en detalle las reglas de estilo para tablas. El resultado del ejemplo 3.20 en el navegador será:

Película	Director
Los otros	Alejandro Amenábar
Los pájaros	Alfred Hitchcock

Ilustración 3.7 · Tablas con caption.

3.8.2. Formato de tablas: bordes, alineación, tamaño, etc.

En HTML5, han desaparecido los atributos del elemento `table` relativos al formato. El modo correcto de añadir estilo a una tabla es mediante hojas de estilo, que se verán más adelante.

3.8.3. Formato de contenido de celdas. Agrupamiento de filas y columnas

Los elementos `thead`, `tbody` y `tfoot` permiten agrupar filas. El elemento `thead` agrupa filas de encabezado para las columnas de la tabla, `tbody` las filas con el cuerpo de la tabla y `tfoot` filas con datos de resumen de la tabla. Estas etiquetas ayudan a explicar la tabla y también se pueden usar para aplicar diferentes estilos. Es habitual que los navegadores muestren las tablas en el orden `thead, tbody, tfoot` independientemente del orden en que aparezcan dentro del elemento `table`.

```
<table>
  <thead>
    <tr>
      <th>Evento</th>
      <th>Visitantes</th>
    </tr>
  </thead>
  <tbody>
    <tr>
```

```

<td>Inauguración</td>
<td>500</td>
</tr>
<tr>
<td>Clausura</td>
<td>400</td>
</tr>
</tbody>
<tfoot>
<tr>
<td>Total</td>
<td>900</td>
</tr>
</tfoot>
</table>

```

Ejemplo 3.21. Elementos `thead`, `tbody` y `tfoot`.

Atributos `colspan` y `rowspan`

Es posible hacer que una de las celdas de la tabla ocupe más de una columna o más de una fila usando los atributos `colspan` y `rowspan`, respectivamente. Se pueden aplicar a los elementos `td` y `th`. Hay que tener cuidado de que las celdas no se solapen.

En el siguiente ejemplo la primera tabla contiene celdas que ocupan más de una columna y la segunda celdas que ocupan más de una fila.

```

<table>
  <caption>Tabla de películas, directores y guionistas</caption>
  <tr>
    <th>Película</th>
    <th>Director</th>
    <th>Guionista</th>
  </tr>
  <tr>
    <td>Los otros</td>
    <td colspan='2'>Alejandro Amenábar</td>
  </tr>
  <tr>

```

```

<td>Los pájaros</td>
<td>Alfred Hitchcock</td>
<td>Evan Hunter</td>
</tr>
</table>
<table>
<caption>Tabla de películas y directores</caption>
<tr>
<th>Película</th>
<th>Director</th>
</tr>
<tr>
<td>Los otros</td>
<td rowspan='2'>Alejandro Amenábar</td>
</tr>
<tr>
<td>Abre los ojos</td>
</tr>
<tr>
<td>Los pájaros</td>
<td>Alfred Hitchcock</td>
</tr>
</table>

```

Ejemplo 3.22. Tabla con colspan y rowspan.

El resultado en el navegador será

Película	Director	Guionista
Los otros	Alejandro Amenábar	
Los pájaros	Alfred Hitchcock	Evan Hunter

Película	Director
Los otros	Alejandro Amenábar
Abre los ojos	
Los pájaros	Alfred Hitchcock

Ilustración 3.8 · Tablas con colspan y rowspan.

Elementos `colgroup` y `col`

También es habitual usar el elemento `colgroup` para aplicar las mismas reglas de estilo a varias columnas, como se muestra en este ejemplo.

```
<table>
  <caption>Tabla de películas, directores y guionistas</caption>
  <colgroup span="2" style="background-color:yellow"></colgroup>
  <tr>
    <th>Película</th>
    <th>Director</th>
    <th>Guionista</th>
  </tr>
  <tr>
    <td>Los otros</td>
    <td>Alejandro Amenábar</td>
    <td>Alejandro Amenábar</td>
  </tr>
  <tr>
    <td>Los pájaros</td>
    <td>Alfred Hitchcock</td>
    <td>Evan Hunter</td>
  </tr>
</table>
```

Ejemplo 3.23. Elemento `colgroup`.

El color de fondo de las dos (valor del atributo `span`) primeras columnas cambia. Usando el atributo `style`, se introduce una regla de estilo para cambiar el color de fondo de esas columnas. El atributo global `style` se trata en profundidad en el capítulo 4.

También es posible usar `colgroup` en combinación con `col`. Podemos indicar reglas de estilo columna a columna. En este caso `colgroup` no puede tener atributo `span`, pero los elementos `col` sí.

```
<table>
  <caption>Tabla de películas, directores y guionistas</caption>
  <colgroup>
    <col style="background-color:yellow"></col>
    <col span="2" style="background-color:lightblue"></col>
  </colgroup>
```

```

<tr>
<th>Película</th>
<th>Director</th>
<th>Guionista</th>
</tr>
<tr>
<td>Los otros</td>
<td>Alejandro Amenábar</td>
<td>Alejandro Amenábar</td>
</tr>
<tr>
<td>Los pájaros</td>
<td>Alfred Hitchcock</td>
<td>Evan Hunter</td>
</tr>
</table>

```

Ejemplo 3.24. Elementos *colgroup* y *col*.

Podemos ver el resultado de los ejemplos de los dos ejemplos anteriores en la ilustración 3.9.

Tabla de películas, directores y guionistas		
Película	Director	Guionista
Los otros	Alejandro Amenábar	Alejandro Amenábar
Los pájaros	Alfred Hitchcock	Evan Hunter

Tabla de películas, directores y guionistas		
Película	Director	Guionista
Los otros	Alejandro Amenábar	Alejandro Amenábar
Los pájaros	Alfred Hitchcock	Evan Hunter

Ilustración 3.9 . Tablas con *colgroup*.

3.3.4 Tablas anidadas

Es posible anidar una tabla dentro de otra. Basta con definir un nuevo elemento *table* dentro de una celda.

```

<table>
  <caption>Tablas anidadas</caption>
  <tr>
    <th>Columna 1</th>
    <th>Columna 2</th>
  </tr>
  <tr>
    <td>Celda normal</td>
    <td>
      <table>
        <tr>
          <td>1</td>
          <td>2</td>
        </tr>
        <tr>
          <td>3</td>
          <td>4</td>
        </tr>
      </table>
    <td>
      <tr>
        <td></td>
      </tr>
    </td>
  </tr>
</table>

```

Ejemplo 3.25. Tablas anidadas.

En el navegador se mostrará así:

Tablas anidadas					
Columna 1	Columna 2				
Celda normal	<table border="1"> <tr> <td>1</td><td>2</td></tr> <tr> <td>3</td><td>4</td></tr> </table>	1	2	3	4
1	2				
3	4				

Ilustración 3.10 . Tablas anidadas.

3.8.5. Buenas prácticas en el uso de tablas

Es importante que la tabla sea fácil de leer. En este sentido se recomienda:

- Distinguir los encabezados del resto de celdas.

- Alternar el color de fondo de las filas.
- Usar bordes.
- Partir las tablas complejas en varias sencillas.
- Usar el elemento *caption* para añadir un texto informativo a la tabla.

Además, solo hay que usar las tablas cuando corresponda:

- Todas las filas tienen que tener los mismos datos. Si hay varias filas con columnas vacías, tal vez sea mejor usar una lista o partir la tabla en varias.
- No hay que usar las tablas para estructurar las páginas. Era una práctica habitual hace años pero ahora debe hacerse mediante hojas de estilo.

3.9. Marcos

Hasta hace unos años, los marcos eran muy habituales en el diseño web. Permiten dividir el área en que se representa la página, la pantalla, en varios marcos o vistas. En cada uno de ellos se carga una página HTML diferente.

En HTML5 los marcos han desaparecido. Se debe a que sirven para maquetar las páginas y, como ya se ha dicho, actualmente la presentación de las páginas web se especifica mediante hojas de estilo. En lugar de usar marcos, lo habitual es usar elemento *div* o las etiquetas semánticas (*head*, *section*, *nav*, *footer*, *aside*) para estructurar la página en secciones y posicionarlas en la pantalla mediante la CSS. Sigue siendo válido el elemento *iframe*, que se trata en el apartado 3.9.7.

3.9.1. Creación de marcos

La última versión de HTML en la que están disponibles los marcos es 4.01 Frameset. En XHTML, la versión equivalente es XHTML 1.0 Frameset. Si queremos incluir marcos en nuestra página hay que usar el DOCTYPE correspondiente (ver apartado 3.2.1).

Para crear marcos usamos los elementos *frameset* y *frame*. El primero nos sirve para definir un conjunto de marcos, y el segundo para cada uno de los marcos.

En una página con marcos el elemento *frameset* sustituye al elemento *body*. No pueden aparecer los dos. El elemento *frameset* tiene dos atributos, *cols* y *rows*. Sirven para especificar el número y el tamaño de las secciones en las que queremos dividir la página. Dentro del elemento *frameset* deben aparecer tantos elementos *frame* como marcos se hayan definido.

Por ejemplo, una estructura básica para una página podría ser la siguiente: una barra lateral con enlaces de navegación y una parte central en la que se muestra el contenido.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd ">

<html>
  <head>
    <title>Página con marcos</title>
  </head>
  <frameset cols="20%,80%">
    <frame src="marco1_base.html"></frame>
    <frame src="marco2_base.html"></frame>
  </frameset>
</html>
```

Ejemplo 3.26. Página con dos marcos verticales.

El resultado en el navegador sería

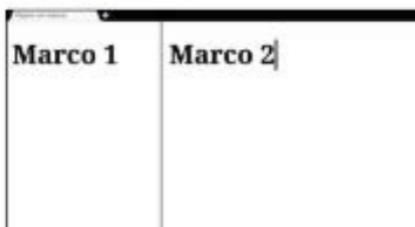


Ilustración 3.11 - Página con dos marcos verticales.

Para especificar el tamaño de las filas o columnas se pueden usar porcentajes, píxeles o una mezcla de ambos. También es posible usar el carácter '*' para indicar al navegador que una columna debe ocupar el espacio que dejen disponibles el resto. La tabla 3.10 muestra varios ejemplos.

<code>cols="20%,80%"</code>	Dos columnas, la primera ocupa el 20% del espacio disponible y la segunda el 80 %
<code>cols="20%,*"</code>	Igual que el anterior
<code>cols="20%,*,30%"</code>	Tres columnas, la primera ocupa el 20% del espacio disponible, la segunda el 50% y la tercera el 30%
<code>cols="20%,400,*"</code>	Tres columnas, la primera ocupa el 20% del espacio disponible, la segunda 400 pixels y la tercera el resto de espacio disponible. Ejemplo: si la pantalla tiene 1000 pixels de ancho, la primera columna ocuparía 200 pixels, y la segunda y la tercera 400 pixels cada una
<code>cols="20%,*,200,2*"</code>	Cuatro columnas, la primera ocupa el 20% del espacio disponible, la tercera 200 pixels. La segunda y la cuarta se reparten el resto del espacio de manera que la cuarta ocupa el doble que la segunda Ejemplo: si la pantalla tiene 1000 pixels de ancho, la primera y la tercera columna ocuparían 200 pixels cada una. Los 600 pixels restantes se reparten entre las otras dos columnas, 200 para la segunda y 400 para la cuarta

Tabla 3.10. Tamaño de los marcos.

El atributo `rows` funciona igual que `cols`, pero divide la página por filas en lugar de por columnas.

```
<frameset rows="20%,80%">
    <frame src="marco1_base.html"></frame>
    <frame src="marco2_base.html"></frame>
</frameset>
```

Ejemplo 3.27. Página con dos marcos horizontales.

En este caso, el resultado es



Ilustración 3.12 · Página con dos marcos horizontales.

Cuando usamos el atributo `cols` pero no `rows`, las columnas definidas ocupan todo el espacio vertical disponible. Igualmente, si solo se especifica `rows`, las filas definidas ocupan todo el ancho disponible. Es posible usar los dos a la vez. Por ejemplo, podemos dividir la página en cuatro marcos, usando dos filas y dos columnas.

```
<frameset rows="20%,80%" cols="50%,50%">
    <frame src="marco1_base.html"></frame>
    <frame src="marco2_base.html"></frame>
    <frame src="marco3_base.html"></frame>
    <frame src="marco4_base.html"></frame>
</frameset>
```

Ejemplo 3.28. Página con cuatro marcos.

Marco 1	Marco 2
Marco 3	Marco 4

Ilustración 3.13 . Página con cuatro marcos.

No es muy habitual usar los dos atributos a la vez. Para conseguir estructuras más complicadas se suelen utilizar marcos anidados.

3.9.2. Ventajas e inconvenientes en el uso de marcos

Ventajas:

- Una manera rápida de implementar estructuras de páginas sencillas.
- Permite organizar el contenido de la página en varios ficheros HTML.

Desventajas:

- Incluye información de presentación dentro del HTML.
- Problemas de compatibilidad, especialmente en dispositivos móviles.

- La URL de una página con un *frameset* no cambia aunque el contenido de los marcos si cambie al seguir un vínculo. Esto es un problema a la hora de guardar un marcador o enviar la URL. Al volver a visitar la página accederemos al contenido inicial de los marcos.
- Causan problemas a los motores de búsqueda, que no saben qué marcos corresponde usar para indexar la página.

3.9.3. Soporte de navegadores

La mayoría de los navegadores modernos (para ordenador) aceptan la especificación de marcos sin problemas. De igual manera, hay diferencias entre navegadores a la hora de interpretar los marcos y es habitual que una misma página no se muestre exactamente igual en todos. Además, normalmente los navegadores para móviles y los lectores de pantalla no incluyen soporte para marcos.

El elemento *<noframes>* se usa habitualmente para mostrar un mensaje apropiado en el caso de que el navegador no tenga soporte para marcos. Cuando sí hay soporte, el navegador ignora este elemento.

```
<noframes>
  <p>Lo sentimos, su navegador no soporta marcos </p>
</noframes>
```

Ejemplo 3.29. Elemento *noframe*.

3.9.4. Formateado de marcos

El elemento *frame* tiene algunos atributos para cambiar la presentación.

Atributo	Valores posibles	Descripción
<i>frameborder</i>	0 - sin borde 1 - con borde, es el valor por defecto	Para mostrar un borde alrededor del marco
<i>marginheight</i>	Un número	Altura del margen en píxeles
<i>marginwidth</i>	Un número	Anchura del margen en píxeles
<i>scrolling</i>	<i>yes</i> <i>no</i> <i>auto</i> - valor por defecto	Indica si se deben mostrar barras de desplazamiento en el marco
<i>noresize</i>	Atributo booleano	Si aparece este atributo, no se permite modificar el tamaño del marco

Tabla 3.11. Atributos del elemento *frame*.

3.9.5. Enlaces entre contenido de marcos

Al pulsar un vínculo, es posible hacer que este se cargue en el marco que deseemos poniendo como valor del atributo target el nombre (atributo `name`) del marco elegido, como se puede ver en el ejemplo 3.30.

3.9.6. Marcos anidados

Es habitual anidar un elemento `frameset` dentro de otro, como por ejemplo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

<html>
  <head>
    <meta charset="UTF-8">
    <title>Página con marcos</title>
  </head>
  <!--Un primer frameset para crear una parte superior de cabecera y una parte central-->
  <frameset rows="20%,80%">
    <!--marco para la parte superior-->
    <frame src="cabecera.html">
    <!--partimos la parte inferior en dos, la izquierda para vínculos la derecha para el contenido-->
    <frameset cols="25%,75%">
      <!--frame para los vínculos de navegación-->
      <frame src="navegacion.html">
      <!--frame en el que se mostrará el contenido, con contenido inicial-->
      <frame name="principal" src="listas.html">
    </frameset>
  </frameset>
</html>
```

Ejemplo 3.30. Marcos anidados.

A continuación podemos ver el aspecto en el navegador.



Ilustración 3.14 . Página con dos marcos añadidos.

