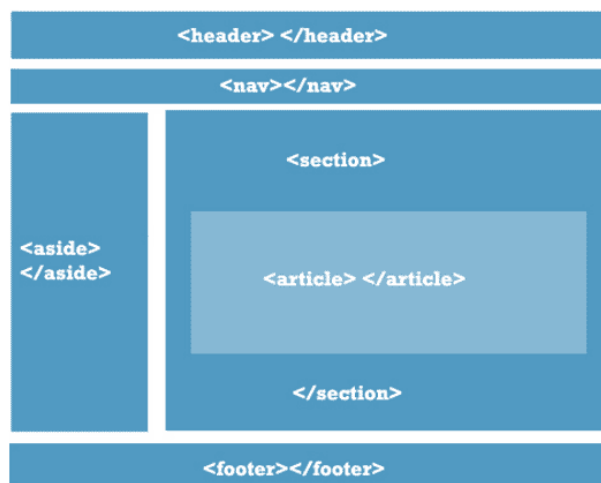


TEMA 2

El lenguaje de marcas HTML



Módulo: Diseño de interfaces web
CFGs. Desarrollo de Aplicaciones Web (DAW)
Curso 2012-2013
I.E.S. COMERCIO

ÍNDICE

1. INTRODUCCIÓN	5
1.1. ¿Qué es HTML?	5
1.2. Breve historia de HTML	5
1.3. Especificación oficial.....	7
1.4. HTML, XHTML y HTML5	8
1.5. HTML y CSS.....	10
2. CARACTERÍSTICAS BÁSICAS.....	10
2.1. Etiquetas HTML.....	10
2.1.1. Comprender los elementos	13
2.2. El documento HTML	14
2.2.1. La definición de tipo de documento.....	14
2.2.2. El esqueleto básico	16
2.3. Etiquetas y atributos	20
2.4. Elementos HTML	28
2.5. Sintaxis de las etiquetas XHTML	31
3. TEXTO	33
3.1. Estructurar	35
3.1.1. Párrafos.....	35
3.1.2. Secciones.....	37
3.2. Marcado básico de texto.....	39
3.3. Marcado avanzado de texto.....	45
3.4. Marcado genérico de texto	49
3.5. Nuevas etiquetas de marcado genérico de texto en HTML5	51
3.5.1. Etiqueta <wbr> (ajuste de línea)	51
3.5.2. Etiqueta <time>	52
3.5.3. Etiqueta <mark>	53
3.5.4. Etiqueta <meter>.....	53
3.6. Espacios en blanco y nuevas líneas.....	54
3.6.1. Nuevas líneas	55
3.6.2. Espacios en blanco	57
3.6.3. Texto preformateado	59

3.7. Codificación de caracteres.....	63
4. ENLACES	67
4.1. URL.....	67
4.2. Enlaces relativos y absolutos.....	71
4.3. Enlaces básicos.....	77
4.4. Enlaces avanzados.....	82
4.4.1. Idioma del enlace (hreflang)	82
4.4.2. Tipo de contenido (type).....	83
4.4.3. Tipo de relación (rel y rev).....	83
4.4.4. Tipo de target	84
4.4.5. Tipo de medio (media). Nuevo en HTML5	85
4.5. Otros tipos de enlaces	87
4.6. Ejemplos de enlaces habituales	90
4.6.1. Enlace al inicio del sitio web.....	90
4.6.2. Enlace a un email.....	90
4.6.3. Enlace a un archivo FTP	91
4.6.4. Enlazar varias hojas de estilos CSS.....	91
4.6.5. Enlazar hojas de estilos CSS para diferentes medios.....	91
4.6.6. Enlazar el favicon	92
4.6.7. Enlazar un archivo RSS	92
4.6.8. Enlazar hojas de estilos, favicon y RSS	92
4.6.9. Indicar que existe una versión de la página en otro idioma.....	92
4.6.10. Indicar que existe una versión de la página preparada para imprimir	93
4.6.11. Indicar que existe una página que es índice de la página actual ..	93
5. LISTAS.....	93
5.1. Listas no ordenadas	93
5.2. Listas ordenadas	95
5.3. Listas de definición	96
6. IMÁGENES Y OBJETOS	100
6.1. Imágenes	100
6.2. Etiquetas figure y figcaption.....	103
6.3. Mapas de imagen	105

6.3. Objetos	107
7. TABLAS	111
7.1. Tablas básicas	112
7.2. Tablas avanzadas.....	123
8. FORMULARIOS.....	130
8.1. Formularios básicos.....	130
8.2. Elementos de formulario	132
8.2.1. Cuadro de texto.....	133
8.2.2. Cuadro de contraseña.....	135
8.2.3. Checkbox	135
8.2.4. Radiobutton.....	136
8.2.5. Botón de envío de formulario	137
8.2.6. Botón de reseteo del formulario	137
8.2.7. Ficheros adjuntos.....	138
8.2.8. Campos ocultos.....	139
8.2.9. Botón de imagen	139
8.2.10. Botón.....	140
8.2.11. Color.....	140
8.2.12. Campos para las fechas y las horas	140
8.2.13. Direcciones de e-mail.....	142
8.2.14. Valores numéricos.....	142
8.2.15. Barras de selección con cursor	143
8.2.16. Campos de búsqueda	143
8.2.16. Números de teléfono	144
8.2.17. Campo para las URL.....	144
8.2.18. Campo de introducción de datos con sugerencias. Etiqueta <datalist>.....	145
8.3. Formularios avanzados.....	147
8.4. Otros elementos de formulario.....	151
8.5. La validación de los formularios.....	158
8.5.1. La validación del lado del cliente.....	158
8.5.2. Desactivar la validación.....	159
8.5.3. Insertar un campo obligatorio.....	159

8.5.4. Los valores autorizados	159
8.5.5. Las expresiones regulares	160
8.5.6. Las ayudas para los usuarios de un formulario.....	162
9. OTROS ELEMENTOS DE HTML5	163
9.1. El elemento <hgroup>.....	163
9.2. Los elementos <details> y <summary>	164
9.3. El elemento <progress>.....	165
10. ESTRUCTURA Y LAYOUT	165
10.1. Los elementos de la estructura en HTML4	165
10.2. Los elementos de la estructura en HTML5	169
10.3. El atributo semántico "role"	171
10.4. Ejemplos de estructura en HTML5.....	172
10.5. Plantillas para sitios web en HTML5.....	175
10.5.1. La plantilla ArchiteXture	175
10.5.2. La plantilla Da Front Page.....	183
10.5.3. La plantilla Learning Center	190
11. METAINFORMACIÓN.....	198
11.1. Estructura de la cabecera	198
11.2. Metadatos	200
12. OTRAS ETIQUETAS IMPORTANTES	202
12.1. Comentarios	202
12.2. JavaScript	203
12.3. CSS	205
12.5. Otras etiquetas	205
13. COMPROBACIÓN DE ERRORES EN LAS PÁGINAS HTML.....	207
14. CARACTERÍSTICAS HTML5 QUE SOPORTA CADA NAVEGADOR	210

1. INTRODUCCIÓN

1.1. ¿Qué es HTML?

Definiéndolo de forma sencilla, *"HTML es lo que se utiliza para crear todas las páginas web de Internet"*. Más concretamente, HTML es el *lenguaje* con el que se *"escriben"* la mayoría de páginas web.

Los diseñadores utilizan el lenguaje HTML para crear sus páginas web, los programas que utilizan los diseñadores generan páginas escritas en HTML y los navegadores que utilizamos los usuarios muestran las páginas web después de leer su contenido HTML. Aunque HTML es un lenguaje que utilizan los ordenadores y los programas de diseño, es muy fácil de aprender y escribir por parte de las personas. En realidad, HTML son las siglas de *HyperText Markup Language* y más adelante se verá el significado de cada una de estas palabras.

El lenguaje HTML es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado **World Wide Web Consortium** (<http://www.w3c.es/>), más conocido como **W3C**. Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de forma muy similar en cualquier navegador de cualquier sistema operativo.

El propio **W3C** define el lenguaje HTML como *"un lenguaje reconocido universalmente y que permite publicar información de forma global"*. Desde su creación, el lenguaje HTML ha pasado de ser un lenguaje utilizado exclusivamente para crear documentos electrónicos a ser un lenguaje que se utiliza en muchas aplicaciones electrónicas como buscadores, tiendas online y banca electrónica.

1.2. Breve historia de HTML

La historia completa de HTML es tan interesante como larga, por lo que a continuación se muestra su historia resumida a partir de la información que se puede encontrar en la Wikipedia.

El origen de HTML se remonta a 1980, cuando el físico **Tim Berners-Lee**, trabajador del CERN (Centro Europeo de Investigaciones Nucleares, <http://www.cern.ch/>) propuso un nuevo sistema de *"hipertexto"* para compartir documentos.

Los sistemas de *"hipertexto"* habían sido desarrollados años antes. En el ámbito de la informática, el *"hipertexto"* permitía que los usuarios accedieran a la información

relacionada con los documentos electrónicos que estaban visualizando. De cierta manera, los primitivos sistemas de "*hipertexto*" podrían asimilarse a los enlaces de las páginas web actuales.

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

El primer documento formal con la descripción de HTML se publicó en 1991 bajo el nombre "*HTML Tags*" (*Etiquetas HTML*, <http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html>) y todavía hoy puede ser consultado online a modo de *reliquia informática*.

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

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

A partir de 1996, los estándares de HTML los publica otro organismo de estandarización llamado **W3C** (*World Wide Web Consortium*, <http://www.w3.org/>). La versión HTML 3.2 se publicó el 14 de Enero de 1997 y es la primera recomendación de HTML publicada por el W3C. Esta revisión incorpora los últimos avances de las páginas web desarrolladas hasta 1996, como *applets* de Java y texto que fluye alrededor de las imágenes.

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

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

Desde la publicación de HTML 4.01, la actividad de estandarización de HTML se detuvo y el W3C se centró en el desarrollo del estándar XHTML. Por este motivo, en el

año 2004, las empresas Apple, Mozilla y Opera mostraron su preocupación por la falta de interés del W3C en HTML y decidieron organizarse en una nueva asociación llamada WHATWG (*Web Hypertext Application Technology Working Group*, <http://www.whatwg.org/>).

La actividad actual del WHATWG se centra en el futuro estándar HTML 5, cuyo primer borrador oficial (<http://www.w3.org/TR/2008/WD-html5-20080122/>) se publicó el 22 de enero de 2008. Debido a la fuerza de las empresas que forman el grupo WHATWG y a la publicación de los borradores de HTML 5.0, en marzo de 2007 el W3C decidió retomar la actividad estandarizadora de HTML (<http://www.w3.org/2007/03/html-pressrelease>). Actualmente el W3C está trabajando en el borrador de HTML 5 (<http://www.w3.org/TR/html5/>).

De forma paralela a su actividad con HTML, W3C ha continuado con la estandarización de XHTML, una versión *avanzada* de HTML y basada en XML. La primera versión de XHTML se denomina **XHTML 1.0** (<http://www.w3.org/TR/xhtml1/>) y se publicó el 26 de Enero de 2000 (y posteriormente se revisó el 1 de Agosto de 2002).

XHTML 1.0 es una adaptación de HTML 4.01 al lenguaje XML, por lo que mantiene casi todas sus etiquetas y características, pero añade algunas restricciones y elementos propios de XML. La segunda versión **XHTML 1.1** (<http://www.w3.org/TR/xhtml11/>) fue publicada el 23 de noviembre de 2010 y pretende modularizar XHTML. También fue publicado el **borrador de XHTML 2.0** (<http://www.w3.org/TR/xhtml2/>) (16 de diciembre de 2010), que supondría un cambio muy importante respecto de las anteriores versiones de XHTML.

1.3. Especificación oficial

El organismo **W3C** (*World Wide Web Consortium*) elabora las normas que deben seguir los diseñadores de páginas web para crear las páginas HTML. Las normas oficiales están escritas en inglés y se pueden consultar de forma gratuita en las siguientes direcciones:

- **Especificación oficial de HTML 4.01** (<http://www.w3.org/TR/html401/>)
- **Especificación oficial de XHTML 1.0** (<http://www.w3.org/TR/xhtml1/>)

El estándar XHTML 1.0 incluye el 95% del estándar HTML 4.01, ya que sólo añade pequeñas mejoras y modificaciones menores. Afortunadamente, no es necesario leer las especificaciones y recomendaciones oficiales de HTML para aprender a diseñar páginas con HTML o XHTML. Las normas oficiales están escritas con un lenguaje

bastante formal y algunas secciones son difíciles de comprender. Por ello, en los próximos apartados se explica de forma sencilla y con decenas de ejemplos la especificación oficial de XHTML.

1.4. HTML, XHTML y HTML5

HTML ha evolucionado de formas nuevas e inesperadas. HTML es el lenguaje original de la Web. La última versión completada (4.01) se formalizó a finales de 1999. en la última década ha sido reemplazado gradualmente por una versión similar pero modernizada denominada XHTML (*Extensible HyperText Markup Language*, Lenguaje extensible de marcado de hipertexto).

Nota: Como XHTML es una variante de HTML y como éste fue el estándar original de la Web, suele decirse que todas las páginas están escritas en HTML (como hacemos en los apuntes de este tema). Al emplearse de este modo, el término, en realidad, significa cualquier versión de HTML o XHTML.

La versión de XHTML usada actualmente es la 1.0 o (menos habitual) la 1.1. Los programadores debían sustituir ambas por un estándar incluso más reciente denominado XHTML 2.

No obstante los desarrolladores de XHTML 2 se perdieron en el intento. Crearon un estándar filosóficamente puro pero prácticamente inservible. Los programadores Web y los fabricantes de los principales navegadores se asustaron del carácter estricto de XHTML 2 y del hecho de su falta de compatibilidad inversa con las páginas Web existentes. Es decir, no se podía convertir una hecha en XHTML 1.0 a una nueva XHTML 2. Es más, el estándar XHTML 2 se produjo cuando un grupo de la competencia comenzó a presentar las versiones iniciales de un estándar denominado HTML5. Aunque es novedoso, diferente y todavía no se admite en ningún navegador, ha generado mucho interés. De hecho, la W3C (la organización de estándares encargada oficialmente de HTML y XHTML) ha abandonado recientemente sus trabajos con el estándar XHTML 2 y ha comenzado a formalizar HTML5, convirtiéndolo en oficial: HTML5 es el futuro de la Web.