Conviene mostrar también el código de *navegacion.html* , el fichero que se carga en el marco inferior izquierdo con los vínculos de navegación. Se puede observar que el valor de *target* de los vínculos es *principal*, que el nombre (atributo *name*) del marco inferior derecho.

```
<!DOCTYPE html>
<html>
<head>
<meta charset='UTF-8'>
<title>Para la barra de navegación</title>
</head>
<body>
    Se abren el el frame de la derecha
    <ul>
        <li><a href="imagenes.html" target="principal">Ejemplo 1</a><br>
        <li><a href="listas.html" target="principal">Ejemplo 2</a><br>
        <li><a href="tablas_colgroup.html" target="principal">Ejemplo 3</a>
    </ul>
</body>
</html>
```

*Ejemplo 3.31. Código de *navegacion.html*, que se usa en el ejemplo 3.30.*

3.9.7. Marcos incrustados (iframes)

El elemento `iframe` es una alternativa a los marcos que hemos visto hasta ahora. Siguen siendo válidos en HTML5. Para colocarlos en la pantalla y darles tamaño se usan las hojas de estilo. El contenido inicial se especifica con el atributo `src` y, como con los elementos `frame`, podemos usar el atributo `name` para que hacer que un enlace se cargue dentro de un `iframe` concreto.

```
<!DOCTYPE html>
<html>
<head>
<meta charset='UTF-8'>
<title>Página con iframe</title>
</head>
<body>
<ul>
<li><a href="cap3_ejemplo19_listas.html" target="principal">Ejemplo 1</a></li>
<li><a href="cap3_ejemplo20_tablas_basicas.html" target="principal">Ejemplo 2</a></li>
<li><a href="cap3_ejemplo10_imagenes.html" target="principal">Ejemplo 3</a></li>
</ul>
<p>Los vínculos se cargan en este iframe</p>
<iframe height="400" width="600" name="principal" src ="cap3_ejemplo19_listas.html">
</body>
</html>
```

Ejemplo 3.32. Página con iframe.

En la ilustración 3.15 se muestra el resultado. Los vínculos se cargan en el `iframe` que tienen debajo.



Ilustración 3.15 . Página con iframe.

El elemento `iframe` tiene más atributos que los que se usan en el ejemplo anterior. La siguiente tabla los resume.

Atributo	Valores posibles	Descripción
<code>height</code>	Un número	Altura del marco en pixeles
<code>width</code>	Un número	Ancho del marco en pixeles
<code>src</code>	Una URL	Ruta al fichero que queremos que cargue el <code>iframe</code>
<code>srcdoc</code>	Una cadena con código HTML	Código HTML para el <code>iframe</code> . Tiene prioridad sobre <code>src</code>
<code>name</code>	Una cadena de caracteres, sin espacios	Nombre del marco. Se usa para cargar el contenido de los enlaces en el <code>iframe</code>
<code>sandbox</code>	allow-forms allow-pointer-lock allow-popups allow-same-origin allow-scripts allow-top-navigation	Opciones de seguridad para contenido del <code>iframe</code> . Pueden aparecer varios valores separados por espacios

Tabla 3.12. Atributos de iframe.

3.10. Formularios

Hay muchas novedades en HTML5 relativas a los formularios, una parte muy importante del diseño web. Hay que tener en cuenta que el tratamiento

completo de los formularios requiere en general procesamiento tanto en el cliente como en el servidor, áreas que no se tratan en este libro.

3.10.1. Descripción general y uso de formularios

Los formularios se usan principalmente para solicitar información al usuario, aunque también es posible manejarlos desde JavaScript en aplicaciones que requieran interacción con el usuario, como una calculadora. Muestran al usuario una serie de controles como: campos de texto, botones, casillas de verificación, listas desplegables... Suelen tener un botón de envío. Cuando se pulsa el botón de enviar, los datos se envían al servidor web que se encargará de procesarlos. Como ejemplo típico, podemos pensar en la página de acceso (*login*) a nuestro correo web.

1. El primer paso consiste en acceder al servidor de correo mediante el navegador.
2. El servidor envía la página, que contiene un formulario de entrada similar a este.

```
<!doctype html>
<html>
    <head>
        <title>Formularios</title>
    </head>
    <body>
        Por favor, introduzca usuario y contraseña
        <form name="login" autocomplete action="procesar.php"
method="post">
            Usuario: <input type="text" name="user"><br>
            Contraseña: <input type="password"
name="passw"><br>
            <input type="submit" value="Enviar">
            <input type="reset" value="Limpiar">
        </form>
    </body>
</html>
```

Ejemplo 3.33. Formulario de login.

El aspecto de este formulario en el navegador, será

A screenshot of a web browser window titled "Formularios". Inside, there is a login form with the following fields:

- A label "Por favor, introduzca usuario y contraseña".
- A text input field labeled "Usuario: " followed by the placeholder "usuario".
- A text input field labeled "Contraseña: " followed by a series of dots representing a password.
- Two buttons at the bottom: "Enviar" (Send) on the left and "Limpiar" (Clear) on the right.

Ilustración 3.16 . Formulario de login.

3. El usuario rellena nombre de usuario y contraseña y pulsa a enviar. El servidor recibe los datos que acaba de introducir.
4. El servidor comprueba en su base de datos si el nombre de usuario y contraseña recibidos son correctos.
5. Si son correctos, se muestra la bandeja de entrada del usuario. Si no son correctos, se muestra un mensaje de error y se vuelve a ofrecer el formulario de acceso.

En el ejemplo anterior podemos ver algunas de las características básicas de los formularios:

- Para crear formularios se usa el elemento `form`. Puede tener un nombre (atributo `name`). No debe haber dos formularios con el mismo nombre en una página.
- Dentro del elemento `form` se colocan todos los controles que forman parte del usuario. En este caso dos cajas de texto y dos botones.
- Los botones de envío o de limpiar campos afectan a los controles situados dentro del mismo elemento `form`.
- Los atributos `action` y `method` definen la comunicación con el servidor. Los trataremos en el apartado 3.10.3.
- El atributo booleano `autocomplete` indica que debe permitirse la opción de autocompletado para el formulario. También tiene que estar activada en el navegador.
- Todos los elementos que permiten al usuario introducir información tienen que tener un atributo `name`. Este nombre se usa para pasar la información introducida al servidor.

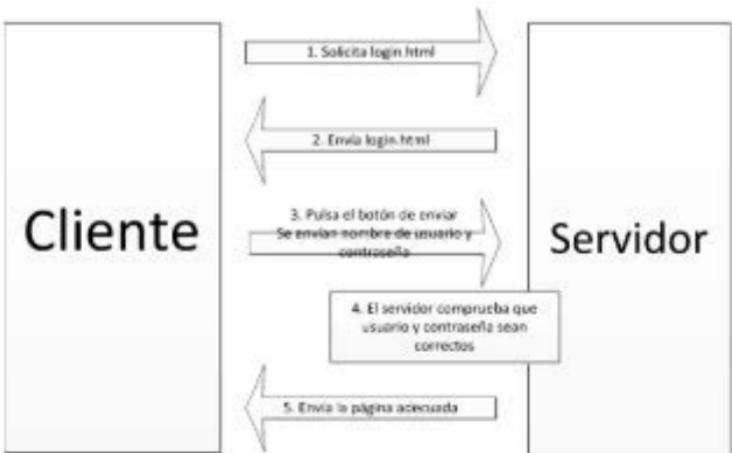


Ilustración 3.17 . Esquema básico de comunicación entre cliente y servidor.

3.10.2. Elementos de un formulario: texto, botones, etc.

El elemento básico para crear los controles de un formulario es `input`. Se puede usar para varios tipos de controles, según el valor que tome el atributo `type` (en la especificación de HTML, estos valores se denominan *estados del control*). Algunos de estos valores se han introducido en HTML5 y por el momento los navegadores solo los soportan parcialmente.

Algunos de los más habituales son:

- Campo de texto (`type='text'`, o sin `type`, ya que es el valor por defecto). Muestra para una caja de texto de una sola línea. Se puede crear con un texto inicial usando el atributo `value`.
- Campo para contraseña (`type='password'`). Muestra un campo de texto como el anterior, pero el texto introducido se muestra enmascarado.
- Botones de radio (`type='radio'`). Cuando se marca un botón de radio, los demás botones que tengan el mismo valor en el atributo `name` se deseleccionan. Si deseamos tener varios grupos de botones de radio, hay que asegurarse de que tengan nombres diferentes. Para que un botón de radio esté marcado al cargar la página, se le añade el atributo `checked`.

- Casillas de verificación (`type='checkbox'`). Muestra unas casillas que se pueden marcar o desmarcar. No afecta a otras casillas de verificación, aunque tengan el mismo valor en el atributo `name`. Para que una casilla esté marcada al cargar la página, se le añade el atributo `checked`.
- Botones. Con `type='submit'` se crea un botón de envío del formulario. Con `type='reset'`, un botón que limpia todos los campos del formulario. Con `type='button'`, crea un botón genérico al que habrá que dotar de funcionalidad usando *JavaScript*. En los tres casos se puede especificar el texto del botón con el atributo `value` (para los botones de `submit` y `reset` hay valores por defecto si no se usa el atributo `value`).
- Selector de ficheros (`type='file'`). Sirve para seleccionar ficheros del ordenador del cliente. Muestra un que abre un cuadro de diálogo para realizar la selección. También se muestra la ruta del fichero seleccionado en un campo de texto o similar (hay diferencias entre navegadores).
- Correo electrónico (`type='email'`). Es un campo de texto que solo admite uno o varios correos electrónicos. No permite el envío del formulario hasta que no se introduzca una dirección de correo válida. No comprueba que la que la dirección de correo exista realmente, sino que la cadena introducida cumpla ciertas condiciones. Novedad en HTML5.
- Selector de fechas (`type='date'`). Para introducir fechas, es una novedad en HTML5. En Chrome se muestra así:

Seleccione una fecha: : ▾

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Ilustración 3.18 - Elemento input con type='date'.

- Campo para números (`type='number'`). Es una caja de texto que debe contener un número (puede tener decimales). No se permite enviar el formulario mientras el campo no tenga un valor válido. Si se usan los atributos `min` y `max` también se comprobará que el valor introducido cumpla esas condiciones.

- Campo oculto (`type='hidden'`). Se trata de una caja de texto que no se muestra al usuario. Aunque pueda parecer inútil, su uso es bastante frecuente. Es habitual que los datos introducidos por el usuario se transformen mediante *JavaScript* antes de enviarse al servidor. Un campo oculto se puede usar para almacenar los datos transformados y que se envíen como un campo más al servidor. También es posible usarlos para mandar al servidor información que no queremos que el usuario pueda ver o modificar.

El siguiente ejemplo muestra algunas de estas posibilidades.

```
<!DOCTYPE html>
<html>
<head>
<title>Formularios. Elemento input</title>
<meta charset="UTF-8">
</head>
<body>
<h1>Bienvenido al IES Clara del Rey</h1>
<h2>Formulario de Registro</h2>
<form action="procesar.php" method="get">
<h3>Datos personales</h3>
Nombre y apellido: <input required name="apellido">
<!--botones de radio, dos grupos diferentes-->
<p>Hombre<input type="radio" name="sexo" value="hombre">
Mujer<input type="radio" checked name="sexo" value="mujer">
<p>Mayor de edad<input type="radio" name="edad" value="mayor">
Menor de edad<input type="radio" checked name="edad"
value="menor"></p>
<!--checkbox-->
<p>Antiguo alumno <input name="antiguo" type="checkbox"></p>
<p>Solicita comprobación<input name="conv2" type="checkbox"></p>
<!--para seleccionar un fichero-->
<p>Adjunte fichero con la fotografía<input name="foto" type="file"
></p>
<!--para seleccionar un color-->
<p>Seleccione su color favorito<input name="color" type="color"></p>
<p>Seleccione número de hermanos o hermanas:
<!-- este campo solo admite numeros entre 0 y 99-->
<input name="hermanos" type='number' max="99" min="0"></p>
<!--botones-->
```

```

<input type="submit">
<input type="reset">
</form>
</body>
</html>

```

Ejemplo 3.34. Formulario con varios tipos (estados) de input.

El resultado en el navegador. Está abierto el cuadro de diálogo para selección de ficheros.



Ilustración 3.19 - Elemento input.

Esta tabla resume los posibles valores del atributo `type` para el elemento `input`.

Valor de <code>type</code>	Descripción
<code>text</code>	Campo de texto, es el valor por defecto
<code>password</code>	Contraseña, el texto tecleado se muestra enmascarado
<code>radio</code>	Botones de radio
<code>checkbox</code>	Casillas de verificación
<code>submit</code>	Botón para enviar el formulario
<code>reset</code>	Botón para limpiar todos los campos del formulario
<code>file</code>	Permite seleccionar uno o varios ficheros del cliente
<code>button</code>	Botón genérico

email	El campo debe contener uno o varias direcciones de correo
number	El campo debe contener un número
range	Se muestra un selector que permite escoger un valor dentro de un rango
color	Para seleccionar un color a través de una paleta de colores
hidden	Campo de texto oculto
image	Botón de envío con una imagen, que se especifica con el atributo src
date, time, url, tel	Campos de texto para introducir fecha, hora, una URL o un teléfono, respectivamente

Tabla 3.13. Estados del elemento input.

Otros atributos para el elemento `input`

La siguiente tabla muestra los atributos comunes para los diferentes tipos de `input`. No todos se pueden aplicar a todas los tipos de `input`. Algunos ya se han usado en el ejemplo anterior.

Atributo	Valores posibles	Descripción
maxlength	Un número entero	Número máximo y mínimo de caracteres que puede tener el campo
minlength	Un número	Valor máximo y mínimo
size	Un número	Tamaño con que se muestra el control
required Atributo booleano		Campo obligatorio
multiple	Atributo booleano	El usuario puede introducir más de un valor
list	El id de un datalist	Asocia el campo con un datalist
placeholder	Una cadena de texto	Texto que se muestra en el campo hasta que el usuario introduzca un valor (no es un valor inicial)
step	Un número entero	Valor en que se incrementa o decrementa el campo mediante los botones (ver ejemplo 3.34)
pattern	Una expresión regular	El valor introducido debe cumplir la expresión

Tabla 3.14. Otros atributos del elemento input.

Otros controles de usuario

Además de los diferentes tipos de elemento `input`, hay más controles de usuario disponibles. Los más importantes son:

- El elemento `select` crea una lista desplegable. Con el elemento `option` se añaden las opciones que se quieran mostrar en la lista. Se puede especificar la opción seleccionada inicialmente con el atributo `selected`.
El atributo `value` se usa para especificar el valor que recibirá el servidor si se selecciona esa opción. Si no hay atributo `value`, se pasa el texto contenido en la etiqueta `option`.
- El elemento `optgroup` es opcional. Sirve para agrupar varias opciones de una lista desplegable. Las opciones agrupadas aparecerán bajo un título común, que se especifica con el atributo `label` de `optgroup`.
- Con `datalist` se crea una lista de valores predefinidos que se puede asociar a uno o más elementos `input`. El elemento `datalist` debe tener un atributo `id` y el control asociado un atributo `list` con el valor de ese id. Cuando el usuario empieza a introducir texto, el navegador muestra las opciones que coinciden con el texto ya tecleado. Es posible seleccionar una de esas opciones en lugar de escribir todo el texto.
- El elemento `textarea` es como un campo de texto pero con más de una línea. Sus atributos más importantes son `cols` y `rows`, que indican respectivamente el número de columnas y filas.
- El elemento `button` sirve para crear un botón. El texto del botón se indica entre las etiquetas. El atributo `type` admite los valores '`submit`', '`reset`' y '`button`'. Con los dos primeros conseguimos botones para enviar y limpiar el formulario. Con el tercero es botón no hace nada, hay que añadirle funcionalidad por medio de `JavaScript`.

Este ejemplo muestra cómo usarlos:

```
<!DOCTYPE html>
<html>
<head>
  <title>Formularios. Más controles</title>
  <meta charset='UTF-8'>
</head>
<body>
  <form name="formu" action="procesar.php" method="post">
    Elemento textarea<br>
    <textarea name="texto" cols="80" rows="6"></textarea>
    <p>Lista desplegable</p>
    <select name="ciclo">
```

```

<optgroup label="Informática">
<option value="daw">Desarrollo de aplicaciones web</option>
<option value="dam">Desarrollo de aplicaciones multiplataforma</option>
<optgroup label="Administración">
<option value="ga">Gestión administrativa</option>
<option value="ci">Comercio internacional</option>
</select>
<!--saltos de linea para que se vea bien el formulario-->
<br><br><br><br>
<p>Caja de texto asociada a un <em>datalist</em></p>
<datalist id="provincias">
<option value="Palencia"/>
<option value="Valencia"/>
<option value="Madrid"/>
</datalist>
<input name="prov" list="provincias">
<br>
<!--botones-->
<button type="submit">Enviar</button>
<button type="reset">Borrar</button>
</form>
</body>
</html>

```

Ejemplo 3.35. Listas desplegables, cajas de texto y datalist.



Ilustración 3.20 , Otros controles de usuario.

3.10.3. Procesamiento de formularios

Cuando se pulsa el botón de envío. Primero el navegador comprueba si los campos del formulario cumplen con todas las restricciones, por ejemplo, el atributo `required` o los tipos de `input` que restringen el contenido (`type='email'`, `type='number'`).



Ilustración 3.21. Error al validar correo electrónico.

También es habitual validar ciertos campos usando JavaScript. Es interesante hacer todas las validaciones posibles en el cliente para evitar una comunicación adicional con el servidor.

Si pasa ambas validaciones, la información del formulario se envía a la URL especificada en el atributo `action` usando el método indicado en el atributo `method`.

En `action` se debe indicar la URL a la que se envían los datos del formulario. Suele ser un script en PHP, el lenguaje más utilizado en el lado del servidor. En `method` se especifica el método que se usa para la comunicación con el servidor. El protocolo HTTP define ocho métodos o verbos para la comunicación con el servidor: GET, POST, HEAD, PUT, CONNECT, DELETE, TRACE, OPTIONS. Se usa un método u otro en función del tipo de operación que se esté solicitando al servidor. Para solicitar una página a un servidor se usa el método GET. Para los formularios se suele usar POST, puesto que suponen un envío de información. Una diferencia al usar GET o POST es que con GET, cuando se envía el formulario, la URL muestra tanto los nombres como los valores de los controles del formulario. Por esta razón, es habitual que se use el método GET en los ejemplos sobre formularios aunque lo más apropiado sea POST.

En el ejemplo 3.33 (`login`), si cambiamos el atributo `method` por GET, al pulsar el botón de envío obtendremos un mensaje de error puesto que el fichero `procesar.php` no existe. Si nos fijamos en la barra de direcciones veremos que el final de la URL es parecido a esta

```
procesar.php?usuario=usuario&password=123123
```

Podemos observar:

- La primera parte es el nombre del fichero indicado en `action`
- A partir del símbolo '?' se especifican los campos del formulario, que se separan con el carácter '&'.
- Cada campo del formulario se envía con el formato `nombre=valor`. El nombre es el atributo `name` del control, y el valor es el introducido por el usuario.

Si probamos a llenar el formulario del ejemplo 3.34 (y cambiamos el atributo `method` por GET), la URL podría ser

```
procesar.php?apellido=Manuel-Pez&sexo=hombre&edad=mayor&antiguo=on&foto=india.JPG&color=%23000000&hermanos=5
```

Para los botones de radio, se usa como valor el campo `value` del botón de radio seleccionado

3.10.4. Formateado de formularios: atajos de teclado, orden de edición, grupos, etiquetas, etc.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Formularios: atajos de teclado, grupos, etiquetas, orden de
    edición...</title>
  </head>
  <body>
    <form action="procesar.php" method="post" >
      <fieldset>
        <!--label implícito: el control está dentro del elemento label-->
        <label>Nombre <input type="text" name="nombre" tabIndex="2"
          accesskey="n"></label>
        <!--label explícito usando el atributo for, contiene el valor del atributo
        id del control-->
        <label for="campoApellido">Apellido</label>
        <input tabIndex="1" name="apellido" autofocus accesskey="a p"
          id="campoApellido">
        <!--botones-->
        <legend>Datos personales</legend>
```

```

</fieldset>
<button tabIndex="3" type="submit">Enviar</button>
<button tabIndex="4" type="reset">Limpiar</button>
</form>
</body>
</html>

```

Ejemplo 3.36. Formulario con fieldset, legend y atajos de teclado.

En los ejemplos que hemos visto hasta ahora hemos añadido texto al formulario colocándolo al lado del control correspondiente. Es posible relacionar controles y etiquetas usando el elemento *label*. Se puede hacer de manera explícita usando el atributo *for* de la etiqueta *label* y el atributo *id* del control con el queremos asociarlo. También se puede hacer de manera implícita, situando el control dentro del elemento *label*. En el ejemplo 3.36 se muestran ambas opciones.



Ilustración 3.22 . Elementos fieldset y legend.

Con elemento *fieldset* se define un grupo de controles. Opcionalmente puede contener el elemento *legend*, que sirve para dar un nombre al grupo. Los controles que están dentro del mismo elemento *fieldset* se representan dentro de una misma caja en el formulario.

Los atajos de teclado se crean mediante el atributo *accesskey*. Es un atributo global, por lo que se puede usar en cualquier elemento HTML, no sólo en los formularios. Permite especificar una o varias letras de acceso para un elemento. Cuando el usuario pulsa una de esas teclas, el foco se sitúa sobre el elemento correspondiente.

Para establecer el orden de edición se usa el atributo *tabindex*. También es un atributo global. Por ejemplo, si el foco está situado sobre un control con *tabindex='2'*, al pulsar el tabulador el foco pasará al elemento con *tabindex='3'*.

3.11. Elementos específicos para tecnologías móviles

La aparición de móviles y tabletas ha supuesto nuevas oportunidades para los diseñadores web, pero también les ha dado bastantes quebraderos de

cabeza. Durante mucho tiempo fue necesario un desarrollo diferenciado para móviles, pero actualmente las tecnologías para móviles y ordenadores normales empiezan a converger.

3.11.1. Selección del lenguaje de marcas para tecnologías móviles

Los primeros dispositivos móviles con capacidad para acceder tenían unas características muy limitadas. Pantallas pequeñas, pocas posibilidades gráficas y capacidad de cómputo escasa. Se desarrolló un protocolo de comunicación (WAP) y lenguajes de marcas específicos para este tipo de dispositivos como Handheld Device Markup Language (HDML), Wireless Markup Language (WML), C-HTML (*Compact HyperText Markup Language*), y XHTML Mobile Profile.

Actualmente, los nuevos *smartphones* no tienen problemas con las páginas en HTML5 y por tanto no hace falta un lenguaje de marcas específico.

3.11.2. Hojas de estilo en dispositivos móviles

Con CSS3 es posible hacer hojas de estilo que se adapten a diversos tamaños de pantalla, o a la orientación de la misma. Es posible hacer que una misma página se vea de maneras diferentes en ordenador y en un móvil, por ejemplo. En la sección 4.3 se trata este tema en profundidad.

3.12. Elementos en desuso (*deprecated*)

Con la evolución del HTML muchas etiquetas han desaparecido de la especificación del lenguaje, y otras han quedado relegadas a último recurso. Se trata en general de las etiquetas relacionadas con el formato del texto, como la fuente o el color.

3.12.1. Texto parpadeante

El elemento *blink* es un elemento no estándar, nunca llegó a formar parte de ninguna de las versiones de HTML. Lo incluyó el navegador Netscape y durante algún tiempo fue soportado por otros. Siempre tuvo muy mala fama entre los diseñadores web al considerar que dificultaba la lectura de las páginas.

```
<blink>Texto parpadeante, una opción poco recomendada</blink>
```

La manera correcta para conseguir este efecto es con animaciones de JavaScript o CSS.

3.12.2. Marquesinas

Las marquesinas son texto en movimiento. Era habitual usarlas en los títulos, para que fueran más vistosos. Gustaba más a los diseñadores caseros que a los profesionales del diseño web. Como antes, si se desea animar un título u otro elemento es mejor hacerlo mediante JavaScript o CSS.

3.12.3. Alineaciones

Los atributos *align* y *valign* fueron válidos hasta la versión 4.01 Loose (incluida). Eran aplicables a muchos elementos. Por ejemplo, para centrar un párrafo horizontalmente en la pantalla.

```
<p align="center">Párrafo centrado en la pantalla</p>
```

Con *valign* se indicaba la alineación vertical del elemento dentro del elemento contenedor.

```
<p valign="top">Párrafo en la parte de abajo</p>
```

Para centrar horizontalmente elementos también existía *<center>*. En el capítulo 4 se explica cómo posicionar los elementos de una página web mediante hojas de estilo.

3.12.4. Otros elementos en desuso

Entre los elementos obsoletos que más siguen usándose podemos citar los siguientes:

- *font*, para fuentes de texto.
- atributos *background* y *background-color*.
- *applet*, para incrustar *applets* de Java en un fichero HTML.
- *frame* y *frameset*. Como se comentó en la sección 3.9, HTML5 no tiene soporte para marcos.

4. Hojas de estilo web

Contenido

- 4.1. Tipos de hojas de estilo: estáticas y dinámicas
- 4.2. Elementos y estructura de una hoja de estilo
- 4.3. Diseño de estilos para diferentes dispositivos
- 4.4. Buenas prácticas en el uso de hojas de estilo

Las hojas de estilo se ocupan de los aspectos de presentación de una página web. A medida que los elementos presentacionales han ido desapareciendo del HTML las hojas de estilo han ido ganando en importancia. A día de hoy son un elemento fundamental en el diseño web y con la última versión, CSS3, su utilidad se ha incrementado notablemente. Mediante las hojas de estilo podemos:

- Asignar fuentes de texto, colores y tamaño a los elementos de la página.
- Crear la estructura de la página: posición y tamaño de las secciones.
- Adaptar el contenido al tipo de dispositivo con el que se accede a la página: móvil, ordenador, lector de pantalla para ciegos o interfaz braille.
- Adaptar el contenido según el tamaño de la pantalla o la orientación de la misma.
- Crear animaciones, transiciones y otros efectos avanzados.

El lenguaje CSS se puede usar para dar formato a HTML, XHTML y otros lenguajes de marcas basados en XML. Define una serie de propiedades para los elementos. Por ejemplo, el color de fondo y el tamaño de la fuente son propiedades. Una hoja de estilo consiste en una serie de reglas para fijar los valores de las propiedades de los elementos de la página.

La versión 2.1 es la última recomendación completa del lenguaje. A partir de CSS3 se usa un diseño modular. El lenguaje se ha partido en varias partes o módulos, y es posible definir nuevas versiones de un módulo con independencia de los demás. Se han definido más de 50 módulos para CSS3, pero solo cuatro han llegado a convertirse en recomendaciones por ahora. El W3C se encarga del mantenimiento de la especificación del lenguaje. En este capítulo se muestran las propiedades de estilos más relevantes para los elementos HTML. Están muy lejos de ser una revisión exhaustiva, ya que hay más de 300 propiedades definidas.

4.1. Tipos de hojas de estilo: estáticas y dinámicas

La mayoría de las hojas de estilo son estáticas. Es decir, su contenido no varía. El servidor enviará la misma hoja de estilo a todos los clientes. Como iremos viendo a lo largo de este capítulo, hay mecanismos para adaptar las hojas de estilo a diversos tipos de dispositivos, así que las hojas estáticas suelen cubrir todas las necesidades.

Por hojas de estilo dinámicas se entienden aquellas que son generadas por el servidor. El servidor no devolverá siempre la misma hoja de estilo, sino que

la generará con cada petición como se hace con PHP y HTML. Su uso es muy reducido.

También se distingue entre:

- Hojas de estilo de autor: las crea el diseñador de la página web. Al hablar de hojas de estilo, en general nos referimos a estas.
- Hojas de estilo de usuario: las crea el usuario de la página, para visualizar la mejor o para eliminar elementos molestos.
- Hojas de estilo de agente de usuario (navegador): son las que usan los navegadores para mostrar las páginas. Definen, por ejemplo, cómo se mostrará el texto marcado con `strong` o `em`.

4.2. Elementos y estructura de una hoja de estilo

Las hojas de estilo están formadas por una o más reglas. Cada regla consta de un selector y un bloque de declaraciones, que va entre llaves (`{...}`). Un bloque consiste en una o más declaraciones separadas por un punto y coma (`;`). No es necesario ponerlo cuando solo hay una declaración. Cada declaración está formada por una propiedad y un valor, separados con dos puntos (`:`).

En el ejemplo 4.1 podemos ver una regla sencilla. El selector es `p`, y hay dos declaraciones. La primera asigna el valor `red` a la propiedad `color`, y la segunda el valor `xx-large` a la propiedad `font-size`. Estas reglas harán que el texto de los elementos `p` se vea en rojo y muy grande.

```
p { color: red;  
    font-size: xx-large; }
```

Ejemplo 4.1. Regla de estilo para elementos `p`.

4.2.1. Creación de hojas de estilo

Es posible incluir hojas de estilo en una página de tres maneras:

- Hojas de estilo externas.
- Hojas de estilo internas o incrustadas (*embedded*), usando el elemento `style`.
- Hojas de estilo en línea, usando el atributo `style`

Las hojas de estilo externas se crean en un fichero de texto con la extensión .css. Para incluirlos en una página web se usa el elemento `link` en la cabecera de la página.

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="ej1.css" media="all">
  </head>
...

```

Ejemplo 4.2. Hoja de estilo externa.

También es posible incluir normas de estilo directamente en el fichero HTML, como hemos visto en algunos ejemplos del capítulo 3. Se hace durante el desarrollo o para ejemplos pequeños, pero no es habitual para páginas en producción. Se puede hacer con el elemento `style` o con el atributo `style`.

El elemento va `style` dentro de la cabecera.

```
<head>
  <style>p { color: red }</style>
</head>
```

Ejemplo 4.3. Elemento style.

El atributo global `style` se usa para dar estilo a un elemento en concreto. En este caso no hace falta usar selector.

```
<p style="color: red">Texto en rojo</p>
```

Ejemplo 4.4. Atributo style.

4.2.2. Aplicación de estilos

Ya hemos visto que la primera parte de una regla es el selector. El selector se usa para determinar a qué elementos de la página afecta la regla. Los selectores más importantes son:

- Selector universal. Afecta a todos los elementos. Por ejemplo, con

```
* { color: red }
```

El texto de todos los elementos de la página se escribirá en rojo.

- Selector por nombre de elemento.

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

- Selector por atributo id. Permite aplicar las reglas a los elementos con cierto id. Para usarlo usamos el carácter '#' seguido del identificador.

```
#principal { color: red }
```

- Selector por atributo class. Selecciona los elementos con determinado valor en el atributo class. En este caso se usa el carácter '.' seguido de la clase en cuestión. Es uno de los selectores más utilizados. Por ejemplo, esta regla se aplica a todos los elementos con clase importante:

```
.importante { color: red }
```

También es posible restringir el valor a ciertos elementos. Se usan los dos selectores seguidos, el de elemento y el de la clase. La siguiente regla se aplica solo a los elementos p con clase importante.

```
p.importante { color:red }
```

Es importante no dejar espacio entre ambos selectores, porque entonces se trataría del selector descendiente.

- Selector por atributo. Podemos usarlo para seleccionar elementos que tengan un atributo con un valor determinado. El siguiente ejemplo cambia el tamaño de la fuente a los elementos que tengan atributo target con valor blank.

```
[target='_blank'] { font-size: x-large }
```

Además de buscar que valor el atributo sea una cadena en concreto, podemos buscar que contenga, empiece o termine por una subcadena.

```
/*reglas para imágenes con extensión jpg (nombre acaba en jpg)*/
img[src$='jpg']

/*reglas para imágenes cuyo nombre empieza por verano*/
img[src^='verano']

/*reglas para imágenes con la palabra 'persona' como subcadena
en el atributo alt*/
img[alt*='persona']
```

Ejemplo 4.5. Selector por atributo.

También es posible restringirlo solo a algunos elementos juntando ambos selectores. En el caso del atributo target tiene bastante sentido que la regla se aplique solo a los elementos *a*:

```
a[target='_blank'] { font-size: x-large }
```

Otra opción es usarlo para seleccionar a los elementos que tengan cierto atributo, sin importar que valor tenga.

```
/*selecciona los elemento a con atributo target*/
a[target] { font-size: x-large }
```

- Selector descendiente y selector hijo. Permiten seleccionar elementos según su situación en el árbol del documento (ver apartado 3. 2). Por ejemplo, es posible cambiar el estilo de los vínculos según estén en la barra de navegación o en otra parte de la página. Es habitual que los vínculos de la barra de navegación tengan aspecto de botones, en lugar de aparecer simplemente como texto subrayado. Podemos seleccionar solo esos vínculos así:

```
/*elementos a dentro de nav*/
nav a{...}

/*elementos a hijo de nav*/
nav>a{...}

/*elementos a dentro de footer*/
footer a{...}
```

En el ejemplo 4.6 se puede ver la diferencia entre ambos.