Los diseñadores de HTML5 lo hicieron compatible con todas las funciones que admite XHTML en la actualidad, lo que significa que se pueden crear documentos con este lenguaje ya mismo (como haremos en clase), sin necesidad de las nuevas prestaciones que los navegadores todavía no reconocen. Todo funcionará a la perfección. Lo mejor de todo es que estará listo para un futuro no tan lejano, en el que se utilizarán nuevos navegadores y se podrán empezar a usar las atractivas funciones exclusivas de HTML5.

Nota: HTML5 acepta todo lo que admite XHTML, sin necesidad de cambios. Sin embargo, la sintaxis de XHTML es más estricta que la de HTML5, por lo que éste acoge determinadas prácticas que XHTML no acepta. Los programadores web no se ponen de acuerdo en si estas prácticas son propias de un estilo inadecuado o simplemente útiles soluciones. Para acostumbrarnos a los hábitos correctos, en los apuntes usaremos la sintaxis más estricta del estilo XHTML.

Más sobre HTML5: Familiarizarse con HTML5 puede ser todo un reto. Parte del problema es que la palabra “HTML5” incluye la última versión de idioma HTML junto a numerosos estándares emergentes que siguen en fase de desarrollo. Para que todo sea más interesante, algunos de ellos funcionan con los navegadores actuales más utilizados y otros todavía no se han implementado ni siquiera en las últimas versiones de HTML5. Curiosamente, este lenguaje solo ofrece ciertas novedades a la familia estándar de etiquetas que veremos. Muchas de ellas proporcionan a los navegadores web motores de búsqueda y otras herramientas automatizadas, además de la información estructural sobre sus páginas web. Por ejemplo, se pueden usar para indicar qué sección de tu página web contiene los enlaces de navegación.

HTML5 también ofrece una nueva forma de añadir elementos de vídeo a su página web sin necesidad de un complemento, como sucede en muchos navegadores actuales. El principal objetivo de HTML5 y de sus estándares relacionados es proporcionar un conjunto de potentes instrumentos para desarrollar aplicaciones y páginas web con las que interactuar. Por ejemplo, incluye un lienzo para dibujar en dos dimensiones, una función para determinar la ubicación geográfica de los visitantes y una técnica para que las páginas interactivas funcionen sin conexión. Todas estas herramientas son la vanguardia del diseño web y no son compatibles con versiones de Internet Explorer anteriores a IE 9.

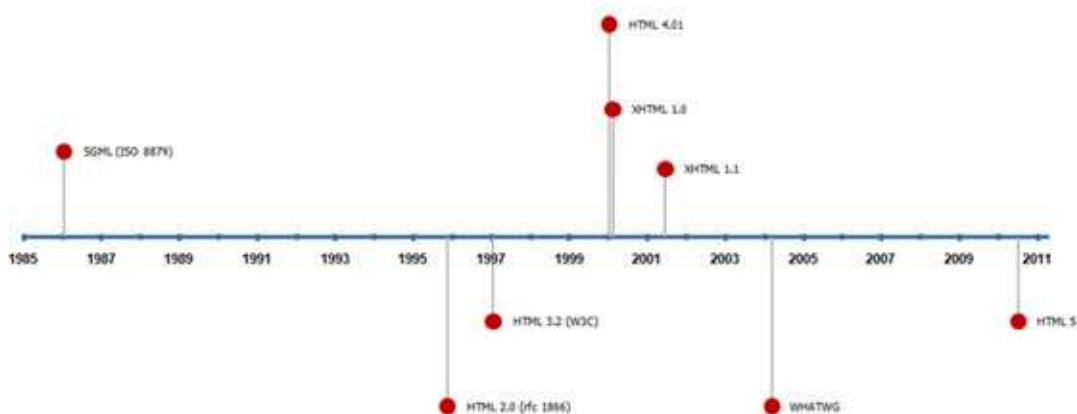


Figura 2.1. Evolución de HTML

1.5. HTML y CSS

Originalmente, las páginas HTML sólo incluían información sobre sus contenidos de texto e imágenes. Con el desarrollo del estándar HTML, las páginas empezaron a incluir también información sobre el aspecto de sus contenidos: tipos de letra, colores y márgenes.

La posterior aparición de tecnologías como JavaScript, provocaron que las páginas HTML también incluyeran el código de las aplicaciones (llamadas *scripts*) que se utilizan para crear páginas web dinámicas.

Incluir en una misma página HTML los contenidos, el diseño y la programación complica en exceso su mantenimiento. Normalmente, los contenidos y el diseño de la página web son responsabilidad de diferentes personas, por lo que es conveniente separarlos.

CSS es el mecanismo que permite separar los contenidos definidos mediante XHTML y el aspecto que deben presentar esos contenidos:

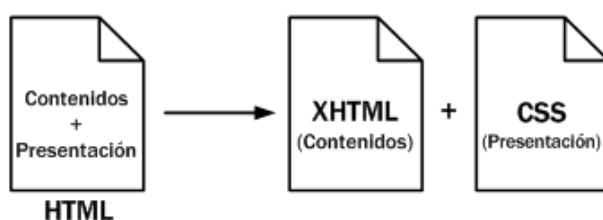


Figura 2.2. Esquema de la separación de los contenidos y su presentación

Una ventaja añadida de la separación de los contenidos y su presentación es que los documentos XHTML creados son más flexibles, ya que se adaptan mejor a las diferentes plataformas: pantallas de ordenador, pantallas de dispositivos móviles, impresoras y dispositivos utilizados por personas discapacitadas.

De esta forma, utilizando exclusivamente XHTML se crean páginas web "*feas*" pero correctas. Aplicando CSS, se pueden crear páginas "*bonitas*" a partir de las páginas XHTML correctas.

2. CARACTERÍSTICAS BÁSICAS

2.1. Etiquetas HTML

Uno de los retos iniciales a los que se tuvo que enfrentar la informática fue el de cómo almacenar la información en los archivos digitales. Como los primeros archivos sólo

contenían texto sin formato, la solución utilizada era muy sencilla: se codificaban las letras del alfabeto y se transformaban en números.

De esta forma, para almacenar un contenido de texto en un archivo electrónico, se utiliza una tabla de conversión que transforma cada carácter en un número. Una vez almacenada la secuencia de números, el contenido del archivo se puede recuperar realizando el proceso inverso.

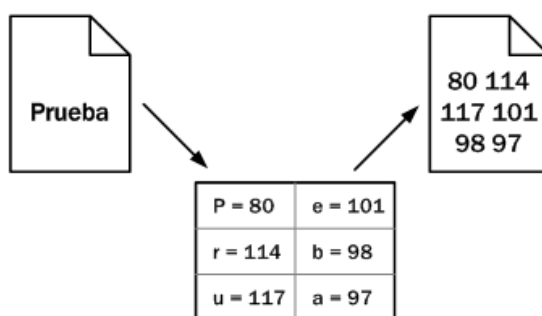


Figura 2.3. Ejemplo sencillo de codificación de caracteres

El proceso de transformación de caracteres en secuencias de números se denomina **codificación de caracteres** y cada una de las tablas que se han definido para realizar la transformación se conocen con el nombre de **páginas de código**. Una de las codificaciones más conocidas (y una de las primeras que se publicaron) es la codificación ASCII. La importancia de las codificaciones en HTML se verá más adelante.

Una vez resuelto el problema de almacenar el texto simple, se presenta el reto de almacenar los contenidos de texto con formato. En otras palabras, ¿cómo se almacena un texto en negrita? ¿y un texto de color rojo? ¿y otro texto azul, en negrita y subrayado?

Utilizar una tabla de conversión similar a las que se utilizan para textos sin formato no es posible, ya que existen infinitos posibles estilos para aplicar al texto. Una solución técnicamente viable consiste en almacenar la información sobre el formato del texto en una zona especial reservada dentro del propio archivo. En esta zona se podría indicar dónde comienza y dónde termina cada formato.

No obstante, la solución que realmente se emplea para guardar la información con formato es mucho más sencilla: el archivo electrónico almacena tanto los contenidos como la información sobre el formato de esos contenidos. Si por ejemplo se quiere dividir el texto en párrafos y se desea dar especial importancia a algunas palabras, se podría indicar de la siguiente manera:

```
<parrafo>  
Contenido de texto con <importante>algunas palabras</importante>  
resaltadas de forma especial.  
</parrafo>
```

El principio de un párrafo se indica mediante la palabra `<parrafo>` y el final de un párrafo se indica mediante la palabra `</parrafo>`. De la misma manera, para asignar más importancia a ciertas palabras del texto, se encierran entre `<importante>` y `</importante>`.

El proceso de indicar las diferentes partes que componen la información se denomina **marcar** (*markup* en inglés). Cada una de las palabras que se emplean para marcar el inicio y el final de una sección se denominan **etiquetas**.

Aunque existen algunas excepciones, en general las etiquetas se indican por pares y se forman de la siguiente manera:

- **Etiqueta de apertura:** carácter `<`, seguido del nombre de la etiqueta (sin espacios en blanco) y terminado con el carácter `>`
- **Etiqueta de cierre:** carácter `<`, seguido del carácter `/`, seguido del nombre de la etiqueta (sin espacios en blanco) y terminado con el carácter `>`

Así, la estructura típica de las etiquetas HTML es:

```
<nombre_etiqueta> ... </nombre_etiqueta>
```

HTML es un **lenguaje de etiquetas** (también llamado **lenguaje de marcado**) y las páginas web habituales están formadas por cientos o miles de pares de etiquetas. De hecho, las letras "ML" de la sigla HTML significan "*markup language*", que es como se denominan en inglés a los *lenguajes de marcado*. Además de HTML, existen muchos otros lenguajes de etiquetas como XML, SGML, DocBook y MathML.

La principal ventaja de los lenguajes de etiquetas es que son muy sencillos de leer y escribir por parte de las personas y de los sistemas electrónicos. La principal desventaja es que pueden aumentar mucho el tamaño del documento, por lo que en general se utilizan etiquetas con nombres muy cortos.

Nota: Añadir etiquetas a un texto sin formato se llama marcar un documento y las etiquetas se conocen como marcado HTML. Cuando se observa un texto HTML sin formato, se puede estar interesado en ver el contenido (el texto que hay entre las etiquetas) o en el marcado (las etiquetas HTML en sí).

2.1.1. Comprender los elementos

Como hemos visto, las etiquetas van en parejas. Cuando utilizamos una inicial (como `` para la negrita) debemos incluir también una final (como ``). Esta combinación de inicio y fin y el texto entre ellas forman un elemento HTML.

Es la idea básica: **los elementos son contenedores**. Se coloca cierto contenido (como texto) dentro de uno de ellos. Por ejemplo, cuando empleamos las etiquetas `` y ``, se crea un contenedor que aplica negrita al texto que hay en medio. Se pone el texto dentro de él para que tenga esa apariencia. Cuando crees tus páginas web, envolverás distintas partes del texto en diferentes contenedores que hacen diversas cosas. Si piensas en los elementos de esta forma, nunca olvidarás incluir la etiqueta final.

```
<b> ¡Atención! </b>
```

Nota: Cuando alguien se refiere al elemento ``, hace mención a todo; etiqueta de inicio, etiqueta final y contenido. Si alguien habla de la etiqueta ``, simplemente se refiere a la instrucción que da inicio al elemento.

Por supuesto, hay excepciones. Cuando uno se adentra en ellos, en realidad hay dos tipos de elementos:

- **Elementos contenedores:** El elemento contenedor es, de lejos, el más común. Aplica formato al contenido anidado entre las etiquetas inicial y final.
- **Elementos independientes:** No activan o desactivan el formato. En lugar de eso, insertan algo en la página, como una imagen. Un ejemplo es el elemento `
` que incorpora un salto de línea en una página. Estos elementos no se muestran en parejas, como los contenedores, y suelen llamarse elementos vacíos porque no se puede colocar texto en su interior.

En los apuntes, todos los elementos independientes incluyen una barra antes del signo de cierre (`>`), una especie de etiqueta de inicio y de fin en un mismo objeto. Por ello, el salto de línea se ve como `
` en lugar de `
`. Este formato, denominado **sintaxis de elementos vacíos**, es muy útil, ya que separa claramente los contenedores de los independientes. De ese modo, se evitan confusiones.

Nota: En un pasado no muy lejano, los programadores Web tenían que usar la sintaxis de elementos vacíos, dado que era una parte oficial del lenguaje XHTML. No obstante, en HTML5 es opcional y permite a los independientes usar la misma sintaxis de las etiquetas de inicio (así, se puede emplear `
` o `
` para añadir un salto de línea).

2.2. El documento HTML

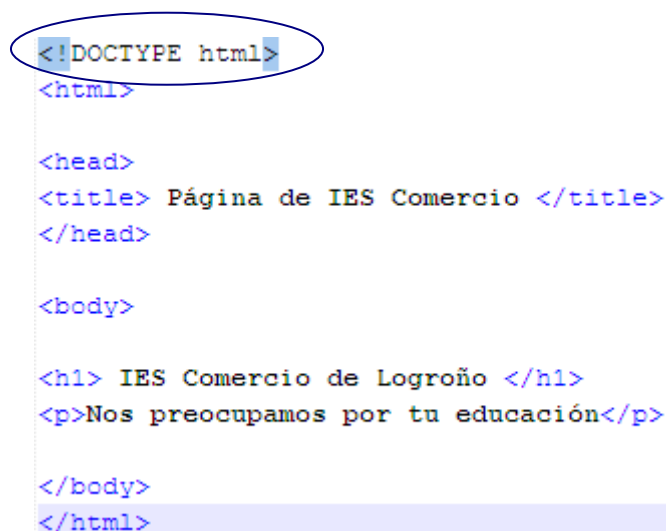
Hasta ahora hemos visto fragmentos de HTML, partes de un documento HTML. En este punto aprenderemos a ponerlo todo junto y a generar nuestra primera página web.

2.2.1. La definición de tipo de documento

En los albores de Internet, los navegadores Web sufrían numerosos defectos. Cuando los usuarios diseñaban páginas Web, tenían que tenerlos en cuenta. Por ejemplo, podían calcular los márgenes alrededor de los cuadros flotantes de texto de diversas maneras, de modo que las páginas se veían bien en un navegador pero no en otros.

Años después, las reglas de HTML y CSS (el estándar de hojas de estilo que veremos en el siguiente tema), se estandarizaron con mayor detalle. Con ellas, todos los navegadores podían mostrar la misma página de igual forma. Pero este cambio supuso un serio problema para los navegadores ya asentados que habían existido en la época oscura de HTML, como era el caso de Internet Explorer. Tenían que admitir los nuevos estándares y seguir enseñando correctamente las páginas Web existentes, incluidas las que dependían de los antiguos defectos.

La comunidad Web adoptó una solución sencilla. Al diseñar una nueva página Web moderna, se indica este hecho mediante la inclusión de un código denominado **definición de tipo de documento** (*Document Type Definition, DTD*), que se sitúa al inicio del documento HTML (véase la figura 2.4)



```
<!DOCTYPE html>
<html>

  <head>
    <title> Página de IES Comercio </title>
  </head>

  <body>

    <h1> IES Comercio de Logroño </h1>
    <p>Nos preocupamos por tu educación</p>

  </body>
</html>
```

El código de ejemplo muestra la estructura básica de un documento HTML. La primera línea, `<!DOCTYPE html>`, está circunscrita por un óvalo azul, indicando su importancia como la definición de tipo de documento. El resto del código define un documento con un título, un encabezado, un cuerpo con un título principal y un párrafo, y finalmente el cierre de las etiquetas de nivel superior.

Figura 2.4. La definición de tipo de documento es la primera información de un archivo HTML e indica al navegador qué estándar de marcado se usa para crear la página.

Cuando un navegador detecta una página con una DTD, cambia a modo de estándares. Tras ello, la representa de la forma más coherente estandarizada posible. El resultado final es una página con el mismo aspecto en todos los navegadores modernos.

Pero cuando descubre un documento HTML sin una DTD, aparecen los problemas. Algunos, como Internet Explorer, cambian al temido modo de defectos. En él, IE intenta comportarse como hacían versiones anteriores de hace 10 años. Así, las páginas Web antiguas tienen el mismo aspecto que cuando se crearon inicialmente, aunque dependan de errores de navegador antiguos que ya se han solucionado. Desafortunadamente, cada uno tiene su idiosincrasia. Por ello, al ver una página sin una DTD, es probable que experimente diferentes tamaños de texto, márgenes y bordes incoherentes y contenido colocado de manera incorrecta. Por ese motivo, las páginas Web sin DTD no son recomendables y debes evitarlas a toda costa.

Los diseñadores Web pueden usar distintas DTD para indicar el estándar de marcado usado (como por ejemplo, HTML, XHTML, HTML5 o HTML 4.01). Sin embargo, los navegadores tienen un secreto: no les importa lo que diga la DTD. Simplemente, quieren que se incluya una. Se debe a que la mayoría de las DTD activan el modo de estándares. Por ello, puedes asignar una DTD XHTML 1.0 a tu página web pero incluir contenido HTML5 en su interior. A tu navegador no le importará. En la actualidad, muchos programadores Web usan la DTD HTML5, que posee este aspecto:

```
<!DOCTYPE html>
```

Comprobarás que no indica el número de versión (5); en su lugar, solo señala que el lenguaje es HTML. No es un error. Refleja la filosofía de HTML: admitir documentos antiguos y nuevos. También significa que al añadir nuevas funciones todas estarán automáticamente disponibles en sus páginas Web existentes. Así funcionan los navegadores. Con HTML5, simplemente se oficializa.

Como comparación, veamos la DTD de XHTML 1.0, mucho más compleja, que puedes ver en páginas Web antiguas:

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

Incluso los programadores Web más experimentados tenían que copiar la DTD XHTML 1.0 de una página Web ya existente para evitar errores.

En los apuntes usaremos la DTD HTML5. Aunque no hay navegadores que admitan todas sus novedades (y algunas versiones no lo admiten en absoluto), todos la aceptan y activan el modo de estándares cuando la detectan. Significa que cualquier página Web con la DTD HTML5 se representa de la misma forma que si tuviera la DTD XHTML 1.0 tradicional.

Nota: La ventaja de usar la DTD HTML5 es que prepara tus páginas para el futuro. Sin embargo, solo por incluirla, no asumas que puedes emplear sus funciones específicas. De hecho, debes evitar muchas de ellas por el momento, ya que no se admiten de modo generalizado.

2.2.2. El esqueleto básico

Una vez seleccionado el tipo de documento, ya se puede completar el resto de la página Web. Para crear un verdadero documento HTML comenzaremos con tres elementos contenedores: `<html>`, `<head>` y `<body>`. Estos tres se combinan para describir la estructura básica de la página.

- `<html>`: Este elemento envuelve todo el contenido de la página (excepto la DTD). En el interior de la etiqueta `<html>` se definen la cabecera y el cuerpo del documento HTML y todo lo que se coloque fuera de la etiqueta `<html>` se ignora.
- `<head>`: Este elemento designa la parte de la cabecera del documento. Puede incluir información opcional sobre la página Web, como puede ser el título, que se indica con la etiqueta `<title>`, (el navegador lo muestra en la barra del título), el idioma de la página, palabras clave de búsqueda opcionales y una hoja de estilo opcional. Los contenidos indicados en la cabecera no son visibles para el usuario, con la excepción de la etiqueta `<title>`.
- `<body>`: Este elemento alberga el contenido de la página Web. El cuerpo encierra todos los contenidos que se muestran al usuario (párrafos de texto, imágenes, tablas). En general, el `<body>` de un documento contiene cientos de etiquetas HTML, mientras que el `<head>` no contiene más que unas pocas.

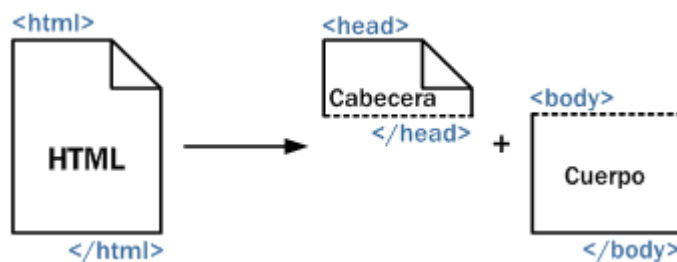


Figura 2.5. Esquema de las etiquetas principales que contiene un documento HTML

Sólo hay una manera correcta de combinar estos tres elementos. He aquí su colocación exacta, con el *doctype* al comienzo de la página:

```
<!DOCTYPE html>

<html>

<head>
...
</head>

<body>
...
</body>

</html>
```

Toda página Web utiliza esta estructura básica. Los puntos suspensivos (...) muestran dónde se insertaría la información adicional. Los espacios entre líneas no son necesarios, sólo están ahí para ver la estructura más fácilmente.

Una vez colocado el esqueleto XHTML, hay que añadir dos contenedores más. Toda página Web requiere un elemento `<title>` en la sección del encabezado. A continuación, habrá que crear un contenedor para el texto en la sección del cuerpo de texto (`<body>`). Un elemento contenedor de texto multiuso es `<p>`, que representa un párrafo. Veamos con más detalle los elementos que hay que agregar:

- `<title>`: Establece el título de la página Web, el cual tiene varias funciones. Primero, los navegadores lo muestran en la parte superior de la ventana. Segundo, cuando un visitante crea un marcador para la página, el navegador emplea el título para etiquetarlo en el menú *Marcadores* (o *Favoritos*). Tercero, cuando la página aparece en una búsqueda Web, el motor de búsqueda suele enseñar este título como primera línea en los resultados, seguido de un fragmento de contenido de la página.
- `<p>`: Indica un párrafo. Los navegadores no los sangran pero añaden un pequeño espacio entre varios consecutivos para mantenerlos separados.

He aquí la página Web con estos dos nuevos ingredientes:

```
<!DOCTYPE html>

<html>

<head>
<title>Todo lo que sé sobre Diseño Web</title>
</head>
```

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

</html>
```

Si abres este documento en un navegador Web, verás que la página está vacía pero ahora aparece el título.

Tal y como está ahora, este documento HTML es una buena plantilla para futuras páginas. La estructura básica está en su lugar; simplemente hay que cambiar el título y añadir algo de texto.

A continuación se muestra el código HTML de una página web muy sencilla:

```
<!DOCTYPE html>

<html>
<head>
<title>El primer documento HTML</title>
</head>

<body>
<p>El lenguaje HTML es <strong>tan sencillo</strong> que
prácticamente se entiende sin estudiar el significado
de sus etiquetas principales.</p>
</body>