- Selector adyacente. Se trata de un elemento hermano que aparece justo después en documento. Esta regla afecta a los elementos *p*adyacentes a un *h1*.

```
h1 + p { font-size: x-large }
```

- Selector múltiple. Es posible usar varios selectores separados por comas. La regla se aplica a los elementos seleccionados por cualquiera de ellos.

```
/*elementos h1 y elementos con clase importante*/
h1, .importante{...}
```

En este ejemplo se muestra cómo usar algunos de estos selectores.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Ejemplo de selectores</title>
<style>
/*para elementos con clase grande*/
.grande{ font-size: x-large }
/*para elemento con id enorme*/
#enorme{ font-size: xx-large }
/*para elementos a con atributo target="_blank"*/
a[target="_blank"]{ font-size: x-large }
/*para elementos p hijos de section */
section > p{ text-decoration: underline }
/*para elementos p descendentes de section */
section p{font-size: x-large }
</style>
</head>
<body>
<p>Párrafo normal</p>
<p class="grande">Párrafo con clase grande</p>
<p id="enorme">Párrafo con id enorme</p>
<p>Vínculo en la misma ventana <a href="http://www.w3.org">W3C</a></p>
```

```
<p>Vínculo en ventana nueva <a href="http://www.w3.org" "target=_blank">W3C</a> </p>
<section>
  <p>Párrafo hijo de section</p>
  <div>
    <p>Párrafo nieto de section</p>
  </div>
</section>
</body>
</html>
```

Ejemplo 4.6. Selectores básicos.

El resultado de este ejemplo en el navegador será:



Ilustración 4.1 . Selectores básicos.

Pseudoclases y pseudoelementos

Las pseudoclases amplían las posibilidades de los selectores. En esta sección vemos las más importantes, pero iremos viendo más a lo largo del capítulo. Su nombre siempre empieza por dos puntos (:).

- **Pseudoclases estructurales.** Permiten seleccionar elementos a partir de su posición en el árbol del documento. Algunas son:
 - :first-child. Permite seleccionar a los elementos que son el primer hijo de otro. Esta regla se aplica a los párrafos que sean el primer hijo de un elemento *section*

```
section > p:first-child
```

Valdría para este caso:

```
<section>
  <p>Primer hijo</p>
</section>
```

Pero no para este otro:

```
<section>
  <h1>Encabezado</h1>
  <p>Segundo hijo</p>
</section>
```

— :nth-child(n). El hijo n-ésimo. Admite :nth-child(odd) y :nth-child(even) para diferenciar entre hijos pares e impares y también expresiones más complejas.

```
/*tercer hijo*/
:nth-child(3){ color: red }

/*hijos pares*/
:nth-child(even){ color: red }

/*hijos impares*/
:nth-child(odd){ color: red }

/*hijos 1,4,7,10*/
:nth-child(3n+1){ color: red }
```

— :first-of-type. Selecciona el primer elemento de cierto tipo. Selecciona al primer elemento *p* que aparece como hijo de un elemento *section*.

```
section > p:first-of-type
```

- Pseudoclases dinámicas. Dentro de estas se distinguen entre las pseudoclases de vínculos (*:link* y *:visited*, que veremos en detalle en la sección 4.2.3) y las de acción de usuario, que son estas tres:

- `:hover`. Se aplica cuando el usuario está pasando el ratón por encima del elemento. Para cambiar el color del texto de un párrafo cuando el ratón pasa por encima, se puede hacer

```
p:hover { font-size: x-large; }
```

- `:active`. Cuando el usuario active el elemento, por ejemplo al pulsar con el ratón en un botón o vínculo. La siguiente regla cambia el tamaño del texto del vínculo solo mientras el ratón está pulsado.

```
a:active { font-size: x-large; }
```

- `:focus`. Se activa cuando el elemento recibe el foco. Sirve para los elementos que aceptan entrada por teclado o de algún otro tipo, como los controles de un formulario. Esta regla se aplica a las cajas de texto solo cuando el usuario esté editándolas.

```
input[type='text']:focus { font-size: x-large; }
```

Los pseudoelementos permiten seleccionar elementos que no han sido definidos explícitamente en el documento. Su nombre empieza por ‘::’.

- Pseudoelemento `::first-line`. Permite seleccionar la primera linea de texto de un elemento.

```
/* primera linea de los elementos p */  
p::first-line { font-size: x-large; }
```

- Pseudoelemento `::first-letter`: Similar al anterior, pero solo para la primera letra.

```
/* primera letra de los elementos p */  
p::first-letter { font-size: x-large; }
```

Tipo de selector	Ejemplo	Descripción
Universal	<code>* { color: red; }</code>	Todos los elementos
Por etiqueta	<code>p { color: red; }</code>	Todos los elementos p
Por id	<code>#principal { color: red; }</code>	Elemento con id="principal"
Por clase	<code>.importante { color: red; }</code>	Elementos con class="importante"

Selector múltiple p, h1, .importante { color: red }	Elementos p, h1 o con class="importante"
Selector hijo	section>p {color: red}
Selector descendiente	section p {color: red}
Selector adyacente	header ~ section
Selector de atributos	a[target="_blank"]

Tabla 4.1. Selectores básicos.

4.2.3. Herencia de estilos y aplicación en cascada

La herencia y la aplicación de estilos en cascada son conceptos básicos en el diseño web con CSS. Para muchas propiedades, el valor por defecto es el que tenga el elemento padre, de ahí el nombre de herencia. Por ejemplo, si fijamos el color del texto para el elemento html

```
html { color: red }
```

todos los elementos contenidos dentro del mismo (es decir, toda la página) heredarán esa propiedad. También es posible forzar que se herede un atributo en los atributos que por defecto no se heredan usando el valor *inherit* (se puede aplicar a todas las propiedades CSS):

```
p { display: inherit }
```

Aplicación en cascada

Cuando varias reglas afectan al mismo elemento se aplican una serie de normas de prioridad. La expresión "en cascada" hace referencia a la manera en que se aplican las reglas cuando hay varias posibilidades. En general, la regla más específica prevalece. Por ejemplo, si tenemos las reglas

```
p { color: blue }
#enrojo { color: red }
```

El elemento

```
<p id="enrojo">Prioridad de las reglas</p>
```

es seleccionado por ambas. En este caso se mostrará en rojo, pero a veces puede ser difícil saber qué regla prevalecerá.

Por defecto, las reglas definidas en una hoja de estilo de autor tienen preferencia sobre las de usuario, y estas a su vez sobre las del navegador. Para alterar este orden es posible marcar las declaraciones de las reglas de estilo como importantes. Entre reglas importantes, se invierte el orden.

```
/*esta regla se aplicará frente a otras más específicas*/
#enrojo {color:red !important}
```

El orden de prioridad es:

- Declaraciones importantes del navegador.
- Declaraciones importantes del usuario.
- Declaraciones importantes del autor.
- Declaraciones normales del autor.
- Declaraciones normales del usuario.
- Declaraciones normales del navegador.

Dentro de las hojas de estilo de autor, las reglas en línea (atributo `style`) tienen prioridad sobre las hojas de estilo internas (elemento `style`), y estas a su vez sobre las externas. Para ordenar las reglas de cada categoría las reglas se ordenan de más a menos específicas. Se asigna a cada regla un número según los selectores que contenga. Para cada regla, hay que contar:

- El número de selectores por id. Llamaremos a ese número A.
- El número de selectores por clase, por atributos y por pseudoclase. Llamaremos a ese número B.
- El número de selectores por tipo elemento y por pseudoelemento. Llamaremos a ese número C.

La prioridad de la regla (*specificity*) se obtiene concatenando los números A, B y C. Esto implica que las reglas con selector por id tienen preferencia respecto a los que no la tienen. La siguiente tabla muestra algunos ejemplos.

Regla	Prioridad
p {color: blue;}	001
#enrojo {color: red;}	100
important	010

p.important	011
input[type='text']:focus { font-size:x-large; }	021
p::first-letter { font-size:x-large; }	002

Tabla 4.2. Prioridad de las reglas.

Por último, si tras aplicar todos los criterios dos reglas tienen la misma prioridad, se aplica la última en declararse.

4.2.4. Formateado de páginas mediante estilos

En esta sección se explican las propiedades de estilo más habituales. Comenzaremos viendo algunas generales y luego veremos las propiedades específicas de los elementos HTML más habituales, como tablas y listas.

Unidades de medida

Muchas propiedades se refieren al tamaño de los elementos. En CSS es posible usar varias unidades de medida. Las más comunes son:

- **Píxeles.** La pantalla del ordenador está dividida en píxeles. El número de píxeles depende de la resolución de la pantalla y por tanto un diseño hecho en píxeles puede variar mucho al verlo en dos dispositivos diferentes. Por este motivo, es mejor no usarlo para especificar el tamaño de las fuentes o de las diversas secciones de la página. Se suele usar especificar el grosor de los bordes, o las sombras de los elementos.
- **Unidad *em*.** Una unidad *em* equivale a la anchura de la letra m en la fuente actual del documento. Es una unidad muy útil para especificar tamaños de las fuentes. Se adapta a la fuente que esté usando el usuario y mantiene la proporción de tamaño entre diferentes elementos.

Esta unidad tiene una particularidad. El valor se interpreta en relación al del elemento contenedor, como se puede ver en este ejemplo. La propiedad *font-size* establece el tamaño de la fuente.

```
<section style="font-size : 2em">
  Sección
  <p style="font-size : 2em">Hola</p>
</section>
```

Ejemplo 4.7. Unidades *em*.

Para el elemento `section` el tamaño de la fuente será 2 em, pero para el elemento `p` será $2 * 2 \text{ em} = 4 \text{ em}$. El resultado en el navegador será



Ilustración 4.2 . Unidades em.

Para que el párrafo tuviera el mismo tamaño de letra que el elemento `section` bastaría con quitar el atributo `style` del elemento `p`. En ese caso, `p` heredaría el tamaño de letra de `section`.

- **Porcentajes.** Permite expresar el tamaño de un elemento en función del de su elemento contenedor. Es la unidad más apropiada para dar tamaño a las secciones de la página si buscamos un diseño adaptable.

```
/*tamaño en pixeles*/
header { width : 100px;
          height : 60px; }

/*unidades em*/
footer { width : 5em;
          height : 3em; }

/*porcentajes*/
section { width : 80%;
          height : 20%; }
```

Ejemplo 4.3. Unidades de medida.

Además de las tres unidades citadas, es posible usar otras. Las llamadas unidades absolutas se recomiendan solo en el diseño para impresión:

- **Puntos (pt)** y **picas (pc)**. Unidades habituales en tipografía para el tamaño de la fuente.
- **Pulgadas (in)**, **centímetros (cm)**, **milímetros (mm)**.

Propiedad display.

Como ya hemos comentado anteriormente, los elementos de HTML pueden ser de linea o de bloque. Con la propiedad `display` se elige cómo se comportará el elemento.

```
/*los vínculos dentro del elemento nav serán de bloque*/
nav a{ display : block }

/*los elementos p serán de linea*/
p { display : inline }

/*el elemento con id oculto no se mostrará*/
#oculto { display : none }
```

Ejemplo 4.9. Propiedad display.

Estilo para las fuentes

La propiedad más usada es `font-family`, para especificar el tipo de letra de la página. Podemos usar el nombre de una familia de fuentes, como *Helvetica* o *Verdana*, o un nombre de familia genérica (hay cinco: *serif*, *sans-serif*, *cursive*, *fantasy* y *monospace*). Como no todas las fuentes están disponibles en todos los sistemas, se recomienda especificar varias.

```
p{ font-family: "Times New Roman" sans-serif }
```

El navegador usará la primera disponible de la lista. Es buena idea poner como último elemento el nombre una familia genérica, para asegurarse de que el navegador tiene alguna de las indicadas.

La propiedad `font-size` permite establecer el tamaño de la fuente. Hay varias opciones:

- Mediante cualquiera de las unidades válidas en CSS. Lo más habitual es usar em.
- Tamaños predefinidos
 - Absolutos: xx-small, x-small, small, medium, large, x-large, xx-large
 - Relativos: smaller, larger. La fuente será menor o mayor que la del elemento padre.

```
p{ font-size: 4em; }

/*elementos con class='enGrande'*/
.enGrande{ font-size: xx-large }
```

La propiedad `font-weight` establece el grosor de la fuente. Se puede usar un número de 100 a 900 o una de las siguientes palabras: `normal`, `bola` (negrita), `bolder` (más grueso que negrita) o `lighter` (menos que normal). Un valor de 400 equivale a texto normal, y un valor de 700 a texto en negrita.

```
p{ font-weight bold }
/*elementos con class='muyNegrita'*/
.muyNegrita{ font-weight: 900 }
```

Propiedad	Valores posibles	Descripción
<code>font-size</code> Unidad CSS tamaño predefinido		Tamaño de la fuente
<code>font-weight</code>	100, 200, 300, 400, 500, 600, 700, 800, 900, normal, bold, bolder, lighter,	Grosor de la fuente
<code>font-family</code>	Nombre de familia o familia genérica	Familia de la fuente
<code>font-style</code>	<code>italic</code> <code>oblique</code> <code>normal</code>	Muestra el texto en cursiva (<code>italic</code> y <code>oblique</code>)
<code>font-variant</code>	<code>normal</code> <code>smallcaps</code>	Muestra el texto en mayúsculas pequeñas
<code>font</code>	<font-style> <font-variant> <font-weight> <font-size> <font-family>	Propiedad resumen

Tabla 4.3. Prioridades de estilo para las fuentes.

La propiedad `font-variant` solo ofrece el valor `small-caps`, que sirve para mostrar el texto en mayúsculas pequeñas. Algunos diseñadores usan este efecto en las primeras líneas o palabras de un párrafo. Con `font-style` es posible hacer que el texto se muestre en cursiva con cualquiera de los dos valores posibles, `italic` y `oblique`.

```
p:first-line{ font-variant: small-caps }
/*elementos con class='cursiva'*/
.cursiva{ font-weight: 900 }
```

Por último, con la propiedad resumen `font` es posible especificar todas las propiedades a la vez. Estas propiedades resumen son habituales en CSS, veremos más en las siguientes secciones.

```
p{ font: bold 4em "comic"}
```

Estilo para el texto

Además de la fuente, hay muchas propiedades relativas al texto. La propiedad `color` indica el color del texto. Se puede especificar el color con cualquiera de los formatos que vimos en el apartado 3.3.

```
header{ color: lime }  
section{ color: #0000ff }
```

La propiedad `text-align` sirve para posicionar horizontalmente un texto. Admite los valores `left` (texto justificado a la izquierda), `right` (a la derecha), `justify` (izquierda y derecha) y `center` (centrado).

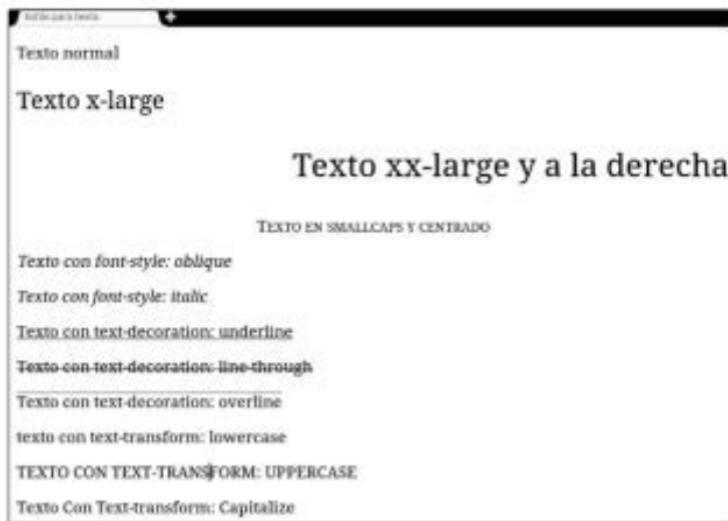


Ilustración 4.3. Propiedades para texto y fuente.

Con la propiedad `white-space` es posible determinar cómo se tratarán los espacios y los saltos de línea. Los valores posibles son:

- * `normal`: el modo por defecto. Los espacios y saltos de línea se colapsan.
Introduce saltos de línea para adaptar el texto al elemento contenido.

- *white-space*: equivalente al elemento *pre*. El texto se mostrará tal cual aparece en el documento, preservando espacios en blanco y saltos de línea. No adapta el texto al elemento contenedor.
- *nowrap*: como normal pero no rompe las líneas aunque desborden el elemento contenedor.
- *pre-wrap*: preserva los espacios y los saltos de línea. Introduce saltos de línea para adaptar el texto al elemento contenedor.
- *pre-line*: colapsa los espacios en blanco pero preserva saltos de línea. Introduce saltos de línea para adaptar el texto al elemento contenedor.



Ilustración 4.4. El texto puede adaptarse o no a su elemento contenedor.

La propiedad *text-decoration* ofrece tres posibilidades para decorar el texto: subrayarlo, tacharlo y añadir una linea por encima del texto (Ilustración 4.3).