</html>
```

Si quieres probar este primer ejemplo, debes hacer lo siguiente:

1. Abre un editor de archivos de texto y crea un archivo nuevo
2. Copia el código HTML mostrado anteriormente y pégalo tal cual en el archivo que has creado
3. Guarda el archivo con el nombre que quieras, pero con la extensión `.html`

Para que el ejemplo anterior funcione correctamente, es imprescindible que utilices un editor de texto sin formato. Si tu sistema operativo es Windows, puedes utilizar el *Bloc de notas*, *Wordpad*, *EmEditor*, *UltraEdit*, *Notepad++*, etc. pero no puedes utilizar un procesador de textos como *Word* o *Open Office*. Si utilizas sistemas operativos tipo Linux, puedes utilizar editores como *Gedit*, *Kedit*, *Kate* e incluso *Vi*, pero no utilices *KOffice* ni *Open Office*.

Después de crear el archivo con el contenido HTML, ya se puede abrir con cualquier navegador para que se muestre con el siguiente aspecto:

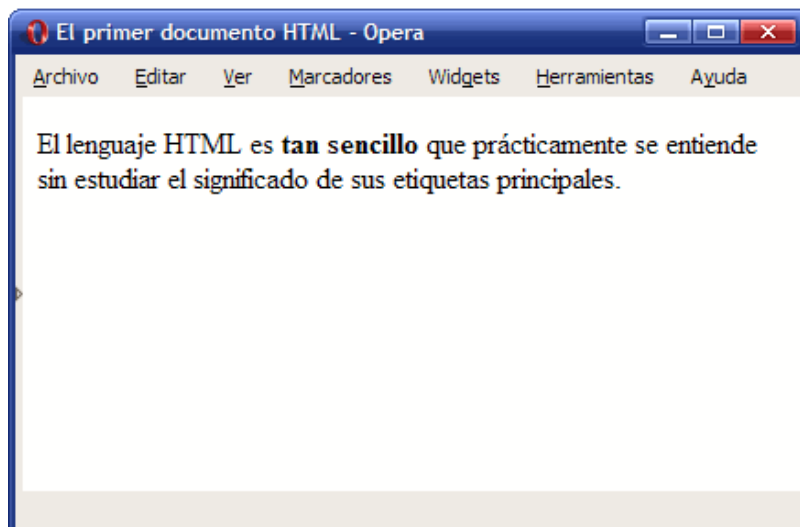


Figura 2.6. Aspecto que muestra el primer documento HTML en cualquier navegador

Si ya estás viendo tu primera página HTML en el navegador, prueba a pulsar sobre el menú **Ver > Código fuente** y podrás ver el código HTML de la página que está cargada en el navegador. De hecho, puedes ver el código HTML de cualquier página de Internet mediante la opción **Ver > Código fuente**. Prueba a ver el código HTML de tu página preferida y verás cuántas etiquetas puede llegar a tener una página compleja.

Ejercicio 1: Determinar el código HTML correspondiente a la siguiente página:

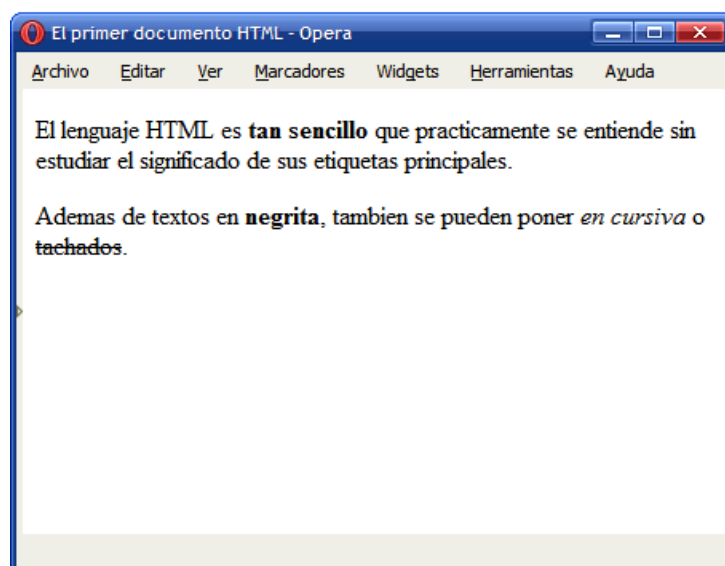


Figura 2.7. Página HTML sencilla que resalta algunas partes del texto

Pistas: `` y ``

2.3. Etiquetas y atributos

HTML define las siguientes etiquetas que los diseñadores pueden utilizar para *marcar* los diferentes elementos que componen una página:

En **amarillo** aquellas etiquetas introducidas en esta la versión 5, en **azul** las etiquetas que han sido cambiadas todo o en parte y en **gris** las etiquetas eliminadas de esta versión. Si bien en la práctica los navegadores no lo están teniendo en cuenta para evitar perder cuota de mercado.

Etiqueta	Atributos	Comentarios
<!-- -->	Estándar o ninguno	
<!DOCTYPE>	Estándar o ninguno	
<a>	href target rel hreflang media type	Atributo Añadido: <i>media</i> Atributo cambiado: Target
<abbr>	Estándar o ninguno	
<acronym>		Etiqueta Eliminada
<address>	Estándar o ninguno	
<applet>		Etiqueta eliminada
<area>	Estándar o ninguno	
<article>	Atributos globales	Nueva etiqueta
<aside>	Atributos globales	Nueva etiqueta
<audio>	autobuffer autoplay controls loop src	Nueva etiqueta
	Atributos globales	Etiqueta cambiada
<base>	Estándar o ninguno	
<basefont>		Etiqueta eliminada
<bb>	Estándar o ninguno	
<bdo>	Estándar o ninguno	
<big>		Etiqueta eliminada
<blockquote>	Estándar o ninguno	
<body>	Estándar o ninguno	
 	Estándar o ninguno	
<button>	Estándar o ninguno	
<canvas>	height width	Nueva etiqueta
<caption>	Estándar o ninguno	
<center>		Etiqueta eliminada
<cite>	Atributos globales	Etiqueta cambiada
<code>	Estándar o ninguno	
<col>	Estándar o ninguno	

<colgroup>	Estándar o ninguno	
<command>	checked default disabled hidden icon label radiogroup type	Nueva etiqueta
<datagrid>	Estándar o ninguno	
<datalist>	Atributos globales	Nueva etiqueta
<dd>	Estándar o ninguno	
	Estándar o ninguno	
<details>	open	Nueva etiqueta
<dialog>	Atributos globales	Nueva etiqueta
<dir>		Etiqueta eliminada
<div>	Estándar o ninguno	
<dfn>	Estándar o ninguno	
<dl>	Estándar o ninguno	
<dt>	Estándar o ninguno	
	Estándar o ninguno	
<embed>	height src type width	Nueva etiqueta
<fieldset>	Estándar o ninguno	
<figure>	Atributos globales	Nueva etiqueta
		Etiqueta eliminada
<footer>	Atributos globales	Nueva etiqueta
<form>	Estándar o ninguno	
<frame>		Etiqueta eliminada
<frameset>		Etiqueta eliminada
<h1> ... <h6>	Estándar o ninguno	
<head>	Estándar o ninguno	
<header>	Atributos globales	Nueva etiqueta
<hgroup>	Atributos globales	Nueva etiqueta
<hr>	Ninguno	Etiqueta cambiada
<html>	Estándar o ninguno	
<i>	Ninguno	Etiqueta cambiada
<iframe>	Estándar o ninguno	
	Estándar o ninguno	
<input>	accept alt auto-complete autofocus checked disabled form formaction formenctype formmethod formnovalidate formtarget height list max maxlength min multiple name pattern1 placeholder readonly required size src step type value width	Etiqueta cambiada: Añadidos 13 elementos a type
<ins>	Estándar o ninguno	
<isindex>		Etiqueta eliminada
<kbd>	Estándar o ninguno	
<label>	Estándar o ninguno	
<legend>	Estándar o ninguno	

	Estándar o ninguno	
<link>	Estándar o ninguno	
<mark>	Atributos globales	Nueva etiqueta
<map>	Estándar o ninguno	
<menu>	Estándar o ninguno	
<meta>	Estándar o ninguno	
<meter>	high low max min optimum value	Nueva etiqueta
<nav>	Atributos globales	Nueva etiqueta
<noframes>		Etiqueta eliminada
<noscript>	Estándar o ninguno	
<object>	Estándar o ninguno	
	Estándar o ninguno	
<optgroup>	Estándar o ninguno	
<option>	Estándar o ninguno	
<output>	form	Nueva etiqueta
<p>	Estándar o ninguno	
<param>	Estándar o ninguno	
<pre>	Estándar o ninguno	
<progress>	max value	Nueva etiqueta
<q>		
<ruby>	cite	Nueva etiqueta
<rp>	Atributos globales	Nueva etiqueta
<rt>	Atributos globales	Nueva etiqueta
<s>		Etiqueta eliminada
<samp>	Estándar o ninguno	
<script>	Estándar o ninguno	
<section>	cite	Nueva etiqueta
<select>	Estándar o ninguno	
<small>	Atributos globales	Etiqueta Cambiada
<source>	media src type	Nueva etiqueta
	Estándar o ninguno	
<strike>		Etiqueta eliminada
	Estándar o ninguno	
<style>	Estándar o ninguno	
<sub>	Estándar o ninguno	
<sup>	Estándar o ninguno	
<table>	Estándar o ninguno	

<tbody>	Estándar o ninguno	
<td>	Estándar o ninguno	
<textarea>	Estándar o ninguno	
<tfoot>	Estándar o ninguno	
<th>	Estándar o ninguno	
<thead>	Estándar o ninguno	
<time>	datetime pubdate	Nueva etiqueta
<title>	Estándar o ninguno	
<tr>	Estándar o ninguno	
<tt>		Etiqueta eliminada
<u>	Define texto que debe tener un estilo diferente del texto normal ³	
	Estándar o ninguno	
<var>	Estándar o ninguno	
<video>	src poster autobuffer autoplay loop controls width height	Nueva etiqueta
<xmp>		Etiqueta eliminada

A pesar de que se trata de un número de etiquetas muy grande, no es suficiente para crear páginas complejas. Algunos elementos como las imágenes y los enlaces requieren cierta información adicional para estar completamente definidos.

La etiqueta **<a>** por ejemplo se emplea para incluir un enlace en una página. Utilizando sólo la etiqueta **<a>** no es posible establecer la dirección a la que apunta cada enlace. Como no es viable crear una etiqueta por cada enlace diferente, la solución consiste en personalizar las etiquetas HTML mediante cierta información adicional llamada **atributos**.

De esta forma, se utiliza la misma etiqueta **<a>** para todos los enlaces de la página y se utilizan los atributos para indicar la dirección a la que apunta cada enlace.

```
<!DOCTYPE html>

<html>
<head>
```



```
<title>Ejemplo de atributos en las etiquetas</title>
</head>
<body>
<p>
  Los enlaces son muy fáciles de indicar:
  <a>Soy un enlace incompleto, porque no tengo dirección de
destino</a>.
  <a href="http://www.google.com">Este otro enlace apunta a la
página de Google</a>.
</p>
</body>
</html>
```

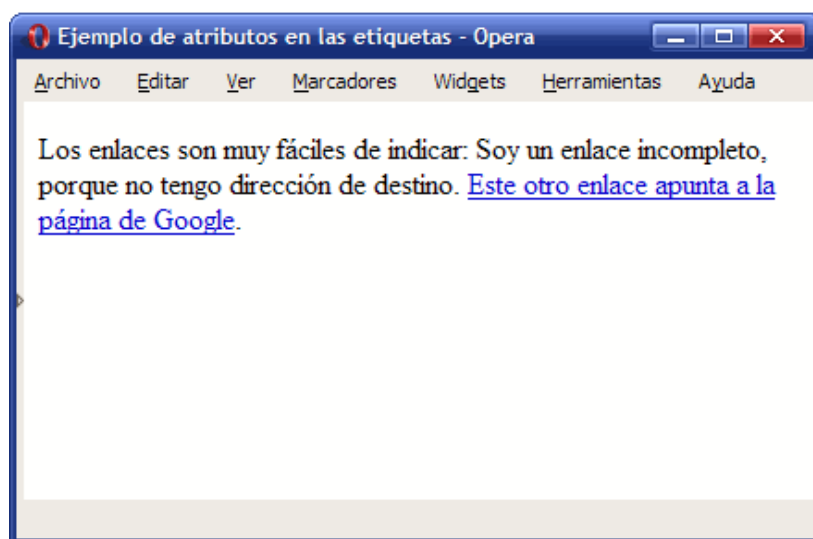


Figura 2.8. Los atributos permiten personalizar las etiquetas HTML

El primer enlace del ejemplo anterior no está completamente definido, ya que no apunta a ninguna dirección. El segundo enlace, utiliza la misma etiqueta `<a>`, pero añade información adicional mediante un atributo llamado `href`. Los atributos se incluyen dentro de la etiqueta de apertura. Por ahora no es importante comprender la etiqueta `<a>` ni el atributo `href`, ya que se explicarán con todo detalle más adelante.

No todos los atributos se pueden utilizar en todas las etiquetas. Por ello, cada etiqueta define su propia lista de atributos disponibles. Además, cada atributo también indica el tipo de valor que se le puede asignar. Si el valor de un atributo no es válido, el navegador ignora ese atributo.

Aunque cada una de las etiquetas HTML define sus propios atributos, algunos de los atributos son comunes a muchas o casi todas las etiquetas. De esta forma, es habitual explicar por separado los atributos comunes de las etiquetas para no tener que volver a hacerlo cada vez que se explica una nueva etiqueta. Los **atributos comunes** se dividen en cuatro grupos según su funcionalidad:

1.- Atributos básicos: se pueden utilizar prácticamente en todas las etiquetas HTML

Atributo	Descripción
<code>id = "texto"</code>	Establece un identificador único a cada elemento dentro de una página HTML
<code>class = "texto"</code>	Establece la clase CSS que se aplica a los estilos del elemento
<code>style = "texto"</code>	Establece de forma directa los estilos CSS de un elemento
<code>title = "texto"</code>	Establece el título a un elemento (mejora la accesibilidad y los navegadores lo muestran cuando el usuario pasa el ratón por encima del elemento)

La mayoría de páginas web actuales utilizan los atributos `id` y `class` de forma masiva. Sin embargo, estos atributos sólo son realmente útiles cuando se trabaja con CSS y con Javascript.

Respecto al valor de los atributos `id` y `class`, sólo pueden contener guiones medios (-), guiones bajos (_), letras y/o números, pero no pueden empezar por números. Además, los navegadores distinguen mayúsculas de minúsculas y no se recomienda utilizar letras como ñ y acentos, ya que no es seguro que funcionen correctamente en todas las versiones de todos los navegadores.

2.- Atributos para internacionalización: los utilizan las páginas que muestran sus contenidos en varios idiomas o aquellas que quieren indicar de forma explícita el idioma de sus contenidos:

Atributo	Descripción
<code>lang = "codigo de idioma"</code>	Indica el idioma del elemento mediante un código predefinido
<code>xml:lang = "codigo de idioma"</code>	Indica el idioma del elemento mediante un código predefinido
<code>dir</code>	Indica la dirección del texto (útil para los idiomas que escriben de derecha a izquierda)

En las páginas XHTML, el atributo `xml:lang` tiene más prioridad que `lang` y es obligatorio incluirlo siempre que se incluye el atributo `lang`.

Como la palabra `internacionalización` es muy larga, se suele sustituir por la abreviatura `i18n` (el número `18` se refiere al número de letras que existen entre la letra `i` y la letra `n` de la palabra `internacionalización`).

3.- Atributos de eventos: sólo se utilizan en las páginas web dinámicas creadas con JavaScript.

Atributo	Descripción
<code>onclick</code> , <code>ondblclick</code> , <code>onmousedown</code> , <code>onmouseup</code> , <code>onmouseover</code> , <code>onmousemove</code> , <code>onmouseout</code> , <code>onkeypress</code> , <code>onkeydown</code> , <code>onkeyup</code>	Permiten controlar los eventos producidos sobre cada elemento de la página

Cada vez que el usuario pulsa una tecla, mueve su ratón o pulsa cualquier botón del ratón, se produce un evento dentro del navegador. Utilizando JavaScript y los atributos anteriores, es posible responder de forma adecuada a cada evento.

4.- Atributos para los elementos que pueden obtener el foco:

Cuando el usuario selecciona un elemento de la interfaz de una aplicación, se dice que *"el elemento tiene el foco del programa"*. Si por ejemplo un usuario pincha con su ratón sobre un cuadro de texto y comienza a escribir, ese cuadro de texto tiene el foco del programa, llamado *"focus"* en inglés. Si el usuario selecciona después otro elemento, el elemento original pierde el foco y el nuevo elemento es el que tiene el foco del programa.

Los elementos de las páginas web también pueden obtener el foco de la aplicación (en este caso, el foco del navegador) y HTML define algunos atributos específicos para controlar cómo se seleccionan los elementos.

Atributo	Descripción
<code>accesskey = "letra"</code>	Establece una tecla de acceso rápido a un elemento HTML
<code>tabindex = "numero"</code>	Establece la posición del elemento en el orden de tabulación de la página. Su valor debe estar comprendido entre <code>0</code> y <code>32.767</code>
<code>onfocus</code> , <code>onblur</code>	Controlan los eventos JavaScript que se ejecutan cuando el elemento obtiene o pierde el foco

Cuando se pulsa repetidamente la tecla del tabulador sobre una página web, el navegador selecciona de forma alternativa todos los elementos de la página que se pueden seleccionar (principalmente los enlaces y los elementos de formulario). El

atributo `tabindex` permite alterar el orden en el que se seleccionan los elementos, por lo que es muy útil cuando se quiere controlar de forma precisa cómo se seleccionan los campos de un formulario complejo.

Por su parte, el atributo `accesskey` permite establecer una tecla para acceder de forma rápida a cualquier elemento. Aunque la tecla de acceso rápido se establece mediante HTML, la combinación de teclas necesarias para activar ese acceso rápido depende del navegador. En el navegador Internet Explorer se pulsa la tecla `ALT` + la tecla definida; en el navegador Firefox se pulsa `Alt` + `Shift` + la tecla definida; en el navegador Opera se pulsa `Shift` + `Esc` + la tecla definida; en el navegador Safari se pulsa `Ctrl` + la tecla definida.

En el resto del tema, se emplearán las palabras "`básicos`", "`i18n`", "`eventos`" y "`foco`" respectivamente para referirse a cada uno de los grupos de atributos comunes definidos anteriormente.

Nuevos atributos globales en HTML5

Los siguientes atributos globales pueden ser usados sobre cualquier elemento HTML5. Sin embargo, no todos los navegadores principales los soportan, e incluso puede haber algún atributo no soportado en ninguno de ellos.

Atributo	Descripción
<code>contenteditable = "true false inherit"</code>	Especifica si el contenido de un elemento es editable o no.
<code>contextmenu="menu_id"</code>	Especifica un menú contextual para un elemento. El menú contextual aparece cuando un usuario hace clic en el elemento.
<code>draggable="true false auto"</code>	Especifica si un elemento es arrastrable o no.
<code>dropzone="copy move link"</code>	Especifica si los datos arrastrados son copiados, movidos o vinculados, cuando se dejan caer sobre un elemento.
<code>hidden= "hidden"</code>	Especifica que un elemento no es visible aún, o no es relevante.
<code>spellcheck="true false"</code>	Especifica si el elemento debe tener su ortografía y gramática comprobada o no.

Para obtener más información sobre los nuevos atributos globales de HTML5 y ver ejemplos: http://www.w3schools.com/tags/ref_standardattributes.asp

2.4. Elementos HTML

Además de etiquetas y atributos, HTML define el término **elemento** para referirse a las partes que componen los documentos HTML.

Aunque en ocasiones se habla de forma indistinta de "elementos" y "etiquetas", en realidad un elemento HTML es mucho más que una etiqueta, ya que está formado por:

- Una etiqueta de apertura.
- Cero o más atributos.
- Texto encerrado por la etiqueta.
- Una etiqueta de cierre.

El texto encerrado por la etiqueta es opcional, ya que algunas etiquetas de HTML no pueden encerrar ningún texto. El siguiente esquema muestra un elemento HTML, formado por una etiqueta `<p>`, atributos y contenidos de texto:

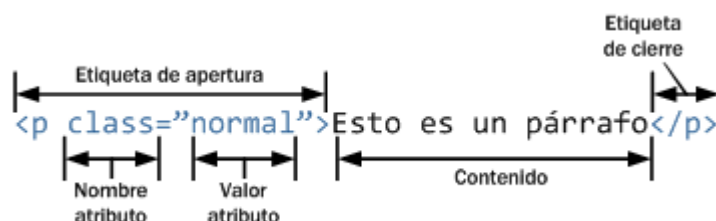


Figura 2.9. Esquema de las partes que componen un elemento HTML

La estructura mostrada en el esquema anterior es un elemento HTML ya que comienza con una etiqueta de apertura (`<p>`), contiene cero o más atributos (`class="normal"`), dispone de un contenido de texto (`Esto es un párrafo`) y finaliza con una etiqueta de cierre (`</p>`).

Por tanto, si una página web tiene dos párrafos de texto, la página contiene dos elementos y cuatro etiquetas (dos etiquetas `<p>` de apertura y dos etiquetas `</p>` de cierre). De todas formas, aunque estrictamente no son lo mismo, es habitual intercambiar las palabras "elemento" y "etiqueta".

Por otra parte, el lenguaje HTML clasifica a todos los elementos en dos grupos: **elementos en línea** (*inline elements* en inglés) y **elementos de bloque** (*block elements* en inglés). Para centrarse en la estructura, HTML5 propone nuevos términos. Sugiere que los de línea se denominen **elementos de frase** y los de bloque pasen a llamarse **elementos de flujo**. El objetivo es enfatizar la diferencia de uso y su ubicación, minimizando el impacto en el formato. Sin embargo, los términos elemento

de bloque y elemento de línea son tan comunes que pasará tiempo antes de que se cambien, si la terminología de HTML5 se abre camino.

La principal diferencia entre los dos tipos de elementos es la forma en la que ocupan el espacio disponible en la página. Los elementos de bloque siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea, aunque sus contenidos no lleguen hasta el final de la línea. Por su parte, los elementos en línea sólo ocupan el espacio necesario para mostrar sus contenidos.

Si se considera el siguiente ejemplo:

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de elementos en línea y elementos de
bloque</title>
</head>
<body>
<p>Los párrafos son elementos de bloque.</p>
<a href="http://www.google.com">Los enlaces son elementos en
línea</a>
<p>Dentro de un párrafo, <a href="http://www.google.com">los
enlaces</a>
siguen siendo elementos en línea.</p>
</body>
</html>
```