```
/*subrayado*/
p { text-decoration: underline }

/*tachado*/
p { text-decoration: line-through }

/*línea por encima*/
p { text-decoration: overline }
```

Ejemplo 4.10. Propiedad *text-decoration*

La propiedad *text-transform* cambia las mayúsculas y minúsculas del texto. Es posible poner el texto en mayúsculas (*uppercase*), minúsculas (*lowercase*) o mayúsculas para la primera letra de cada palabra (*capitalize*).

```
/*muestra el texto subrayado*/
p { text-decoration: underline }
```

Propiedad	Valores posibles	Descripción
text-align	right, left, center, justify	Posición horizontal del texto
white-space	normal, pre, pre-line, nowrap, pre-wrap	Cómo tratar espacios en blanco y saltos de linea
text-decoration	underline, overline, line-through, none	Decoración para el texto
text-transform	uppercase, lowercase, capitalize, none	Muestra el texto en mayúsculas/minúsculas
text-indent	Una longitud (normalmente em) Indentado en la primera linea de texto	
word-spacing	Una longitud (normalmente em)	Espacio entre palabras
color	Un color (cualquier formato valido)	Color del texto

Tabla 4.4. Propiedades para el texto.

Estilo para el fondo

La propiedad *background-color* se usa para establecer el color de fondo de un elemento. Usaremos cualquiera de las formas válida para representar el color (ver apartado 3.3).

```
header{ background-color: lime }
section{ background-color: #0000ff }
```

Ejemplo 4.11. Color de fondo.

Con la propiedad *background-image* es posible usar una imagen como fondo del elemento. Si se usa una imagen, hay otras tres propiedades relevantes.

La propiedad *background-position* se utiliza para situar la imagen en relación al fondo. Normalmente, se utilizan dos valores, el primero para especificar la posición horizontal (*right center*, *left*) y el segundo para la vertical (*top*, *center*, *bottom*). Si solo se indica un valor, se asume que el segundo es *center*. Además de estos valores también es posible usar otras unidades CSS. En el ejemplo 4.12 se muestran algunas posibilidades.

La propiedad *background-repeat* establece como repetir la imagen si esta es más pequeña que el fondo. Se puede elegir entre no repetir, repetir horizontalmente, verticalmente o en ambos sentidos. Con la propiedad *background-attachment* podemos fijar la imagen de fondo en su posición aunque el resto de la página se desplace.

Propiedad	Valores posibles	Descripción
background-color	Cualquier formato de color	Color de fondo
background-image	Una URL, absoluta o relativa La	URL de la imagen
background-position pos	X pos Y posX: top, bottom, center posY: top, bottom, center	La posición de la imagen
background-repeat	no-repeat, repeat-x, repeat-y, repeat	Sí se repite la imagen y cómo
background-attachment scroll (se mueve con la página, por defecto) fixed (fija)		Sí la imagen se desplaza con el resto de la página
background	<color> <background-image> <background-repeat> <background-attachment> <background-position>	Propiedad resumen

Tabla 4.5. Propiedades para el fondo.

La propiedad resumen *background* incluye todas las propiedades relacionadas con el fondo.

```
#imagenAbajo { background: limeurl('pezverde.png') no-repeat bottomright }
```

En el siguiente ejemplo se muestran varias opciones de fondo. Hay varios elementos *div*, cada uno con propiedades diferentes.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='UTF-8'>
    <title>Estilo para fondo</title>
    <link rel="stylesheet" href="estiloFondo.css" media="all">
  </head>
  <body>
    <div id="color"></div>
    <div id="imagenCentrada"></div>
    <div id="imagenRepetida"></div>
    <div id="imagenHorizontal"></div>
    <div id="imagenVertical"></div>
    <div id="imagenAbajo"></div>
  </body>
</html>
```

Ejemplo 4.12. HTML para el ejemplo de estilo para el fondo.

El fichero estiloFondo.css es el siguiente.

```
div{  
    width:100px;  
    height:100px;  
    margin: 2em;  
    background-color:white;  
    float:left; }  
body{ background-color: lightblue }  
#color{ background-color: lime }  
#imagenCentrada{  
    background-image: url('pezverde.png');  
    background-position: center center;  
    background-repeat:no-repeat; }  
#imagenRepetida{ background-image: url('pezverde.png') }  
#imagenHorizontal{  
    background-image: url('pezverde.png');  
    background-position: bottom;  
    background-repeat:repeat-x; }  
#imagenVertical{  
    background-image: url('pezverde.png');  
    background-position: top center;  
    background-repeat:repeat-y; }  
#imagenAbajo{ background: lime url('pezverde.png') no-repeat bottom  
right }
```

Ejemplo 4.13. Estilo para el fondo.

Para colocar los `div` utiliza las propiedades `margin` y `float`, que veremos en la sección 2.4.5. El resultado es:



Ilustración 4.5. Estilo para el fondo.

Un efecto muy habitual para el fondo el gradiente o degradado de color. Antes se conseguía por medio de imágenes o JavaScript, pero también es posible hacerlo con CSS3. Hay varios tipos de gradiente (vertical, horizontal, diagonal, radial) y la mayoría funcionan en las últimas versiones de los navegadores.

```
/*de arriba a abajo*/
background: linear-gradient(yellow, white);
/*de izquierda a derecha*/
background: linear-gradient(to right, yellow, blue);
/*de la esquina superior izquierda a la inferior derecha*/
background: linear-gradient(to bottom right, yellow, blue);
/*pasando por más de un color*/
background: linear-gradient(yellow, green, blue);
```

Ejemplo 4.14. Gradientes.

Estilo para vínculos

Es habitual que los vínculos se muestren de manera diferente según hayan sido visitados o no. Podemos diferenciar las reglas usando las pseudoclases `:link` para los vínculos no visitados y `:visited` para los visitados. Actualmente la funcionalidad de `:visited` está restringida en varios navegadores importantes (Firefox y Chrome entre otros) por cuestiones de seguridad y solo es posible modificar algunas propiedades.

En el ejemplo 4.15 se utilizan estas pseudoclases:

```
<!DOCTYPE html>
<html>
<head>
<meta charset='UTF-8'>
<title>Estilo para vínculos</title>
<style>
/*para mostrar uno debajo de otro*/
a{ display: block }
/*no visitados*/
a:link{ background-color: lightblue }
/*visitados*/
a:visited{ background-color: white }
/* al pasar el ratón por encima*/
a:hover{ font-size: 2em;
```

```

text-decoration: none; }

</style>
</head>
<body>
<a href="http://www.w3.org">W3C (sin visitar)<a>
<a href="http://www.w3.org/TR">W3C (visitado)<a>
<a href="http://www.w3.org">W3C (hover)<a>
</body>
</html>

```

Ejemplo 4.15. Pseudoclases para vínculos.

Si el primer vínculo no está visitado pero el segundo sí, el resultado en el navegador será:

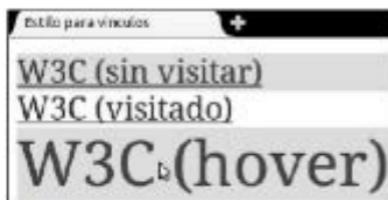


Ilustración 4.6- Estilo para vínculos.

Estilo para listas.

Propiedad	Valores posibles	Descripción
list-style-image	Una URL, relativa o absoluta La URL de la imagen que se usará como marcador	
list-style-position	inside outside (valor por defecto)	Posición del marcador
list-style-type	No ordenadas: disc, square, circle Ordenadas: decimal, lower-roman, lower-alpha...	Tipo de marcador
list-style	<list-style-type> <list-style-position> <list-style-image>	Propiedad resumen

Tabla 4.6. Resumen de propiedades para listas.

Hay tres propiedades específicas para las listas, y una propiedad resumen.

La propiedad *list-style-image* sirve para poner una imagen como marcador de la lista.

```
li { list-style-image: url("imagen.jpg") }
```

La propiedad *list-style-type* permite elegir el estilo del marcador entre muchas opciones disponibles.

- Marcadores sin orden: *square*, *disc*, *circle*.
- Marcadores numéricos: *decimal*, *decimal-leading-zero*, *lower-roman* (número romanos en minúscula), *upper-roman* (número romanos en mayúscula).
- Marcadores alfabéticos: *lower-greek*, *lower-latin*, *upper-latin*, *armenian*, *georgian*, *lower-alpha*, *upper-alpha*.

La propiedad *list-style-position* indica la posición del marcador respecto al texto. Puede tomar los valores *inside* o *outside* (por defecto).

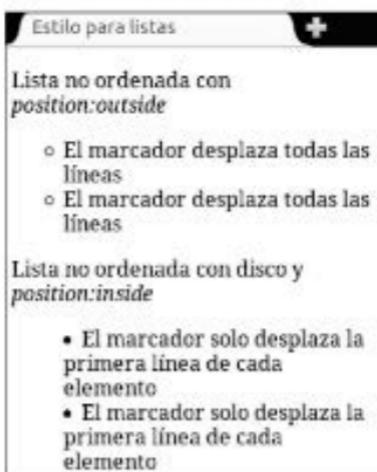


Ilustración 4.7. Propiedad list-style-position.

También existe una propiedad resumen, *list-style*, que incluye las tres propiedades anteriores. Si se especifican imagen y marcador, prevalece la imagen. Si la imagen no está disponible, muestra el marcador.

```

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Estilo para listas</title>
<style>
#conPez{ list-style-image: url("pezverde.png") }
#conCirculo{ list-style-type: circle }
#katakana{ list-style-type: katakana }
#decimal{ list-style-type: decimal-leading-zero; }
#conResumen{ list-style: square inside url("noDisponible.png") }
</style>
</head>
<body>
<p>Lista no ordenada con imagen</p>
<ul id="conPez">
<li>Patatas</li>
<li>Peras</li>
</ul>
<p>Lista no ordenada con circulo</p>
<ul id="conCirculo">
<li>Patatas</li>
<li>Peras</li>
</ul>
<p>Listas ordenadas. Marcador decimal con ceros por delante</p>
<ol id="decimal">
<li>Peras</li>
<li>Manzanas</li>
</ol>
<p>Katakana (japonés)</p>
<ol id="katakana" reversed>
<li>Peras</li>
<li>Manzanas</li>
</ol>
<p>Lista con propiedad resumen. Si la imagen no está disponible  
mostrará un cuadrado como marcador</p>
<ul id="conResumen">
<li>Patatas</li>
<li>Peras</li>
</ul>
</body>
</html>

```

Ejemplo 4.16. Estilo para listas.

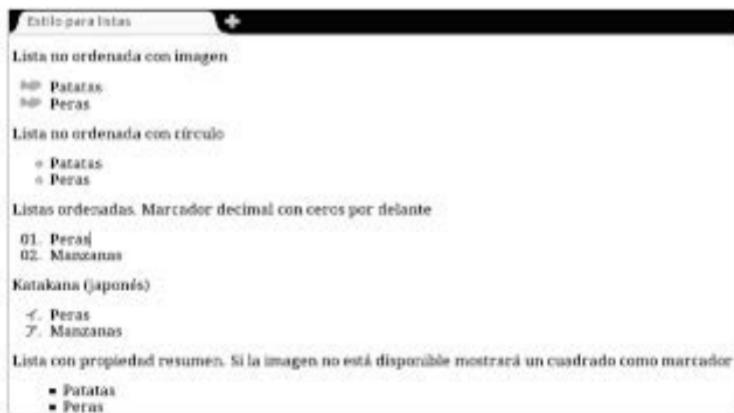


Ilustración 4.8. Estilo para listas. Pantallazo del ejemplo 4.16.

Estilo para tablas

Partiremos de una tabla base e iremos aplicando diferentes reglas de estilo.

```
<table>
  <caption>Tabla de películas y directores</caption>
  <tr>
    <th>Película</th>
    <th>Director</th>
    <th>País</th>
    <th>Año</th>
  </tr>
  <tr>
    <td>Los otros</td>
    <td>Alejandro Amenábar</td>
    <td>España</td>
    <td>2001</td>
  </tr>
  <tr>
    <td>Los pájaros</td>
    <td>Alfred Hitchcock</td>
    <td>USA</td>
    <td>1963</td>
  </tr>
  <tr>
    <td>Amanece, que no es poco</td>
```

```
<td>José Luis Cuerda</td>
<td>España</td>
<td>1988</td>
</tr>
<table>
```

Ejemplo 4.17. Tabla básica para los ejemplo de estilo.

Las tablas suelen mostrarse con algunos bordes.

```
table, td, th { border: black solid 1px; }
```

Ejemplo 4.18. Estilo para bordes básicos.

Las propiedades relacionadas con los bordes se tratan en detalle en la sección 4.2.5, al describir el modelo de cajas.

Tabla de películas y directores				
Película	Director	País	Año	
Los otros	Alejandro Amenábar	España	2001	
Los pájaros	Alfred Hitchcock	USA	1963	
Amanece, que no es poco	José Luis Cuerda	España	1988	

Ilustración 4.9. Tabla con bordes.

Para mejorar el aspecto de la tabla, es habitual que los bordes colapsen usando la propiedad `border-collapse: collapse;`. En el siguiente ejemplo, además hay reglas para fijar el ancho de las columnas, centrar el texto y añadir un espacio de relleno dentro de las celdas.

```
table, td, th { border: black solid 1px;
    border-collapse: collapse; }
/*como hay cuatro columnas, cada parte tendrá de ancho la cuarta parte
que la tabla*/
td, th{ width: 25%; }
/*un poco de relleno para que se lea mejor el texto*/
padding: 1em;
text-align:center; }
```

Ejemplo 4.19. Estilo mejorado para tablas.

Tabla con bordes colapsados, padding y text-align.			
Película	Director	País	Año
Los otros	Alejandro Amenábar	España	2001
Los pájaros	Alfred Hitchcock	USA	1963
Anoche, que no es poco	José Luis Cuerda	España	1988

Ilustración 4.10. Tabla con bordes colapsados, padding y text-align.

Finalmente, podemos dar un color de fondo alternante, que mejora la legibilidad y cambiar la posición del elemento `caption` añadiendo estas reglas al ejemplo anterior:

```
table { caption-side : bottom }
tr:nth-child(odd) { background-color: yellow }
tr:nth-child(even) { background-color: lightblue }
```

Ejemplo 4.10. Estilo para fondo alternante.

El resultado será

Tabla con text-align, and...			
Película	Director	País	Año
Los otros	Alejandro Amenábar	España	2001
Los pájaros	Alfred Hitchcock	USA	1963
Anoche, que no es poco	José Luis Cuerda	España	1988

Ilustración 4.11. Tabla con fondo alternante y caption abajo.

4.2.5. Estructura de páginas mediante estilos

El modelo de cajas

El modelo de cajas define cómo se muestran los elementos en la pantalla. Cada elemento HTML, ya sea de línea o de bloque, se representa dentro de una caja. El aspecto básico de una caja viene definido por su anchura, altura, margen, relleno, borde y margen. Las propiedades `width` y `height` son el ancho y el alto de la caja, respectivamente.

El relleno (*padding*) es el espacio que va desde el contenido hasta el borde de la caja. Tiene el mismo fondo que la caja. Se puede distinguir entre la parte superior (*padding-top*), inferior (*padding-bottom*), izquierda (*padding-left*) y derecha (*padding-right*). También hay una propiedad resumen, *padding*:

```
header { padding: 1em; }
section { padding-right: 0.5em;
           padding-left: 1em;
           padding-bottom: 1.5em;
           padding-top: 2em; }
```

Ejemplo 4.21. Propiedad *padding*.