La siguiente imagen muestra cómo visualizan los navegadores el código HTML anterior (mediante CSS se han añadido bordes que muestran el espacio ocupado por cada elemento):

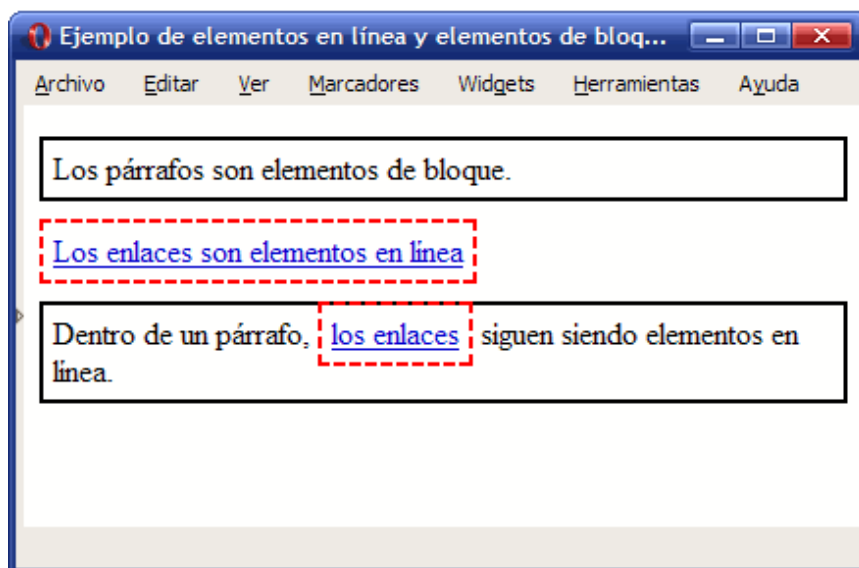


Figura 2.10. Diferencias entre elementos en línea y elementos de bloque

El primer párrafo contiene un texto corto que sólo ocupa la mitad de la anchura de la ventana del navegador. No obstante, el espacio reservado por el navegador para el primer párrafo llega hasta el final de esa línea, por lo que resulta evidente que los elementos `<p>` son elementos de bloque.

Por otra parte, el primer enlace del ejemplo anterior también tiene un texto corto que ocupa solamente la mitad de la anchura de la ventana del navegador. En este caso, el navegador sólo reserva para el enlace el sitio necesario para mostrar sus contenidos. Si se añade otro enlace en esa misma línea, se mostraría a continuación del primer enlace. Por tanto, los elementos `<a>` son elementos en línea.

Por último, el segundo párrafo sigue ocupando todo el espacio disponible hasta el final de cada línea (por ser un elemento de bloque) y el enlace que se encuentra dentro del párrafo sólo ocupa el sitio necesario para mostrar sus contenidos (por ser un elemento en línea).

La mayoría de elementos de bloque pueden contener en su interior elementos en línea y otros elementos de bloque. Los elementos en línea sólo pueden contener texto u otros elementos en línea. En otras palabras, un elemento de bloque no puede aparecer dentro de un elemento en línea. En cambio, un elemento en línea puede aparecer dentro de un elemento de bloque y dentro de otro elemento en línea.

Los elementos en línea definidos por HTML son: `a`, `abbr`, `acronym`, `b`, `basefont`, `bdo`, `big`, `br`, `cite`, `code`, `dfn`, `em`, `font`, `i`, `img`, `input`, `kbd`, `label`, `q`, `s`, `samp`, `select`, `small`, `span`, `strike`, `strong`, `sub`, `sup`, `textarea`, `tt`, `u`, `var`.

Los elementos de bloque definidos por HTML son: `address`, `blockquote`, `center`, `dir`, `div`, `dl`, `fieldset`, `form`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `hr`, `isindex`, `menu`, `noframes`, `noscript`, `ol`, `p`, `pre`, `table`, `ul`.

En HTML5, en la práctica son elementos block, o pueden considerarse como tal: `article`, `aside`, `audio`, `canvas`, `figcaption`, `figure`, `footer`, `header`, `hgroup`, `output`, `section`, `video`.

Los siguientes elementos también se considera que son de bloque: `dd`, `dt`, `frame-set`, `li`, `tbody`, `td`, `tfoot`, `th`, `thead`, `tr`.

Los siguientes elementos pueden ser en línea y de bloque según las circunstancias: `button`, `del`, `iframe`, `ins`, `map`, `object`, `script`.

2.5. Sintaxis de las etiquetas XHTML

El lenguaje HTML original era muy permisivo en su sintaxis, por lo que era posible escribir sus etiquetas y atributos de muchas formas diferentes. Las etiquetas por ejemplo podían escribirse en mayúsculas, en minúsculas e incluso combinando mayúsculas y minúsculas. El valor de los atributos de las etiquetas se podían indicar con y sin comillas ("). Además, el orden en el que se abrían y cerraban las etiquetas no era importante.

La flexibilidad de HTML puede parecer un aspecto positivo, pero el resultado final son páginas con un código HTML desordenado, difícil de mantener y muy poco profesional. Afortunadamente, XHTML soluciona estos problemas añadiendo ciertas normas en la forma de escribir las etiquetas y atributos.

A continuación se muestran las cinco restricciones básicas que introduce XHTML respecto a HTML en la sintaxis de sus etiquetas:

1. Las etiquetas se tienen que cerrar de acuerdo a como se abren:

Ejemplo correcto en XHTML:

```
<p>Este es un párrafo con <a>un enlace</a></p>
```

Ejemplo incorrecto en XHTML (pero correcto en HTML):

```
<p>Este es un párrafo con <a>un enlace</p></a>
```

2. Los nombres de las etiquetas y atributos siempre se escriben en minúsculas:

Ejemplo correcto en XHTML:

```
<p>Este es un párrafo con <a href="http://www.google.com">un  
enlace</a></p>
```

Ejemplo incorrecto en XHTML (pero correcto en HTML):

```
<p>Este es un párrafo con <A HREF="http://www.google.com">un  
enlace</A></P>
```

3. El valor de los atributos siempre se encierra con comillas:

Ejemplo correcto en XHTML:

```
<p>Este es un párrafo con <a href="http://www.google.com">un  
enlace</a></p>
```


Ejemplo incorrecto en XHTML (pero correcto en HTML):

```
<p>Este es un párrafo con <a href=http://www.google.com>un  
enlace</a></p>
```

4. Los atributos no se pueden comprimir:

Ejemplo correcto en XHTML:

```
<dl compact="compact">...</dl>
```

Ejemplo incorrecto en XHTML (pero correcto en HTML):

```
<dl compact>...</dl>
```

Este tipo de atributos en los que el nombre coincide con su valor no son muy habituales.

5. Todas las etiquetas deben cerrarse siempre:

La mayoría de etiquetas HTML encierran un contenido de texto entre la etiqueta de apertura y la etiqueta de cierre. Sin embargo, algunas etiquetas especiales llamadas "*etiquetas vacías*" no necesitan encerrar ningún texto.

La etiqueta `
` por ejemplo, se utiliza para indicar el comienzo de una nueva línea, tal y como se verá más adelante. Por sus características, la etiqueta `
` nunca encierra ningún contenido de texto.

Como el estándar XHTML obliga a cerrar todas las etiquetas abiertas, siempre que se incluya la etiqueta `
` se debería cerrar de forma seguida: `
</br>`. Para que el código resulte más cómodo de escribir, XHTML permite en estos casos escribir de forma abreviada una etiqueta que se abre y se cierra de forma consecutiva.

En lugar de abrir y cerrar de forma consecutiva la etiqueta (`
</br>`) se puede utilizar la sintaxis `
` para indicar que es una etiqueta vacía que se abre y se cierra en ese mismo punto. En la forma compacta es habitual equivocarse con la posición del carácter `/`.

Ejemplo correcto en XHTML:

```
<br/>
```

Ejemplo incorrecto en XHTML (pero correcto en HTML):

```
<br>
```

Además de estas cinco restricciones básicas, XHTML incluye otros cambios más avanzados respecto a HTML:

1. Antes de acceder al valor de un atributo, se eliminan todos los espacios en blanco que se encuentran antes y después del valor. Además, se eliminan todos los espacios en blanco sobrantes dentro del valor de un atributo. En otras palabras, si en el interior de un atributo se incluyen varios espacios en blanco seguidos, se eliminan todos salvo un único espacio en blanco utilizado para separar las diferentes palabras.
2. Como se explicará más adelante al hablar de la etiqueta `<script>`, el código JavaScript debe encerrarse entre unas etiquetas especiales (`<![CDATA[` y `]]>`) para evitar que el navegador interprete de forma errónea caracteres como `&` y `<`.
3. Las páginas XHTML deben prescindir del atributo `name` para identificar de forma única a los elementos. En su lugar, siempre debe utilizarse el atributo `id`. De hecho, en la versión 1.0 del estándar XHTML, el atributo `name` se ha declarado obsoleto para las etiquetas `a`, `applet`, `form`, `frame`, `iframe`, `img` y `map`.

3. TEXTO

La mayor parte del contenido de las páginas HTML habituales está formado por texto, llegando a ser más del 90% del código de la página. Por este motivo, es muy importante conocer los elementos y etiquetas que define HTML para el manejo del texto.

El lenguaje HTML incorpora al tratamiento del texto muchas de las ideas y normas establecidas en otros entornos de publicación de contenidos. De esta forma, HTML define etiquetas para **estructurar** el contenido en secciones y párrafos y define otras etiquetas para **marcar** elementos importantes dentro del texto.

La tarea inicial del editor de contenidos HTML consiste en estructurar el texto original definiendo sus párrafos, titulares y títulos de sección, como se muestra en la siguiente imagen:

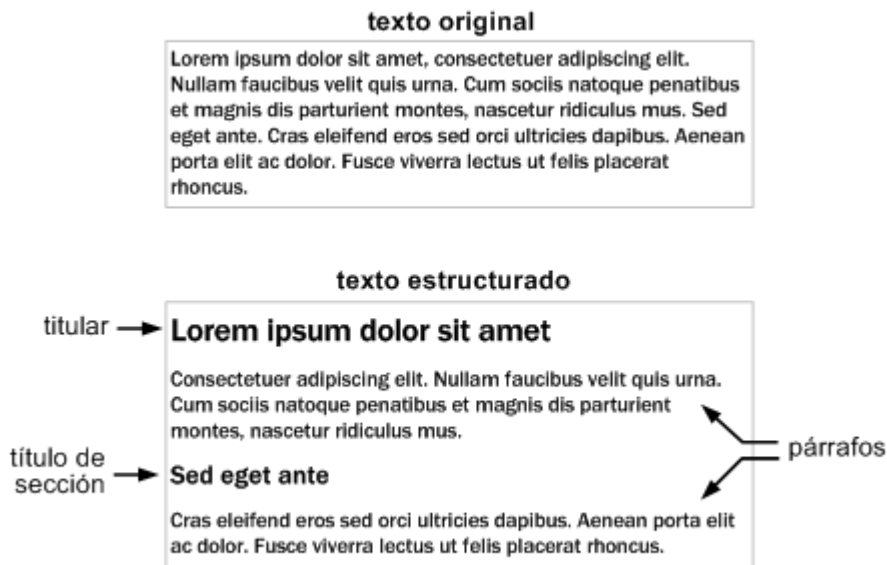


Figura 2.11. Resultado de estructurar un texto sencillo

El proceso de estructurar un texto simple consiste en indicar las diferentes zonas o secciones que componen el texto. De esta forma, los textos estructurados utilizan etiquetas para delimitar cada párrafo y títulos de sección para delimitar cada una de las secciones que forman el texto.

Una vez definida la estructura básica de los contenidos de la página, el siguiente paso consiste en marcar los diferentes elementos dentro del propio texto: definiciones, abreviaturas, textos importantes, textos modificados, citas a otras referencias, etc.



Figura 2.12. Resultado de marcar un texto sencillo

El anterior ejemplo muestra la transformación de un párrafo con un texto simple en un párrafo cuyo texto contiene elementos marcados de forma especial. Así, algunas palabras del texto se muestran en negrita porque se consideran importantes; otras palabras aparecen en cursiva, ya que se han marcado como destacadas e incluso una frase aparece tabulada y entre comillas, indicando que es una cita textual de otro contenido.

En los apartados siguientes se muestran todas las etiquetas que define HTML para estructurar y marcar el texto. Además, se hace una mención especial al tratamiento que hace HTML de los espacios en blanco y las nuevas líneas.

3.1. Estructurar

La forma más sencilla de estructurar un texto consiste en separarlo por párrafos. Además, HTML permite incluir títulos que delimitan cada una de las secciones.

3.1.1. Párrafos

Una de las etiquetas más utilizadas de HTML es la etiqueta `<p>`, que permite definir los párrafos que forman el texto de una página. Para delimitar el texto de un párrafo, se encierra ese texto con la etiqueta `<p>`, como muestra el siguiente ejemplo:

```
<!DOCTYPE html>
<html>

<head>
<title>Ejemplo de texto estructurado con párrafos</title>
</head>

<body>
<p>Este es el texto que forma el primer párrafo de la página.
Los párrafos pueden ocupar varias líneas y el navegador se
encarga de ajustar su longitud al tamaño de la ventana.</p>

<p>El segundo párrafo de la página también se define encerrando
su texto con la etiqueta p. El navegador también se encarga de
separar automáticamente cada párrafo.</p>

</body>

</html>
```

El ejemplo anterior se visualiza de la siguiente manera en cualquier navegador:

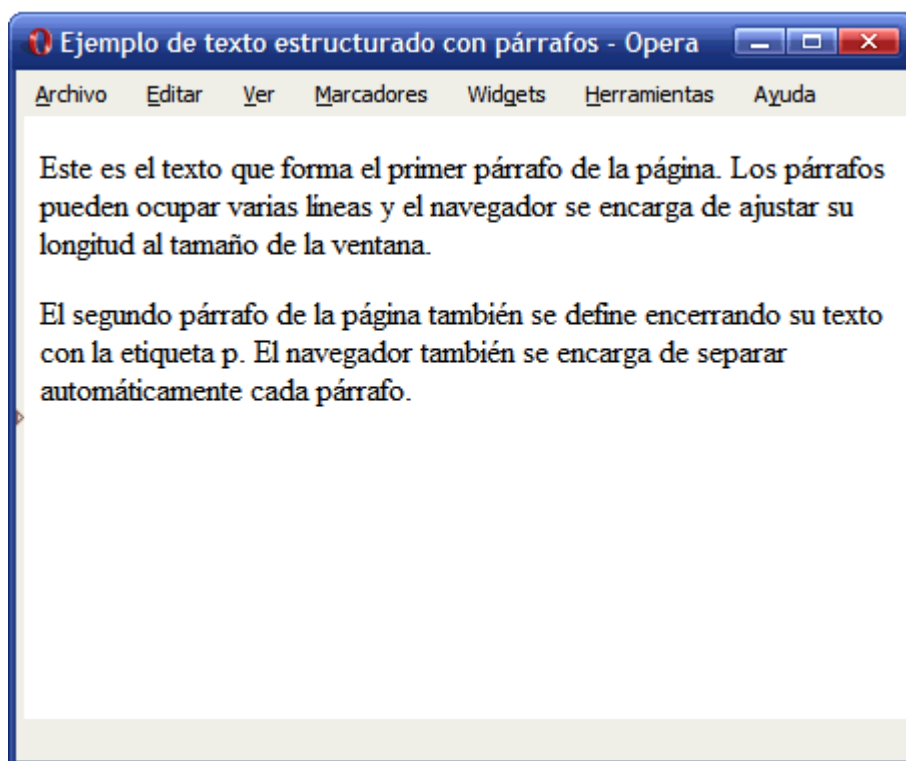


Figura 2.13. Ejemplo de texto HTML estructurado con párrafos

La siguiente tabla recoge la definición formal de la etiqueta `<p>`:

<code><p></code>	Párrafos
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Delimita el contenido de un párrafo de texto

Los párrafos creados con HTML son elementos de bloque, por lo que siempre ocupan toda la anchura de la ventana del navegador. Además, no tienen atributos específicos, pero sí que se les pueden asignar los atributos comunes de HTML básicos, de internacionalización y de eventos.

3.1.2. Secciones

Las páginas HTML habituales suelen tener una estructura más compleja que la que se puede crear solamente mediante párrafos. De hecho, es habitual que las páginas se dividan en diferentes secciones jerárquicas.

Los títulos de sección se utilizan para delimitar el comienzo de cada sección de la página. HTML permite crear secciones de hasta seis niveles de importancia. De esta forma, aunque una página puede definir cualquier número de secciones, sólo puede incluir seis niveles jerárquicos.

Las **etiquetas que definen los títulos de sección** son `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` y `<h6>`. La etiqueta `<h1>` es la de mayor importancia y por tanto se utiliza para definir los titulares de la página. La importancia del resto de etiquetas es descendiente, de forma que la etiqueta `<h6>` es la que se utiliza para delimitar las secciones menos importantes de la página.

A continuación se muestra la definición formal de la etiqueta `<h1>`, siendo idéntica la definición del resto de etiquetas referidas a los títulos de sección:

<code><h1></code>	Sección (titular) de nivel 1
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Define los títulos de las secciones de mayor importancia de la página.

Al igual que la etiqueta `<p>`, las etiquetas de título de sección son elementos de bloque y no tienen atributos específicos.

Las etiquetas `<h1>`, ..., `<h6>` definen títulos de sección, no secciones completas. Por este motivo, no es necesario encerrar los contenidos de una sección con su etiqueta correspondiente. Solamente se debe encerrar con las etiquetas `<h1>`, ..., `<h6>` los títulos de cada sección.

El siguiente ejemplo muestra el uso de las etiquetas de título de sección:

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo de texto estructurado con secciones</title>
</head>

<body>
<h1>Titular de la página</h1>

<p>Párrafo de introducción...</p>

<h2>La primera sub-sección</h2>

<p>Párrafo de contenido...</p>

<h2>Otra subsección</h2>

<p>Más párrafos de contenido...</p>
</body>
</html>
```

Los navegadores muestran el ejemplo anterior de la siguiente manera:

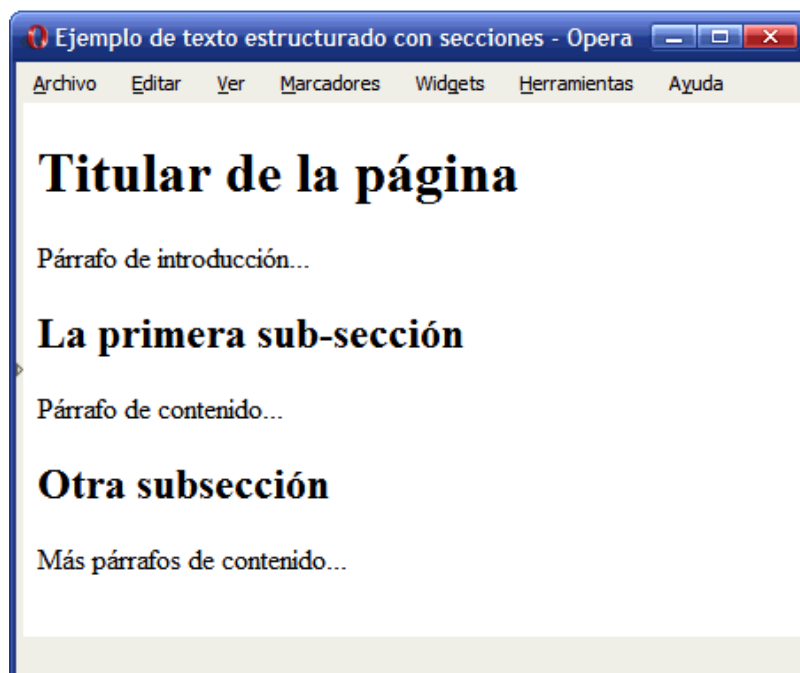


Figura 2.14. Ejemplo de texto HTML estructurado con párrafos y secciones

Los navegadores asignan de forma automática el tamaño del título de cada sección en función de su importancia. Así, los títulos de sección `<h1>` se muestran con el tamaño de letra más grande, ya que son el nivel jerárquico superior, mientras que los títulos de

sección `<h6>` se visualizan con un tamaño de letra muy pequeño, adecuado para el nivel jerárquico de menor importancia.

Evidentemente, el aspecto que los navegadores aplican por defecto a los títulos de sección se puede modificar utilizando las hojas de estilos de CSS. La siguiente imagen muestra el tamaño por defecto con el que los navegadores muestran cada titular:

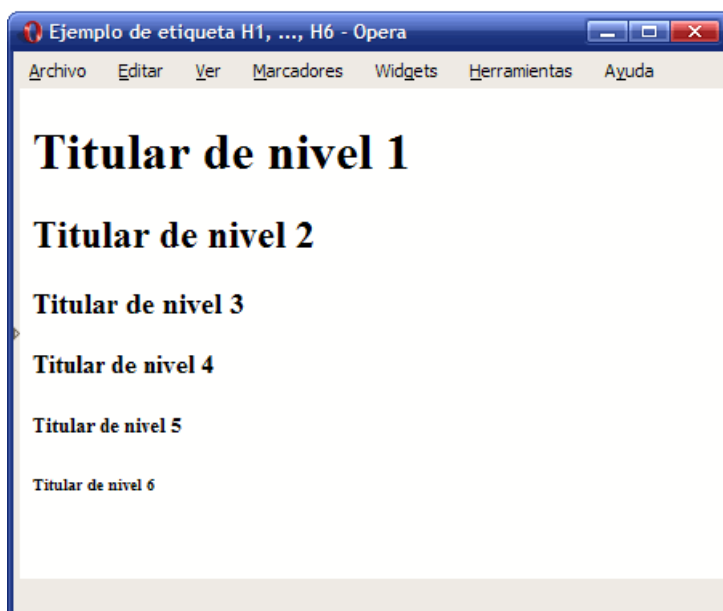


Figura 2.15. Ejemplo de uso de las etiquetas `h1`, `h2`, `h3`, `h4`, `h5` y `h6`

3.2. Marcado básico de texto

Una vez estructurado el texto en párrafos y secciones, el siguiente paso es el marcado de los elementos que componen el texto. Los textos habituales están formados por elementos como palabras en negrita o cursiva, anotaciones y correcciones, citas a otros documentos externos, etc. HTML proporciona varias etiquetas para *marcar* cada uno de los diferentes tipos de texto.

Entre las etiquetas más utilizadas para marcar texto se encuentran `` y ``. La definición formal de estas dos etiquetas se muestra a continuación:

<code></code>	Énfasis
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Realza la importancia del texto que encierra

	Énfasis más acentuado
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Realza con la máxima importancia el texto que encierra

La etiqueta `` marca un texto indicando que su importancia es mayor que la del resto del texto. La etiqueta `` indica que un determinado texto es de la mayor importancia dentro de la página. Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo de etiqueta em y strong</title>
</head>
<body>
<p>El lenguaje HTML permite marcar algunos segmentos de texto
como <em>muy importantes</em> y otros segmentos como <strong>los
más importantes</strong>.</p>
</body>
</html>
```

Por defecto, los navegadores muestran los elementos `` en cursiva para hacer evidente su importancia y muestran los elementos `` en negrita, para indicar que son los más importantes:

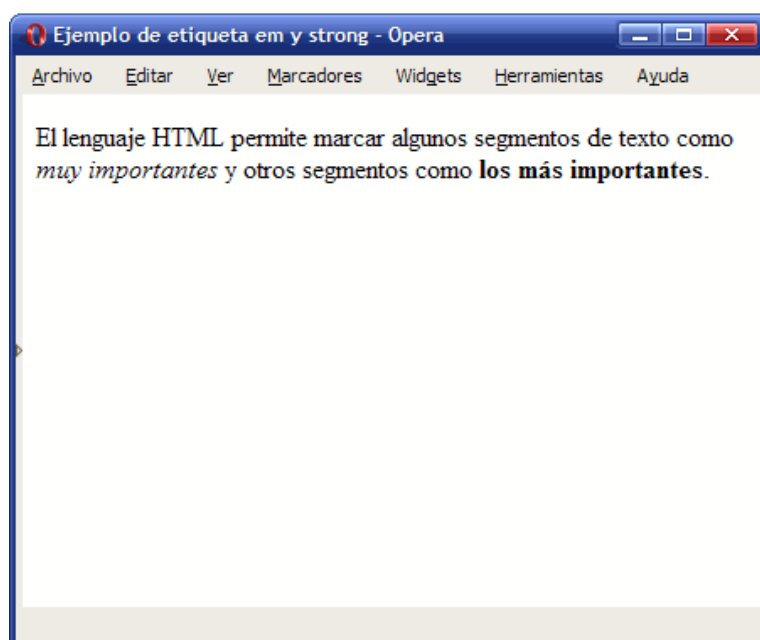


Figura 2.16. Ejemplo de uso de las etiquetas `em` y `strong`

Ejercicio 2: Estructurar y marcar el siguiente texto extraído de la Wikipedia para que el navegador lo muestre con el aspecto de la siguiente imagen:

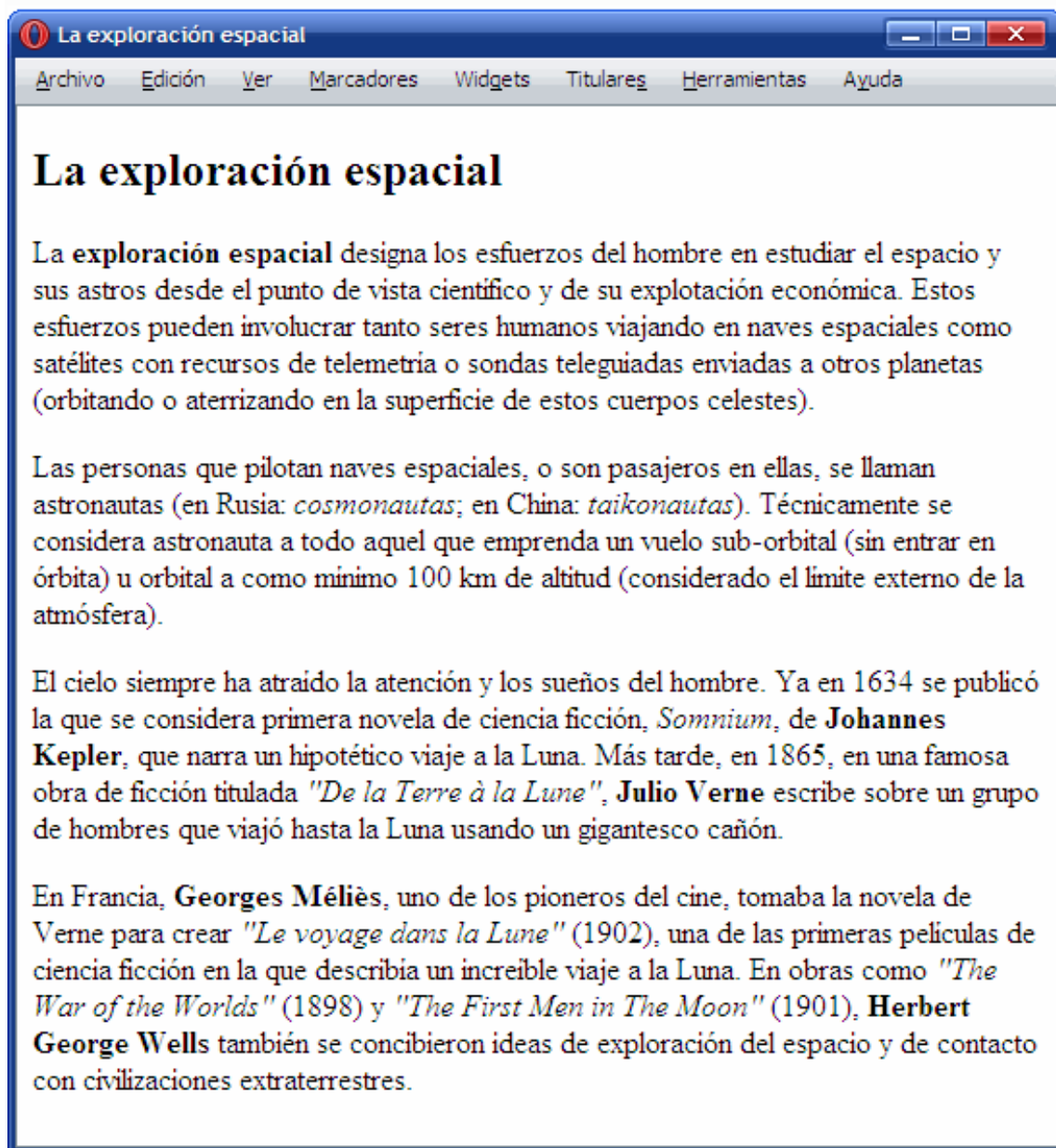


Figura 2.17. Resultado de estructurar y marcar el texto original

HTML también permite marcar de forma adecuada las modificaciones realizadas en el contenido de una página. En otras palabras, HTML permite indicar de forma clara el texto que ha sido eliminado y el texto que ha sido añadido a un determinado texto original. Las etiquetas utilizadas son `<ins>` y ``, cuya definición formal es la siguiente:

<ins>	Inserción
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<p><code>cite = "url"</code> - Indica la URL de la página en la que se puede obtener más información sobre el motivo por el que se realizó la modificación.</p> <p><code>datetime = "fecha"</code> - Especifica la fecha y hora en la que se realizó el cambio</p>
Tipo de elemento	Bloque y en línea
Descripción	Se emplea para marcar una modificación en los contenidos originales consistente en la inserción de un nuevo contenido
	Borrado
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<p><code>cite = "url"</code> - Indica la URL de la página en la que se puede obtener más información sobre el motivo por el que se realizó la modificación.</p> <p><code>datetime = "fecha"</code> - Especifica la fecha y hora en la que se realizó el cambio</p>
Tipo de elemento	Bloque y en línea
Descripción	Se emplea para marcar una modificación en los contenidos originales consistente en el borrado de cierto contenido

Las dos etiquetas cuentan con los mismos atributos específicos, que opcionalmente se pueden añadir para proporcionar más información sobre los cambios realizados. El atributo `cite` se emplea para indicar la dirección de un documento externo en el que se puede encontrar más información relacionada con la inserción o el borrado de texto. El atributo `datetime` puede utilizarse para indicar la fecha y la hora en la que se realizó cada cambio.

Ejemplo:

```
<!DOCTYPE html>

<html>
<head><title>Ejemplo de etiqueta ins y del</title></head>
<body>

<h3>Ejemplo de etiqueta ins y del</h3>

<p>El HTML, acrónimo inglés de Hyper Text Markup Language
(lenguaje de
<del datetime="20091025"
cite="http://www.diw.es/mas_informacion.html">marcado de
hipertexto</del> <ins datetime="20091026"
cite="http://www.diw.es/mas_informacion.html">
marcas hipertextuales</ins>) es un lenguaje de marcación
diseñado para estructurar textos y
presentarlos en forma de hipertexto.</p>

</body>
</html>
```

Los navegadores muestran el ejemplo anterior de la siguiente manera:

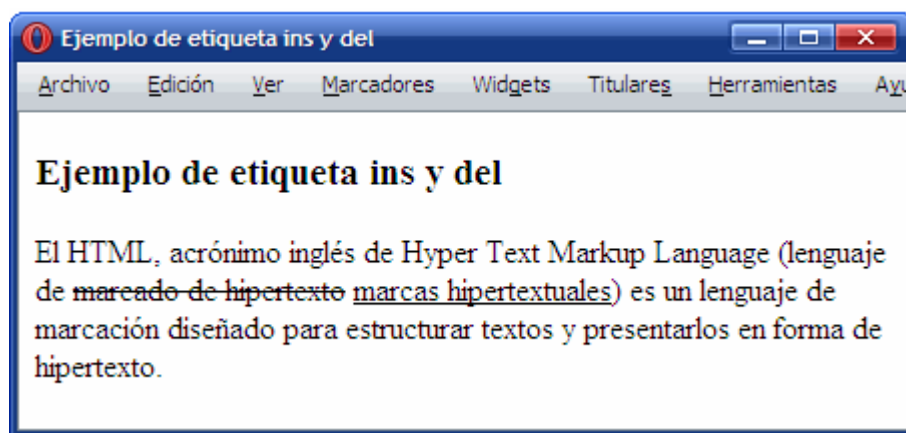


Figura 2.18. Ejemplo de uso de las etiquetas ins y del

Por defecto, el texto eliminado (marcado con la etiqueta ``) se muestra tachado de forma que el usuario pueda identificarlo fácilmente como un texto que formaba parte del texto original y que ya no tiene validez. El texto insertado (marcado con la etiqueta `<ins>`) se muestra subrayado, de forma que el usuario pueda identificarlo como un texto nuevo que no formaba parte del texto original.

Por otra parte, en muchos tipos de páginas (artículos, noticias) es habitual citar literalmente un texto externo. HTML define la etiqueta `<blockquote>` para incluir citas

textuales en las páginas web. La definición de la etiqueta HTML con el nombre más largo se muestra a continuación:

<blockquote>	Citas
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<code>cite = "url"</code> - Indica la dirección de la página web original de la que se extrae la cita
Tipo de elemento	Bloque
Descripción	Se emplea para indicar que el texto que encierra es una cita textual de otro texto externo

Al igual que <ins> y , la etiqueta <blockquote> permite indicar mediante el atributo `cite` la dirección de un documento del que se ha extraído la cita. Ejemplo:

```
<!DOCTYPE html>

<html>
<head><title>Ejemplo de etiqueta blockquote</title></head>

<body>
<p>Según el W3C, el valor del
atributo <em>cite</em> en las etiquetas
<strong>blockquote</strong> tiene el
siguiente significado:</p>

<blockquote
cite="http://www.w3.org/TR/html401/struct/text.html">"El valor
de este atributo
es una dirección URL que indica el documento original de la
cita."</blockquote>
</body>

</html>
```

El aspecto que muestra el ejemplo anterior en cualquier navegador es el siguiente:

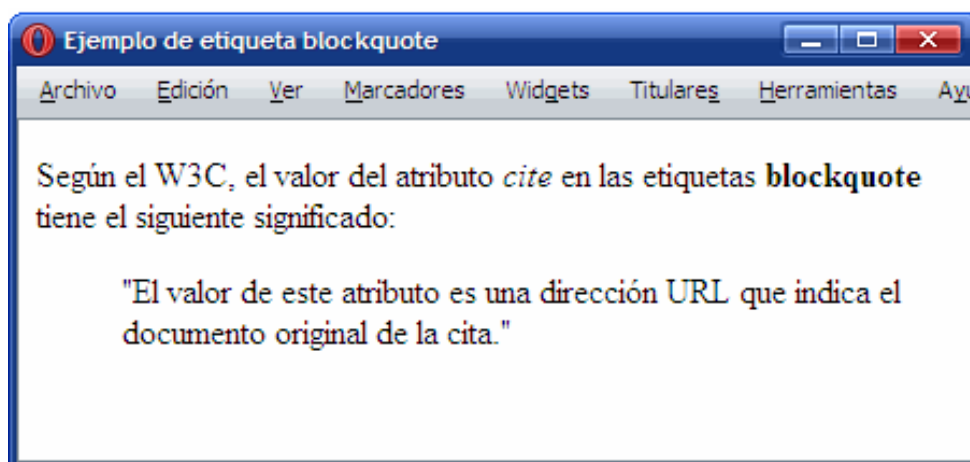


Figura 2.19. Ejemplo de uso de la etiqueta blockquote

Para indicar de forma clara que el texto es una cita externa, los navegadores muestran por defecto el texto del elemento `<blockquote>` con un gran margen en la parte izquierda.

3.3. Marcado avanzado de texto

Las páginas y documentos más avanzados suelen incluir otros elementos importantes que se deben marcar de forma adecuada. Por ello, HTML incluye muchas otras etiquetas que permiten marcar más elementos del texto.