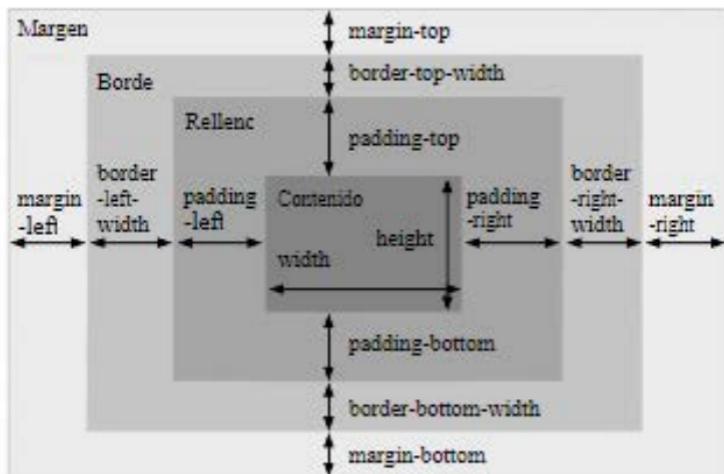


Ilustración 4.12. Modelo de cajas.

El borde rodea la caja y el relleno. Es posible cambiar tamaño, color y estilo, además de distinguir entre los cuatro lados. Para especificar el ancho del borde tenemos las propiedades `border-top-width`, `border-bottom-width`, `border-left-width`, `border-right-width` y la propiedad resumen `border-width`. Se puede escoger entre varios estilos de borde con `border-top-style`, `border-bottom-style`, `border-left-style`, `border-right-style` y la propiedad resumen `border-style`.

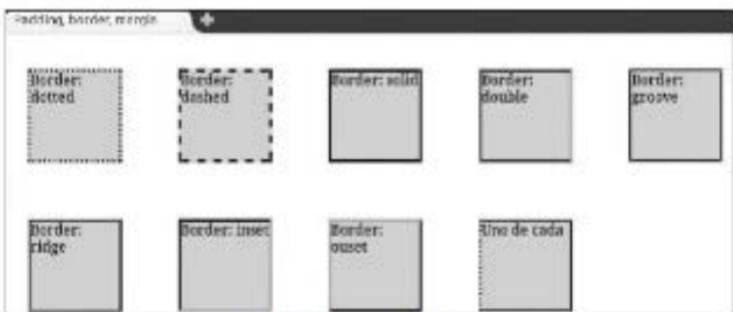


Ilustración 4.13. Estilos de borde.

Para el color, `border-top-color`, `border-bottom-color`, `border-left-color`, `border-right-color` y la propiedad resumen `border-color`. Existe también una propiedad resumen `border` que resume todas las anteriores.

```
nav { border: 1px dotted black }
```

Ejemplo 4.22. Propiedad border

El margen es el espacio que deja la caja con los elementos adyacentes. En principio el margen es transparente y el fondo será el del elemento contenedor de la caja. También se puede distinguir entre los cuatro lados.

```
header { margin: 1em }
section { margin-right: 0.5em;
           margin-left: 1em;
           margin-bottom: 1.5em;
           margin-top: 2em; }
```

Ejemplo 4.23. Propiedad margin

El tamaño de relleno, borde y margen se añaden al ancho y al alto de la caja.

Estilo avanzado para las cajas

Bordes redondeados. Hay una propiedad para cada una de las esquinas y una propiedad resumen: `border-top-left-radius`, `border-top-right-radius`, `border-`

`bottom-right-radius`, `border-bottom-left-radius` y `border-radius`. Cuanto mayor sea el radio, más redondeada será la esquina. Se puede ver el efecto en la ilustración 4.19.

```
/*todas las esquinas igual*/
header { border-radius: 1em }
/*esquinas con radio diferente*/
section { border-radius: 1em 2em 0.5em 1em }
/*solo las esquinas superiores*/
footer { border-top-left-radius: 1em;
          border-top-right-radius: 1em }
```

Ejemplo 4.24. Propiedad border-radius (esquinas redondeadas).

- Sombras. Se puede añadir sombra a las cajas con la propiedad `box-shadow`. La sintaxis es la siguiente.

```
box-shadow: <distancia> <distanciay> <difuminado> <tamaño>
<color> <inset>
```

Solo los dos primeros valores son obligatorios. El valor `inset` sirve para que la sombra esté dentro del elemento en vez de fuera.

```
section { box-shadow: 10px 10px }
header { box-shadow: 10px 10px 5px black }
footer { box-shadow: 10px 10px inset }
```

Ejemplo 4.25. Propiedad box-shadow.

Se puede ver el efecto en la ilustración 4.19.

Modos de posicionamiento

La posición que ocupa un elemento en la pantalla depende, entre otras cosas, de su modo de posicionamiento. Hay cinco: estático, relativo, absoluto, fijo y flotante. El modo de posicionamiento se elige con las propiedades `position` y `float`. Según el modo de posicionamiento que usemos, también pueden ser útiles las propiedades `right`, `left`, `top` y `bottom`.

El *posicionamiento estático o normal* es el modo por defecto. El navegador va mostrando los elementos en el orden en que aparecen en el fichero. Cuando se trata de un elemento de bloque se introduce un salto de línea antes y después del mismo. Los elementos de línea se muestran uno al lado del otro mientras quepan en el elemento contenedor. En este modo de posicionamiento las propiedades *top*, *bottom*, *right* y *left* se ignoran.

Para los ejemplos sobre los modos de posicionamiento estático, relativo y absoluto usaremos esta sencilla página web básica.

```
<!DOCTYPE html>
<html>
<head>
    <title>Posicionamiento estático</title>
    <meta charset='UTF-8'>
    <style>
        header, footer, section { padding: 3em;
            height: 3em;
            border: 1px solid black; }
        header, footer { background-color: lightgreen; }
        section { background-color: lightblue; }
    </style>
</head>
<body>
    <header>Cabeza</header>
    <section>Contenido principal</section>
    <footer>Pie de página</footer>
</body>
</html>
```

Ejemplo 4.16. HTML básico para los ejemplos de posicionamiento.

Se puede ver el resultado en la ilustración 4.19.

El *posicionamiento relativo* permite desplazar a los elementos en relación a la posición que tendrían en el modo estático. El desplazamiento se indica con las propiedades *right*, *left*, *top* y *bottom*. Los demás elementos ignoran este desplazamiento y se posicionan como si el elemento estuviera en la posición normal, así que se pueden producir solapamientos.

Por ejemplo, la regla

```
section { position: relative;  
background-color: lightblue;  
top: 2em;  
left: 3em; }
```

Ejemplo 4.27. Posicionamiento relativo.

posiciona el elemento con id cabecera 2em más abajo y 3em más a la derecha de lo que estaría con posicionamiento normal. En la siguiente ilustración se comparan los dos estilos posicionamiento.



Ilustración 4.14. Posicionamientos estático y relativo.

Con el modo absoluto se especifica la posición del elemento en relación a otro elemento. En general, es en relación al primer antecesor con posicionamiento no estático. En el ejemplo que sigue, es en relación al elemento `body`. Si cambiamos la regla del ejemplo anterior por esta

```
section { background-color: lightblue;  
position: absolute;  
top: 16em;  
left: 3em; }
```

Ejemplo 4.28. Posicionamiento absoluto.

el resultado será:

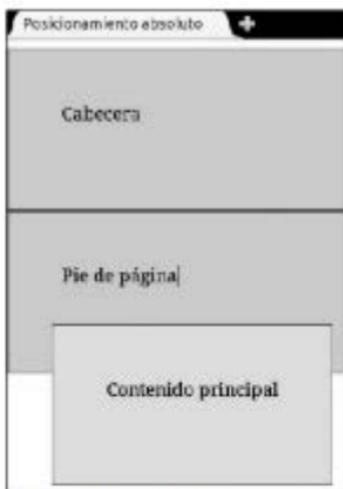


Ilustración 4.15. Posicionamientos absolutos.

Los demás elementos se posicionan como si los elementos con posicionamiento absoluto no existieran, así que pueden darse solapamientos. Además, se puede observar que el ancho del elemento cambia, no se ajusta para ocupar todo el espacio disponible como hacen los elementos de bloque en el modo normal o el relativo.

El *posicionamiento fijo* sirve para dejar un elemento fijo en la pantalla, aunque haya desplazamiento vertical u horizontal. La posición se especifica como en el absoluto.

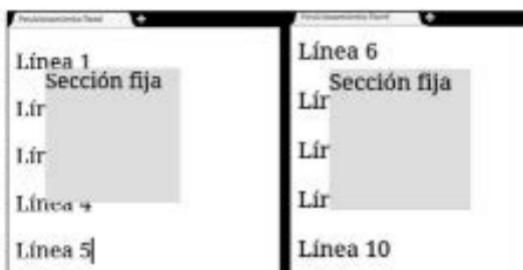


Ilustración 4.16. Posicionamiento fixed.

El último tipo de posicionamiento es el *flotante*. En este caso se usa la propiedad *float*. Los valores posibles son *right*, *left* y *none*. Los elementos flotantes se desplazan hacia la izquierda o la derecha. El resto de elementos se sitúan alrededor de ellos a no ser que se utilice la propiedad *clear*.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Posicionamiento flotante</title>
        <meta charset='UTF-8'>
        <style>
            section { height: 6em; padding: 1em; border: 1px solid black; }
            div { height: 3em; background-color: lime; }
            flotIzq { float: left; }
            flotDer { float: right; }
            .conClear { clear: both; }
        </style>
    </head>
    <body>
        <section>
            <div class="flotIzq"> Elemento flotado a la izquierda</div>
            <p>Texto alrededor del elemento flotado </p>
        </section>
        <section>
            <div class="flotDer"> Elemento flotado a la derecha</div>
            <p>Texto alrededor del elemento flotado</p>
        </section>
        <section>
            <div class="flotIzq"> Elemento flotado a la izquierda</div>
            <div class="flotDer"> Elemento flotado a la derecha</div>
            <p>Texto alrededor de los elementos flotados</p>
        </section>
        <section>
            <div class="flotIzq"> Elemento flotado a la izquierda</div>
            <p class="conClear">Texto con clear</p>
        </section>
    </body>
</html>
```

Ejemplo 4.29. Posicionamiento flotante.

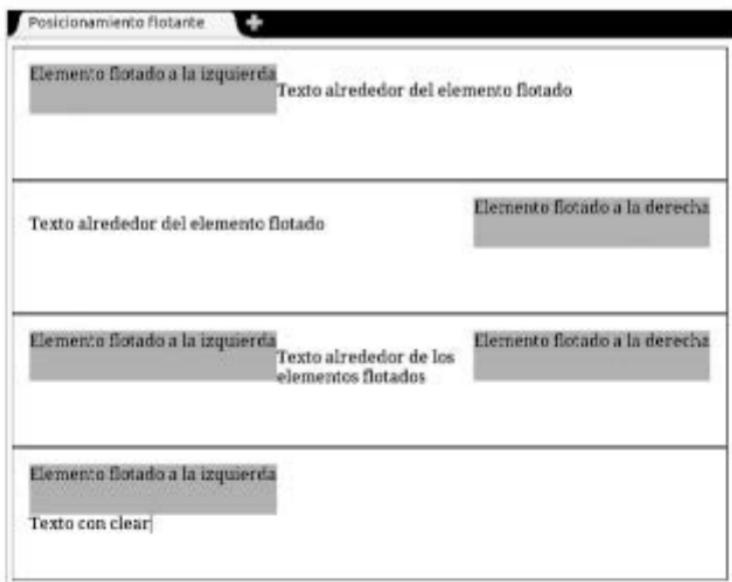


Ilustración 4.17. Posicionamiento flotante.

Si hay varios elementos flotantes seguidos, se colocarán uno detrás de otro mientras haya espacio en el elemento contenedor.

```
<!DOCTYPE html>
<html>
<head>
<title>Posicionamiento flotante</title>
<meta charset='UTF-8'>
<style>
section { float:left;
            height: 6em;
            width: 4em;
            padding: 1em;
            border: 1px solid black; }

</style>
</head>
<body>
```

```
<section><p>Elemento flotado a la izquierda</p></section>
</body>
</html>
```

Ejemplo 4.30. Varios elementos flotantes seguidos.

Ajustando la ventana del navegador podría verse así:



Ilustración 4.18. Varios elementos flotantes seguidos.

Ejemplo: estructura de página con dos columnas

Finalizamos con un ejemplo de estructura bastante habitual usando posicionamiento estático y flotante. Para la anchura de los elementos se han utilizado porcentajes, así que se adapta bastante bien a diferentes tamaños de pantallas. La altura está fija en píxeles porque es un ejemplo sin contenido, solo contiene la estructura básica de la página.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Estructura de una página</title>
    <link rel="stylesheet" href="posicionamiento.css" />
  </head>
```

```
<body>
<header><h1>Página con CSS</h1></header>
<nav>
<a href="http://www.w3c.org">Enlace 1</a>
<a href="http://www.w3c.org">Enlace 2</a>
<a href="http://www.w3c.org">Enlace 3</a>
<a href="http://www.w3c.org">Enlace 4</a>
</nav>
<section> Sección principal</section>
<aside>Contenido lateral</aside>
<footer>Pie de página</footer>
</body>
</html>
```

Ejemplo 4.31. Página sencilla para ejemplo de estructura

El fichero posicionamiento.css referenciado en la cabecera es el siguiente:

```
header, footer {
    border-radius: 2em;
    box-shadow: 10px 10px 10px black; }
aside{ width:20% }
section{ width:70% }
header{
    height:5em;
    background-color:yellow;
    padding: 1em; }
nav{
    margin: 1em;
    height: 2em;
    background-color: lime;
    text-align:center; }
nav a{ margin-right: 2em }
aside, section{
    height:400px;
    float:left;
    margin-left: 1.5%;
    margin-right: 1.5%;
    padding-left: 1%;
```

```
padding-right: 1%;  
padding-top: 1em;  
margin-top: 2em;  
margin-bottom: 2em;  
background-color: lightblue; }  
  
footer{  
text-align: center;  
clear: both;  
background-color: lime;  
height: 2em; }
```

Ejemplo 4.32. Estilo para estructurar la página web.

En la parte superior hay un elemento `header` y debajo un elemento `nav` que hace de barra de navegación. Los enlaces podrían ser las diferentes secciones de la página.

El contenido principal de la página se divide en dos: un elemento `section` grande a la izquierda y un elemento `aside` más pequeño a la derecha. Entre ambos se reparten el 100% del ancho disponible. Además del ancho, (70% y 20%) `section` y `aside` tienen margen y relleno que afectan al ancho total. En ambos casos, hay un 1.5% de margen y un 1% de relleno a cada lado (izquierdo y derecho). Eso hace que `section` ocupe un total de 75% ($70\% + 2 * 1.5\% + 2 * 1\%$) y `aside` un 25% ($20\% + 2 * 1.5\% + 2 * 1\%$).

Abajo del todo hay un elemento `footer`. Los elementos `footer` y `header` tienen bordes redondeados y sombra.



Ilustración 4.19. Estructura de una página con elementos semánticos y CSS.

4.3. Diseño de estilos para diferentes dispositivos

Las *media queries* (se podría traducir como consultas sobre el medio o dispositivo) permiten adaptar la presentación de la página a las características del dispositivo con que se accede a la página. Constituyen la base para conseguir un diseño adaptable. Entre otras cosas, con las *media queries* podemos pre-guntar por:

- El tipo de dispositivo.
- La orientación de la pantalla.
- El ancho de la ventana del navegador.

Valor	Descripción
all	Todos los dispositivos (modo por defecto)
braille	Líneas braille
embossed	Impresoras braille
handheld	Móviles
print	Para impresión
projection	Proyectores
screen	Pantalla de ordenador normales
speech	Lector de pantalla
tty	Para acceso desde terminal o dispositivos con pantalla muy limitada
tv	Para pantallas de televisión (baja resolución, color)

Tabla 4.7. Valores válidos para el atributo media.

Estructura de una consulta

Una *media query* es una expresión lógica que se evaluará a verdadero o a falso. Cuando se evalúa a verdadero, las reglas asociadas a la consulta entran en vigor. Cuando se evalúa a falso, no se tienen en cuenta.

La primera parte

```
@media ...
```

sirve para indicar para qué tipo dispositivo se aplica la regla. En la tabla 4.8 se listan todos los valores posibles (lo más habitual es distinguir entre *screen* y *print*). Además se pueden añadir condiciones sobre diferentes propiedades del dispositivo como el tamaño, la resolución y la orientación de la pantalla.

El siguiente ejemplo cambia el color de fondo de los elementos *header* cuando la orientación de la pantalla es *portrait* (más alta que ancha), para cualquier tipo de dispositivo.

```
@media all and (orientation: portrait) {  
    header { background-color: lime }  
}
```

Varias de las propiedades admiten los prefijos *min* y *max*. Por ejemplo, si queremos definir algunas reglas que sirvan solo para cuando la anchura de la ventana del navegador sea menor o igual que 800 píxeles:

```
@media all and (max-width: 800px) {  
    ...  
}
```

En cambio, la siguiente regla se aplicaría cuando la anchura sea mayor o igual que 800 píxeles y solo para pantallas de ordenador normales:

```
@media screen and (min-width: 800px) {  
    ...  
}
```

La consulta puede incluir múltiples condiciones unidas por *and* y *or*.

```
@media all and (max-width: 800px) and (orientation: portrait){  
    ...  
}
```

También es posible incluir las *medias queries* al vincular la hoja de estilo a la página.

```
<link rel="stylesheet" href="estilo.css" media="@media all and (min-width: 800px)">  
<!-- la siguiente hoja se aplica solo a pantallas normales-->  
<link rel="stylesheet" href="estilo2.css" media="screen">  
<!-- la siguiente hoja se aplica solo a modo impresión-->  
<link rel="stylesheet" href="estilo3.css" media="print">
```

En este caso, la hoja de estilo *estilo.css* solo está vinculada cuando se cumplen las condiciones.

Propiedad	Significado	Valores	¿Acepta prefijos min y max?
width/height	Anchura y altura de la ventana del navegador	Un tamaño en cualquier unidad de medida CSS	Si
orientation	Orientación	<i>landscape</i> si la ventana es más ancha que alta y <i>portrait</i> si es más alta que ancha	No
device-width/ device-height	Anchura y altura de la pantalla del dispositivo	Un tamaño en cualquier unidad de medida CSS	Si
aspect-ratio Ratio entre anchura y altura en la ventana del navegador		Un número real	Si
device-aspect-ratio	Ratio entre anchura y altura de la pantalla del dispositivo	Un número real	Si
color	Bits por cada componente de color	Un número entero	Si
color-index	Número de colores disponibles en la paleta	Un número entero	Si
monochrome	Número de bits por pixel en el buffer (para pantallas monocromo)	Un número entero	Si
resolution	Resolución de la pantalla	Las medidas de resolución son <i>dpi</i> , <i>dppi</i> y <i>dpcm</i> Ej: 300dpi	Si
scan	Tipo de escaneo en dispositivos tipo "tv"	<i>progressive</i> <i>interlace</i>	No
grid	Tipo de pantalla (dispositivo de salida)	<i>bitmap</i> <i>grid</i>	No

Tabla 4.8. Propiedades para las media queries.

A continuación vamos a ver un ejemplo completo con las siguientes adaptaciones.

- * La barra de navegación cambia de posición con la orientación de la página: por defecto la barra aparece a la izquierda del contenido principal, pero si la orientación es *portrait* pasa a estar sobre el contenido principal.

- El pie de página desaparece si la altura es menor que 900 píxeles.

```
header {  
    background-color: lime;  
    padding: 1em; }  
nav {  
    background-color: yellow;  
    float: left;  
    width: 15%; }  
nav a{ display: block }  
section{  
    background-color: lightblue;  
    width: 70%;  
    height: 200px;  
    float: left;  
    width: 85%; }  
footer {  
    background-color: orange;  
    padding: 1em;  
    clear: both; }  
@media all and (orientation: portrait) {  
    nav{  
        float: none;  
        width: 100%; }  
    nav a{  
        display: inline;  
        margin-right: 1em; }  
    section{  
        width: 100%; }  
}  
@media all and (max-height: 800px) {  
    footer{  
        display: none; }  
}
```

Ejemplo 4.33. CSS con media queries.

El código HTML es el siguiente:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Media queries</title>
        <meta charset="UTF-8">
        <link rel="stylesheet" href="ej_mediaqueries1.css"
            media="screen">
    </head>
    <body>
        <header><h1>Página con media queries</h1></header>
        <nav>
            <a href="#">Vínculo 1</a>
            <a href="#">Vínculo 2</a>
            <a href="#">Vínculo 3</a>
        </nav>
        <section>
            <p>Zona para el contenido principal </p>
        </section>
        <footer>
            Pie
        </footer>
    </body>
</html>
```

Ejemplo 4.34. Código HTML para el ejemplo 4.33.

Cuando cambia la orientación, se aplican nuevas reglas a los elementos `nav`, `section`, y a los elementos a dentro de `nav`. En concreto

- El elemento `nav` deja de ser flotante. Así, `nav` y `section` dejarán de estar uno al lado de otro. El elemento `nav` pasa a estar encima de `section`.
- Cambia la anchura de los dos elementos. Antes se repartían el ancho de la pantalla (85% y 15%), ahora los dos ocupan todo el ancho disponible.
- Los vínculos dentro de `nav` (selector `nav a`) pasan a ser elementos de línea. Se añade un margen a la derecha para que estén espaciados.

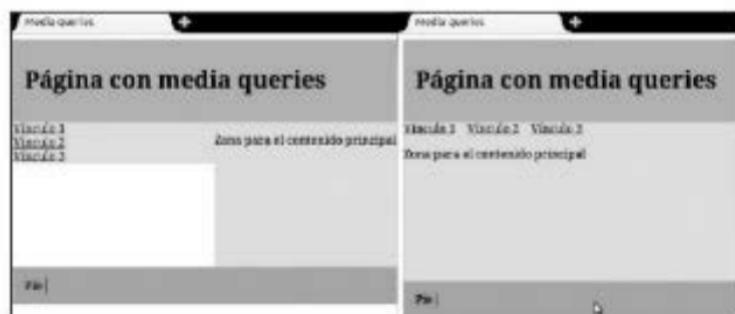


Ilustración 4.20. El aspecto de esta página cambia con la orientación de la pantalla.

Algo similar pasa con el elemento `footer`. Cuando la altura es menor de 800 píxeles se aplica `display:none` y por tanto el elemento deja de mostrarse, como se puede observar en la ilustración 4.21, obtenida usando la *vista de diseño adaptable* de Firefox.

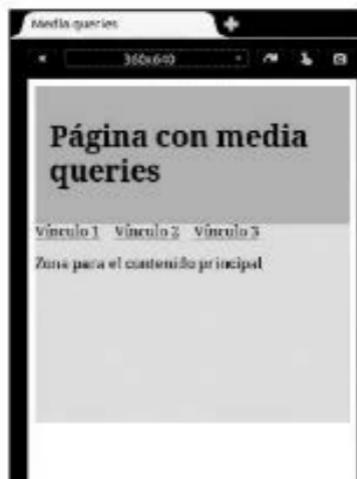


Ilustración 4.21. Cuando el tamaño se reduce, desaparece el footer.

El resto de reglas (las que no están sobreescritas por la `media query`) se siguen aplicando.

4.4. Buenas prácticas en el uso de hojas de estilo

A la hora de escribir hojas de estilo conviene seguir unas pautas básicas, orientadas sobre todo a facilitar su mantenimiento y reutilización:

- Usar comentarios y escribir hojas de estilo fáciles de leer.
- Utilizar hojas de estilo externas.
- Organizar las hojas de estilo de manera modular para que se puedan reutilizar en otras páginas. Además, es más sencillo manejar varias hojas de estilo pequeñas que un grande.
- Combinar elementos con el selector múltiple para no repetir propiedades.
- Utilizar las propiedades resumen.
- Utilizar herramientas específicas para el desarrollo de CSS.

Herramientas útiles

Hay varias herramientas que pueden ayudar en el desarrollo de CSS. A medida que un diseñador va aprendiendo más y desarrollando páginas más complejas, es recomendable que investigue estas herramientas:

- Validador de CSS del W3C.
- Preprocesadores de CSS.
- Programas que evalúan la calidad del código.
- Frameworks para CSS.
- Técnicas para acelerar la carga de páginas.
- Limpiadores de código.

Validador de CSS del W3C

Como en el caso de HTML, el validador del W3C (<http://jigsaw.w3.org/css-validator/>) es la herramienta de referencia para verificar si estamos escribiendo hojas de estilo válidas.

Hay muchos diseñadores web que no están interesados en la validación y que se centran en probar sus páginas en varios dispositivos y tamaños de pantalla. Ciertamente es un punto de vista pragmático y poco discutible. Aun así, conviene pasar todas las hojas de estilo por el validador y tomar nota de los errores y advertencias.

Preprocesadores CSS

Sass es una extensión del lenguaje CSS. Ofrece nuevas opciones de sintaxis para facilitar el desarrollo de hojas de estilo complejas. Por ejemplo, es posible definir variables o herencia entre reglas.

Los desarrolladores de Sass no escriben ficheros CSS normales. Escriben ficheros Sass que se transforman automáticamente a ficheros CSS. Estos son los que finalmente se usarán en producción. Por ejemplo, este fichero

```
$font-stack: Helvetica, sans-serif;  
$primary-color: #444;  
body {  
    font: 100% $font-stack;  
    color: $primary-color; }
```

Ejemplo 4.35. Fichero Sass (.sass)

se transforma en

```
body {  
    font: 100% Helvetica, sans-serif;  
    color: #444444; }
```

En Sass se pueden usar dos sintaxis diferentes. La original, también llamada Sass, no es muy parecida a la sintaxis de CSS (está basada en la indentación, como el lenguaje Python). Posteriormente se publicó SCSS (Sassy CSS), que tienen una sintaxis basada en CSS y está considerada como la opción preferente.

Organizadores y limpiadores de hojas de estilo

Hay muchos optimizadores online disponibles (<http://csstidyonline.com/>, <http://tools.maxcdn.com/processors/>). Su objetivo es crear hojas de estilo que ocupen menos y sean más fáciles de entender y mantener. Aunque las funcionalidades de cada uno pueden variar, las más habituales son

- Limpiar espacios en blanco y comentarios.
- Eliminar reglas redundantes o que no llegan a aplicarse.
- Transformar los valores de los colores al formato que menos ocupen.
- Unir varias propiedades en propiedades resumen.

Técnicas para mejorar la velocidad de carga de las páginas.

Una de las cuestiones más importantes en el diseño web es el tiempo de carga de la página. En el tiempo de carga incluye no sólo la cantidad de información que incluya la página, sino en cuantos ficheros se reparta. Esto es así porque para cada fichero que queramos descargar hay que realizar una nueva petición al servidor, y esta comunicación conlleva un tiempo de establecimiento adicional al de transmisión de datos. Por lo tanto, se tarda más en descargar 10 ficheros de 1k que un fichero de 10k.

Normalmente en una página web habrá más de una CSS. Es habitual que las hojas de estilo grandes se partan en varias para que sean más reusables y fáciles de entender.

Los desarrolladores podrían ocuparse de esto manualmente. A la hora de subir las hojas de estilo al servidor, podría unirlas todas en un solo fichero cortando y pegando, modificar los ficheros HTML para que vincularan la nueva hoja de estilo y subirla al servidor. Lo malo de este enfoque es que cada vez que hagamos un cambio en alguna de las hojas, hay que repetir todo el proceso.

Para automatizarlo podemos usar algo como [minify](#). Es una aplicación PHP que permite acceder a un solo fichero "minificado" a través de una URL. A través de una interfaz gráfica podemos elegir qué ficheros (de los presentes en el servidor) queremos unir en uno y asignarles una URL. Cuando se pida al servidor esa URL, devolverá un solo fichero. Si se realizan cambios en los ficheros originales, basta con subirlos al servidor.

Programas que evalúan la calidad del código.

Las herramientas tipo [linter](#) realizan un análisis estático del código para detectar errores, posibles fuentes de problemas, malas prácticas de programación o inefficiencias. Para CSS, la herramienta CSSLint es una de las más utilizadas. Se puede usar online (<http://csslint.net/>) o a través de línea de comandos.

Por ejemplo, para la regla

```
#contenedor{  
    box-shadow: 20px 20px 10px 5px black;  
    background: linear-gradient(#339933, #00FF00);  
    margin-bottom: 0em;  
    float: left;  
    width: 85%;  
    min-height: 300px; }
```

CSSLint no encuentra errores, pero sí dos advertencias:

- “Values of 0 shouldn’t have units specified”. Se refiere al atributo `margin-bottom`, que tiene un valor de `0em`. Los valores 0 no deberían llevar unidades.
- “Don’t use IDs in selectors”. No recomienda usar el atributo `id` en los selectores (y por tanto en el HTML) porque no son directamente reusables. La opción recomendada es usar `class`, aunque hay cierta polémica entre los diseñadores web puesto que muchos no ven problema en usar el atributo `id`.

En vista de que no hay un consenso sobre la idoneidad de todas reglas de CSSLint, hay opciones configurar las reglas con las queremos que se revisen las hojas de estilo para adaptarlas a nuestras preferencias.

Frameworks para CSS

Los frameworks para CSS proveen una serie de herramientas para diseñar páginas web. Su objetivo es acelerar el desarrollo y facilitar la creación de páginas válidas.

Podemos tomar como ejemplo el framework Bootstrap, que usa SaaS. Fue desarrollado por la compañía Twitter para mejorar la consistencia entre sus aplicaciones y simplificar el mantenimiento. Entre otras herramientas, incluye:

- Hoja de estilo. Una serie de hojas de estilo que cubre la mayoría de los elementos de HTML, para dar un aspecto uniforme y elegante. Hay una hoja de estilo básica y un tema adicional más vistoso.
- Componentes reusables: botones, pestañas, barras de progreso.
- Componentes JavaScript. Incluye plugins de jQuery para crear controles como carruseles o funciones de autocompletado.

Para empezar a usar Bootstrap basta con incluir la hoja de estilo correspondiente. Podemos descargarla o usarla online.

```
<link rel="stylesheet" href="http://netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.min.css">
```

Una vez incluida la hoja de estilo, podemos asignar estilo a través de una serie de clases definidas. Por ejemplo, podemos crear una tabla con borde y fondo de color alternante para las filas así:

```
<table class="table table-striped table-bordered">
<caption>Tabla de películas, directores y guionistas</caption>
```

```

<tr>
<th>Película</th>
<th>Director</th>
<tr>
<tr>
<td>Los otros</td>
<td>Alejandro Amenábar</td>
<tr>
<tr>
<td>Los pájaros</td>
<td>Alfred Hitchcock</td>
<tr>
</table>

```

Ejemplo 4.36. Añadir estilo con Bootstrap.

Película	Director
Los otros	Alejandro Amenábar
Los pájaros	Alfred Hitchcock

Ilustración 4.22. Estilo para tablas en Bootstrap.

Ejercicios propuestos

Test tema 1

1. ¿Cuáles son las tecnologías de diseño web más importantes actualmente?
 - a. HTML, CSS y JavaScript.
 - b. HTML, CSS y Java.
 - c. Flash, Dreamweaver y XML.
2. En el diseño orientado a usuario:
 - a. El usuario solo interviene en las fases finales del proyecto, para retocar la interfaz.
 - b. El usuario solo interviene en las primeras fases del proyecto, para explicar lo que quiere.
 - c. El usuario interviene durante todo el proceso, colaborando con los diseñadores en un proceso iterativo.
3. Marca la afirmación que consideres correcta respecto al diseño orientado a implementación.
 - a. Se ocupa de la usabilidad de la página, básicamente en la interfaz de usuario.
 - b. Se ocupa de cómo está hecha la página. Por ejemplo, en relación al cumplimiento de estándares.
 - c. Se ocupan cuestiones relacionadas con el servidor web donde se aloje la página.
4. El diseñador web normalmente es responsable de:
 - a. El lado diseño de lado del cliente (*front-end*) de una aplicación web.
 - b. El diseño de las interfaces y bases de datos de la aplicación web.
 - c. El diseño del lado del cliente y también los programas necesarios en el servidor.
5. ¿Cuál de las siguientes opciones no recomendarias de cara a la versión para imprimir del recibo de una tienda online?
 - a. Añadir una línea con la URL del sitio web.
 - b. Mantener las animaciones.
 - c. Retirar los elementos de navegación.

6. Marca la afirmación que consideres correcta.

- a. El esquema de una página web debe incluir información sobre color y tipografía.
- b. El esquema de una página web se centra en la disposición de la información y la navegación por el sitio web.
- c. El esquema de una página web se centra en el tiempo de carga.

7. ¿Cuál de los siguientes no te parece un principio de diseño apropiado?

- a. Mostrar la información de forma jerarquizada.
- b. Incluir más de una forma de navegación por el sitio web.
- c. Usar tipografías y colores diferentes en cada sección del sitio.

8. Un sitio web accesible...

- a. Está disponible más del 90% del tiempo.
- b. Es fácil de usar.
- c. Está adaptado a usuarios con limitaciones (por ejemplo, visuales).

9. Respeto de la compatibilidad entre navegadores:

- a. Seguir los estándares del W3C asegura que la página funcione en todos los navegadores.
- b. Seguir los estándares del W3C es una buena práctica, pero no asegura que la página funcione en todos los navegadores.
- c. Todos los navegadores son casi iguales, no hace falta preocuparse de la compatibilidad entre navegadores.

10. Marca la afirmación que consideres correcta respecto al diseño orientado a objetivos.

- a. Se ocupa de la usabilidad de la página, básicamente en la interfaz de usuario.
- b. Es apropiado para las primeras fases del desarrollo.
- c. Se ocupa de mejorar la velocidad de carga.

Respuestas: 1. a, 2. c, 3. b, 4. a, 5. b, 6. b, 7. c, 8. c, 9. b, 10. b.

Actividades

1. Prueba a hacer un esquema (*wireframe*) para un sitio web sencillo. Puede ser una página de presentación personal o la de un negocio pequeño. Utiliza una herramienta específica como Lumzy (<http://www.lumzy.com/>) o mockingbird (<https://gomockingbird.com/>).

Test tema 2

1. Puede decirse que el XHTML...

- a. Es una aplicación del lenguaje HTML.
- b. Es una aplicación del lenguaje SGML.
- c. Es una aplicación del lenguaje XML.
- d. Ninguna de las anteriores.

2. ¿Cómo se pone un comentario en un fichero XML?

- a. /*comentario*/
- b. // comentario
- c. <comment> comentario <comment>
- d. <!-- comentario -->

3. Si se usa un atributo booleano...

- a. No es necesario escribir el valor.
- b. Hay que poner el valor, que solo puede ser 'true' o 'false'.
- c. Se puede poner en la etiqueta de cierre del elemento.
- d. No se pueden usar otros atributos en el mismo elemento.

4. Elige la respuesta incorrecta. Un fichero XML bien formado:

- a. Tiene un solo nodo raíz.
- b. Puede tener varios nodos raíz.
- c. No puede tener elementos sin etiquetas de cierre.
- d. Debe tener los valores de los atributos entre comillas.

5. Marca la afirmación que consideres correcta.

- a. En XML, los caracteres '<', '>' y '&' se pueden usar sin problemas.
- b. En XML, en las etiquetas no se distingue entre mayúsculas y minúsculas.
- c. En XML, el orden de los atributos importa.
- d. En XML, en las etiquetas se distingue entre mayúsculas y minúsculas.

6. Marca la afirmación que consideres correcta sobre DTD y esquemas XML

- a. Ambos sirven para comprobar si un fichero está bien formado.

- b. Los DTD sirven para comprobar si un fichero es válido y los esquemas XML para comprobar si está bien formado.
- c. Los esquemas XML sirven para comprobar si un fichero es válido y los DTD para comprobar si está bien formado.
- d. Ambos sirven para comprobar si un fichero es válido.

7. Una sección CDATA sirve para:

- a. Poner datos adicionales sobre el tipo de fichero XML.
- b. Añadir metadatos en la cabecera XML.
- c. Escribir un contenido que no va a ser interpretado como XML.
- d. No es un campo válido en XML.

8. Marca la opción que consideres correcta.

- a. Un fichero XML puede ser válido sin estar bien formado.
- b. Un fichero XML puede ser bien formado sin ser válido.
- c. Válido y bien formado son conceptos equivalentes: un fichero bien formado es un fichero válido y viceversa.
- d. Ninguna de las anteriores.

9. ¿Cuál de las siguientes opciones es correcta como elemento XML?

- a. <nombre>Pablo</nombre>
- b. <persona nombre="Pablo">
- c. <persona nombre="Pablo"/>
- d. <Nombre="Pablo"></nombre>

10. ¿Qué indica esta instrucción de procesamiento?

```
<?xml version='1.0' encoding='UTF-8'?>
```

- a. La versión del lenguaje y la codificación de caracteres que se usan en el documento.
- b. Solo la versión del lenguaje.
- c. Solo la codificación de caracteres.
- d. Ninguna de las anteriores.

Respuestas: 1. c, 2. d, 3. a, 4. b, 5. d, 6. d, 7. c, 8. b, 9. c, 10. a.

Actividades

1. Busca información sobre lenguajes basados en XML. Por ejemplo:

- SVG. Es un lenguaje para gráficos vectoriales.
- MathML. Para fórmulas matemáticas.
- Busca ejemplos de ambos e intenta visualizarlos con un navegador web.

Test tema 3

- 1. Marca la afirmación que consideres verdadera sobre la etiqueta meta:**
 - a. La etiqueta *meta* especifica el DOCTYPE, que indica qué versión de HTML o XHTML se usa en la página.
 - b. La etiqueta *meta* está en desuso y se considera que su uso es un error de diseño.
 - c. La etiqueta meta incluye información sobre la propia página, como el autor o el idioma.
 - d. La etiqueta meta se usa para vincular a una página documentos tales como una CSS.
- 2. Marca la respuesta que consideres correcta.**
 - a. En el lenguaje HTML se distingue entre mayúsculas y minúsculas.
 - b. El atributo *alt* del elemento *img* es opcional.
 - c. El elemento *title* es obligatorio.
 - d. Ninguna de las anteriores.
- 3. Las páginas de marcos (frameset)...**
 - a. Son una novedad de HTML 5.
 - b. Son la opción ideal para dispositivos móviles.
 - c. No se recomiendan porque mezclan la información con la forma de representarla, entre otras cosas.
 - d. Son la mejor opción para maquetar una página de manera sencilla.
- 4. ¿Qué hay que introducir en los enlaces para que las páginas se carguen en una ventana/pestana nueva del navegador?**
 - a. *target="_blank"*
 - b. *target="_nueva"*
 - c. *newwin="_new"*
 - d. *windows="_top"*
- 5. Elige la etiqueta correcta para el encabezado de mayor tamaño:**
 - a. *h1*
 - b. <header>

- c. <title>
d. h6
6. ¿Cuál de las siguientes opciones es correcta si queremos hacer un enlace desde nuestra página a la página ‘www.elpais.com’?
- a. ...
b. ...
c. <link page="http://www.elpais.com">...
d. <link href="http://www.elpais.com">...
7. Elige la opción correcta para insertar una imagen:
- a. <image src="imagen.gif" alt="MilImagen" />
b. imagen.gif
c.
d. <image data="imagen.gif" alt="MilImagen" />
8. ¿Cómo puedes hacer una lista que use números para los elementos?
- a.
b. <list>
c. <dt>
d.
9. La primera etiqueta fundamental en un documento HTML es...
- a. <a>
b. <title>
c. <head>
d. <html>
10. ¿Qué respuesta contiene solo etiquetas relacionadas con tablas?
- a. col, table, alt, th
b. strong, table, td, th
c. tr, td, href, table
d. tr, td, caption, colgroup

Respuestas correctas: 1. c, 2. c, 3. c, 4. a, 5. a, 6. b, 7. c, 8. a, 9. d, 10. d

Actividades

1. Crea una página con este aspecto.

Elementos básicos de html

En HTML hay listas no numeradas,

- Patatas
- Leche
- Fresas

Numeradas

3. Patatas
4. Leche
5. Fresas

y de definición

Patatas
 Tubérculo
Fresas
 Fruta

También tenemos tablas

Mes	Ahorros
Suma	\$180
Enero	
Febrero	\$80

Podemos dar formato al texto (no se pueden usar las etiquetas i o b)

Énfasis Remarcado

Por supuesto, hay vínculos, que podemos abrir en ventana nueva, como aquí

[El país](#)

o en la misma ventana, como en este otro

[El País](#)

Multimedia

Podemos incluir audio con controles,



Busca un fichero de audio en internet e insértalo en la página. Asegúrate de que se oye bien en varios navegadores. Si necesitas convertir el formato del fichero, utiliza un conversor online (<http://www.online-convert.com>).

2. Escribe una página con un formulario como el de la imagen.

Formulario de solicitud de abono transportes

Selecione zona
A B C D

Tipo de abono
Normal

Nombre

Teléfono

Marque las casillas correspondientes

Primera solicitud de abono

Solicita nueva tarjeta sin contacto

Solicita tarjeta abono anual

Si tiene alguna duda, puede contactar con nosotros en el 123456789.

- La lista desplegable debe mostrar los valores 'Joven', 'Normal' y 'Tercera Edad'. El formulario se envía a 'procesar.php' con el método GET. Observa la URL para comprobar que envía todos los campos.
3. Basándote en el ejemplo 3.32 crea una página con un *iframe* para mostrar las páginas de los ejercicios anteriores.
4. Prueba las páginas que has escrito con el validador de W3C. Haz los cambios necesarios para que no haya errores ni advertencias.

Test tema 4

1. ¿Cuál de las siguientes propiedades de estilo no tiene que ver con el modelo de cajas?
 - a. width
 - b. top
 - c. color
 - d. padding
2. Marca la afirmación que consideres verdadera:
 - a. El objetivo principal de las CSS es acelerar la velocidad de carga de las páginas.
 - b. El objetivo principal de las CSS es acelerar la compatibilidad entre navegadores.
 - c. El objetivo principal de las CSS es separar la información de la forma en la que se presenta.
 - d. El objetivo principal de las CSS es mejorar la legibilidad del código HTML.
3. ¿Qué propiedad sirve para hacer que un elemento sea de bloque o linea?
 - a. display
 - b. background-color
 - c. padding
 - d. list-style
4. Para poner reglas de estilo a un elemento con id="elemento", usaremos:
 - a. elemento[...]
 - b. .elemento{...}
 - c. #elemento{...}
 - d. elemento{...}
5. ¿Cómo se escribe un comentario en CSS?
 - a. <!--comentario-->
 - b. /*comentario*/
 - c. (:comentario:)
 - d. {comentario}

6. Las *media queries* sirven para ...

- a. Crear reglas específicas para diferentes tipos de dispositivos.
- b. Crear bordes redondeados.
- c. Crear sombras.
- d. Realizar consultas a bases de datos.

7. ¿Cuál de los siguientes selectores escogería los elementos a con clase importación?

- a. a importacion{...}
- b. a importacion(...)
- c. a, importacion{...}
- d. a+importacion{...}

8. En caso de conflicto, ¿qué hojas de estilo tienen mayor prioridad?

- a. Las hojas de estilo externas.
- b. Las hojas de estilo internas.
- c. Las hojas de estilo en linea.
- d. Todas tienen la misma prioridad.

9. ¿Cuál de los siguientes no es un modo de posicionamiento?

- a. absolute
- b. mobile
- c. fixed
- d. relative

10. ¿Dónde se incluye una hoja de estilos externa?

- a. En el elemento `head`, con un elemento `link`.
- b. En el elemento `body`, con un elemento `link`.
- c. En el elemento `head`, con un elemento `style`.
- d. En el elemento `body`, con un elemento `style`.

Respuestas correctas: 1. c, 2. c, 3. a, 4. c, 5. b, 6. a, 7. b, 8. c, 9. b , 10. a.

Actividades

1. Haz una página como la de la imagen con HTML y CSS. Usa una hoja de estilos externa.

The screenshot shows a web browser window with the title 'Bienvenido al IES Clara del Rey'. Below the title is a sub-header 'Formulario de Registro' and a note 'No es posible de volver al siguiente formulario'. The form has two columns: 'Datos personales' and 'Datos para dirección'. The 'Datos personales' column contains fields for 'Nombre y apellido' (with placeholder 'Juan Pérez') and 'Sexo' (with options 'Hombre' and 'Mujer'). The 'Datos para dirección' column contains fields for 'Calle', 'Número', 'Piso', 'Localidad', and 'Provincia' (with placeholder 'Madrid'). At the bottom right of the form area are 'Siguiente' and 'Cancelar' buttons. A footer at the bottom left says 'Página realizada por IES Clara del Rey'.

- Todos los vínculos se abren en ventana nueva.
 - La lista mostrará los valores 'Informática', 'Administración' y 'Comercio'.
 - Añade las reglas de estilo necesarias para que el color de fondo de los enlaces de la barra lateral cambie cuando pase el ratón por encima. No debe afectar al vínculo del pie de página.
 - Observa las esquinas redondeadas y las sombras.
 - Utiliza las etiquetas semánticas apropiadas.
2. Valida la hoja de estilo del ejercicio 1 con el validador del W3C (<http://jigsaw.w3.org/css-validator/>). Haz los cambios necesarios para que no tenga errores.
3. Pasa la hoja de estilo del ejercicio 1 por CSSLint (<http://csslint.net/>) y por algún optimizador de CSS (como <http://cssidyonline.com/>) y observa los resultados.

4. Añade reglas de estilo para que la página del ejercicio 1 cambie de aspecto según las características del dispositivo. En concreto:
- Si la ventana sea más alta que ancha, el color de fondo de la cabecera cambia.
 - Si la anchura es menor de 800px, la barra lateral desaparece y la sección del formulario pasa a ocupar todo el ancho disponible.
5. Escribe una página con el aspecto de la ilustración 4.3. Usa una hoja de estilo externa.

Bibliografia

Referencias

Páginas web

- Especificación oficial de HTML. <http://www.w3.org/TR/html5/>
- Estado de CSS3 http://www.w3.org/standards/techs/css#w3c_all
- Wiki del W3C. http://www.w3.org/wiki/Main_Page
- CSS Zen Garden (demonstración de CSS). <http://www.csszengarden.com/>
- Libros.
- Stunning CSS3 A Project - Based Guide To The Latest In CSS. Zoe Gillenwater.
- HTML5 for web designers. Jeremy Keith.
- HTML5 y CSS3. Revolucione el diseño de sus sitios web. Christophe Aubry.
- HTML5 y CSS3. Alexis Goldstein et al.
- El gran libro de HTML5, CSS3 y Javascript. Juan Diego Gauchat.
- Introducing HTML5. Bruce Lawson, Remy Sharp.
- The definitive guide to HTML5. Adam Freeman.
- Smashing HTML5. Bill Sanders.

Validadores

- Validador de CSS. <http://jigsaw.w3.org/css-validator/>
- Validador de HTML y XHTML. <http://validator.w3.org/>
- Validador para móviles. <http://validator.w3.org/mobile/>

Herramientas útiles para el diseño web

- Selector de colores. <http://www.colorpicker.com/>
- Creación de iconos. <http://iconion.com/posts/icon-maker/>
- Conversión de ficheros de audio video. <http://www.online-convert.com>
- CSSLint. <http://csslint.net>
- SaaS. <http://sass-lang.com/>
- Herramientas para esquemas: Lumzy (<http://www.lumzy.com/>), mockingbird<https://gomockingbird.com/>).

Elaboración de documentos web mediante lenguajes de marcas

El mundo actual no se puede concebir sin Internet. En la educación, el trabajo y la vida personal son pocas las personas que no utilizan diariamente una página o aplicación web. Por tanto, el desarrollo web se ha convertido en un área fundamental de la informática y el diseño.

En su corta historia, el desarrollo web ha experimentado muchos cambios. Este libro trata sobre dos lenguajes básicos para crear páginas web en sus últimas versiones, HTML5 y CSS3. El HTML se usa para estructurar el contenido de la página, mientras que con CSS se añade la información de presentación, como colores, tipos de letra y maquetación.

En ambos casos hay muchas novedades respecto a las versiones anteriores. En HTML han aparecido nuevas etiquetas, como las semánticas, y otras han desaparecido o su significado ha cambiado. Las hojas de estilo también han aumentado mucho su funcionalidad y ahora se pueden utilizar para adaptar la página a diferentes dispositivos o para crear animaciones, entre otras novedades.

Todo ello se desarrolla siguiendo los contenidos que marca el BOE para esta unidad formativa 1841 *Elaboración de documentos web mediante lenguajes de marcas*, primera de las tres que integran el módulo formativo 0491_3 *Programación web en el entorno cliente*. El documento legal que define esta unidad se recoge en el RD 1531/2011, modificado por el RD 628/2013, que regula el certificado de profesionalidad *Desarrollo de tecnologías con aplicaciones web*. En este manual se incluyen, además, actividades de autoevaluación y desarrollo que complementan pedagógicamente los contenidos formativos.

Xabier Gaxetábil García es ingeniero de Informática y licenciado en Ciencias Físicas. Funcionario de carrera, ejerce su labor docente en la Comunidad de Madrid.

ISBN: 978-84-283-9831-2