La etiqueta `<abbr>` marca las abreviaturas de un texto y la etiqueta `<acronym>` se emplea para marcar las siglas o acrónimos del texto. Su definición es la siguiente:

<code><abbr></code>	Abreviaturas
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<code>title = "texto"</code> - Indica el significado completo de la abreviatura
Tipo de elemento	En línea
Descripción	Se emplea para marcar las abreviaturas del texto y proporcionar el significado de esas abreviaturas

<acronym>	No soportada en HTML5. Acrónimos o siglas
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<code>title = "texto"</code> - Indica el significado completo del acrónimo o sigla
Tipo de elemento	En línea
Descripción	Se emplea para marcar las siglas o acrónimos del texto y proporcionar el significado de esas siglas

En ambos casos, el atributo `title` se puede utilizar para incluir el significado completo de la abreviatura o sigla. Ejemplo:

```
<!DOCTYPE html>

<html>

<head>
<title>Ejemplo de etiqueta acronym</title>
</head>

<body>
<p>El lenguaje <acronym title="HyperText Markup
Language">HTML</acronym> es estandarizado
por el <acronym title="World Wide Web
Consortium">W3C</acronym>.</p>
</body>

</html>
```

La mayoría de navegadores muestran por defecto un borde inferior punteado para todos los elementos `<abbr>` y `<acronym>`. Al posicionar el puntero del ratón sobre la palabra subrayada, el navegador muestra un pequeño recuadro (llamado *tooltip* en inglés) con el valor del atributo `title`:

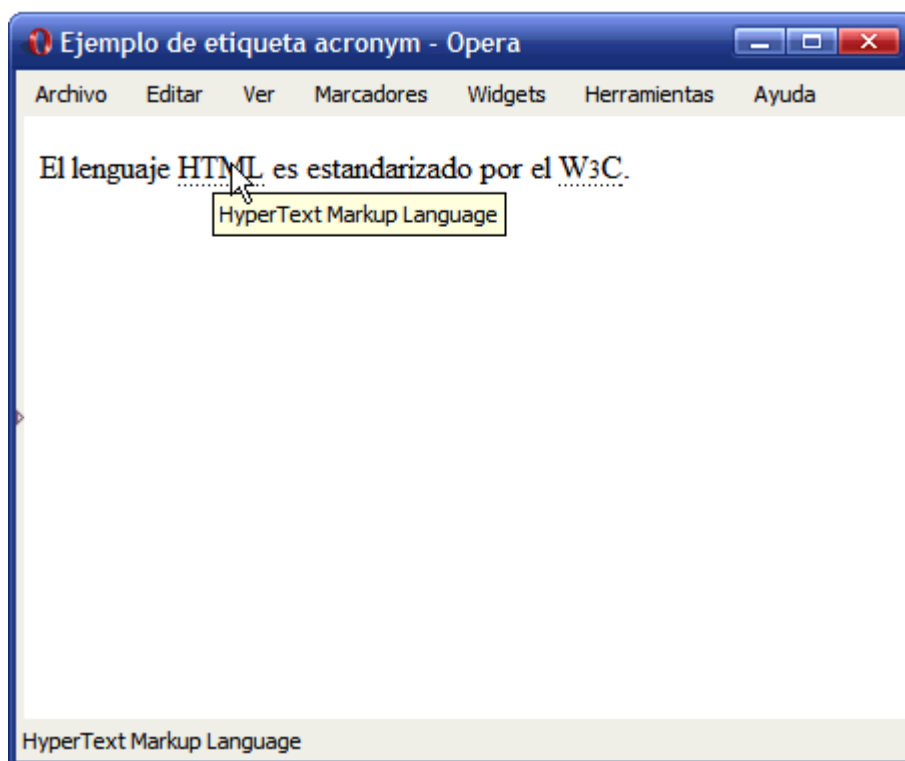


Figura 2.20. Ejemplo de uso de la etiqueta acronym

Por otra parte, en ocasiones resulta útil incluir la definición de una palabra extraña o cuyo uso está restringido a un entorno muy determinado. HTML incluye la etiqueta `<dfn>` para proporcionar al usuario la definición de todas las palabras para las que se considere apropiado. La definición formal de esta etiqueta se muestra a continuación:

<dfn>	Definición
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<code>title = "texto"</code> - Indica el significado completo del término
Tipo de elemento	En línea
Descripción	Se emplea para marcar las definiciones de ciertos términos y proporcionar el significado de esos términos

El siguiente ejemplo muestra cómo se utiliza la etiqueta `<dfn>` para incluir la definición completa de una palabra cuyo uso no es habitual fuera de los ámbitos médicos y psicológicos:

```
<p>Con estos síntomas, podría tratarse de un caso de <dfn  
title="Imagen o sensación subjetiva, propia de un sentido,  
determinada por otra sensación que afecta a un sentido  
diferente">sinestesia</dfn></p>
```

Por último, HTML incluye una etiqueta que se puede utilizar para marcar un texto como una citación:

<cite>	Cita
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Se emplea para marcar una cita o una referencia a otras fuentes

En ocasiones, no está clara la diferencia entre `<cite>` y `<blockquote>`. El elemento `<cite>` marca el autor de la cita (persona, documento, etc.) y `<blockquote>` marca el contenido de la propia cita. En el siguiente ejemplo, `<blockquote>` encierra el contenido de una frase célebre y `<cite>` encierra el nombre de su autor:

```
Como dijo <cite>Mahatma Gandhi</cite>:  
<blockquote>Vive como si fueras a morir mañana y aprende como si  
fueras a vivir para siempre.</blockquote>
```

Ejercicio 3: Estructurar y marcar el siguiente texto para que el navegador lo muestre con el aspecto de la siguiente imagen:

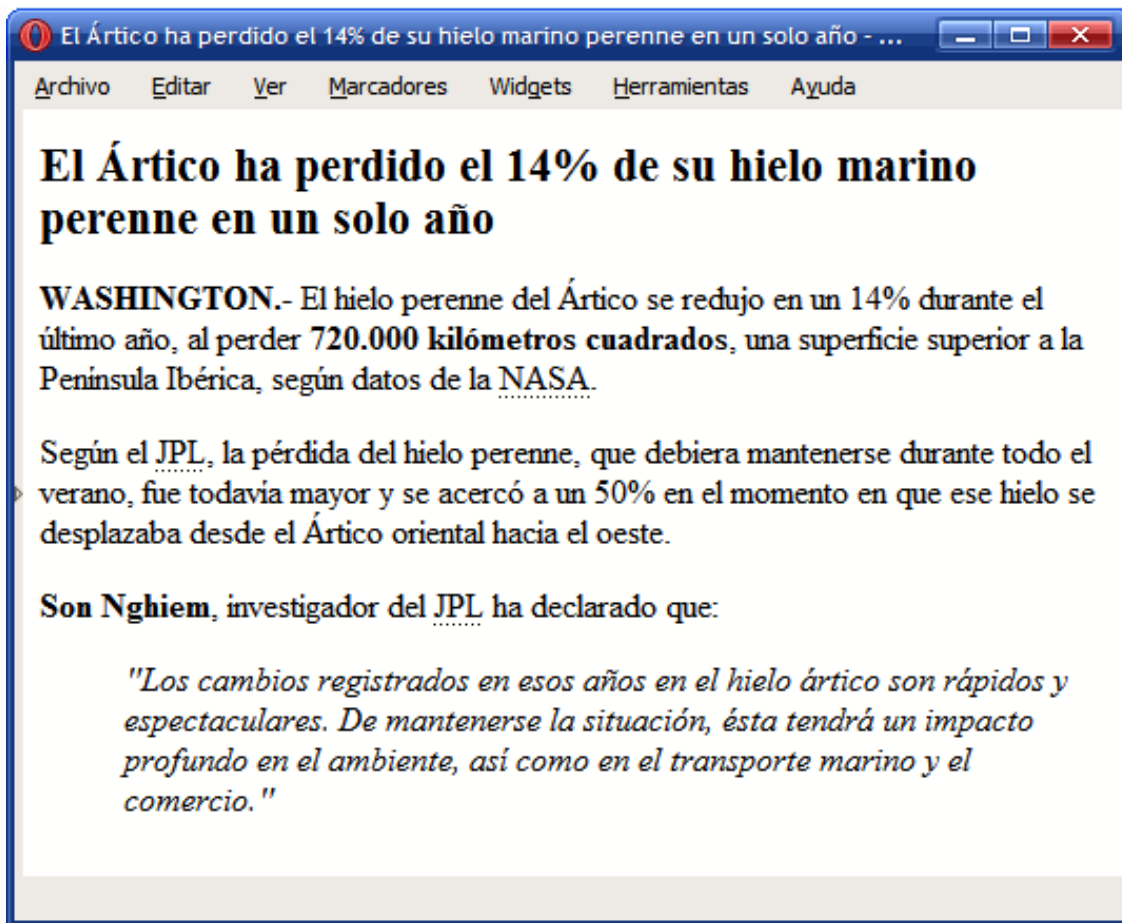


Figura 2.21. Texto HTML correctamente estructurado y marcado

3.4. Marcado genérico de texto

El estándar HTML/XHTML incluye numerosas etiquetas para marcar los contenidos de texto. No obstante, la infinita variedad de posibles contenidos textuales hace que no sean suficientes. Si se considera el siguiente ejemplo:

Importante: si quiere ponerse en contacto con la empresa ACME, puede hacerlo en el teléfono 900 555 555 o a través de la dirección de correo electrónico contacto@acme.org

El texto del ejemplo anterior contiene elementos de texto importantes, siglas, números de teléfono y direcciones de correo electrónico. XHTML define la etiqueta `` para marcar los elementos importantes y `<acronym>` para marcar las siglas:

```
<strong>Importante</strong>: si quiere ponerse en contacto con la  
empresa <acronym>ACME</acronym>, puede hacerlo en el teléfono 900 555  
555 o a través de la dirección de correo electrónico contacto@acme.org
```

Desafortunadamente, XHTML no define ninguna etiqueta específica para marcar números de teléfono o direcciones de correo electrónico. De la misma forma, no define etiquetas para otros posibles elementos que se pueden encontrar en los contenidos de texto.

Por este motivo, el estándar HTML/XHTML incluye una etiqueta llamada `` que se emplea para marcar cualquier elemento que no se puede marcar con las otras etiquetas definidas. Siguiendo con el ejemplo anterior, la etiqueta `` se utiliza para marcar el teléfono y la dirección de correo electrónico:

```
<strong>Importante</strong>: si quiere ponerse en contacto con la  
empresa <acronym>ACME</acronym>, puede hacerlo en el teléfono  
<span>900 555 555</span> o a través de la dirección de correo  
electrónico <span>contacto@acme.org</span>
```

La etiqueta `` se visualiza por defecto con el mismo aspecto que el texto normal. Por tanto es habitual utilizar esta etiqueta junto con los atributos `id` y `class` para modificar posteriormente su aspecto con CSS:

```
<strong>Importante</strong>: si quiere ponerse en contacto con la  
empresa <acronym>ACME</acronym>, puede hacerlo en el teléfono <span  
class="telefono">900 555 555</span> o a través de la dirección de  
correo electrónico <span class="email">contacto@acme.org</span>
```

La etiqueta `` sólo se puede utilizar para encerrar contenidos y etiquetas en línea. Cuando se quieren estructurar elementos de bloque, se utiliza la etiqueta `<div>`, tal y como se verá en apartados posteriores.

En este punto vamos a ver las nuevas etiquetas de marcado que han aparecido con HTML5.

Esta etiqueta, `<wbr>` de “word break” o más bien “word wrap”, hace es un “Ajuste de línea condicional”.

Ejemplo:

Dependiendo de la resolución de tu monitor verás el siguiente texto como una sola línea o dos líneas, estrecha el ancho de la ventana de tu navegador y verás cómo se va rompiendo la línea en los sitios que hemos insertado la etiqueta `<wbr>`:

```
<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de etiqueta wbr</title>

</head>

<body>

<p>Pluriblanduzcosplusgrandifecho<wbr>entrifugadlPluriblanduzcos
plusgrandifechocentrifugadlo<wbr>smurciélagoPluriblanduzcosplusg
randifecho<wbr>centrifugadlosmurciélagoPluriblanduzcosplusgrandi
fecho<wbr>centrifugadlosmurciélagoélago
</p>

</body>

</html>
```

3.5.2. Etiqueta <time>

La etiqueta `<time>` define bien una hora (24 horas de reloj) o una fecha en el calendario, opcionalmente con una hora y un desplazamiento de zona horaria.

Este elemento se puede utilizar como una forma de codificar las fechas y horas de una forma legible para el ordenador, de modo que, por ejemplo, se puede utilizar para añadir recordatorios de cumpleaños o eventos programados para el calendario del usuario, y los motores de búsqueda pueden producir resultados más inteligentes.

<time>	tiempo
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<code>datetime="fecha"</code> - Indica la fecha/hora especificada, sino, la fecha/hora viene dada por el contenido del elemento. <code>pubdate="fecha"</code> - Indica la fecha/hora es la de publicación del documento o del elemento artículo más cercano
Tipo de elemento	En línea
Descripción	Se emplea para marcar una cita o una referencia a otras fuentes

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de etiqueta time</title>
</head>
<body>
<p>Hoy es <time>2012-10-10</time></p>

<p>La reunión empieza a las <time>2012-10-10</time></p>

<p>Abrimos a las <time datetime="11:00">11:00</time></p>

<p>Tengo una cita el <time datetime="2012-12-25">día de
Navidad</time>.</p>
</body>
</html>
```

3.5.3. Etiqueta <mark>

La etiqueta `<mark>` define texto marcado. Se utiliza para resaltar partes del texto. Soporta los atributos comunes: básicos, i18n y eventos.

Ejemplo

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo de etiqueta mark</title>
  </head>
  <body>
    <p>No olvides comprar <mark>leche</mark> hoy.</p>
  </body>
</html>
```

3.5.4. Etiqueta <meter>

La etiqueta `<meter>` define una medida escalar con un rango conocido o un valor fraccional. Por ejemplo, el uso del disco, la relevancia del resultado de una encuesta, etc. No se debe utilizar para indicar el progreso (en una barra de progreso). Para este fin hay que usar la etiqueta `<progress>`

Soporta los atributos comunes: básicos, i18n y eventos.

Atributos específicos:

Atributo	Valor	Descripción
form	<i>form_id</i>	Specifies one or more forms the <meter> element belongs to
high	<i>number</i>	Specifies the range that is considered to be a high value
low	<i>number</i>	Specifies the range that is considered to be a low value
max	<i>number</i>	Specifies the maximum value of the range
min	<i>number</i>	Specifies the minimum value of the range
optimum	number	Specifies what value is the optimal value for the gauge
value	<i>number</i>	Required. Specifies the current value of the gauge

Ejemplo:

```
<!DOCTYPE html>

<html>
  <head>
    <title>Ejemplo de etiqueta meter</title>
  </head>
  <body>
    <p>Han aprobado la asignatura <meter value="2" min="0"
max="10">2 de 10</meter> alumnos.</p>
    <p>El candidato a las elecciones X ha ganado por un <meter
value="0.6">60%</meter></p>

  </body>
</html>
```

3.6. Espacios en blanco y nuevas líneas

El aspecto más sorprendente del lenguaje HTML cuando se desarrollan los primeros documentos es el tratamiento especial de los “*espacios en blanco*” del texto. HTML considera *espacio en blanco* a los espacios en blanco, los tabuladores, los retornos de carro y el carácter de nueva línea ([ENTER](#) o [Intro](#)).

El siguiente ejemplo ilustra este comportamiento:

```
<!DOCTYPE html>

<html>
<head><title>Ejemplo de etiqueta p</title></head>
<body>
<p>Este primer párrafo no contiene saltos de línea ni otro tipo
de espaciado.</p>

<p>Este segundo párrafo sí que contiene saltos
de
línea
y otro tipo de espaciado.</p>
</body>
</html>
```

El anterior código HTML se visualiza en cualquier navegador de la siguiente manera:

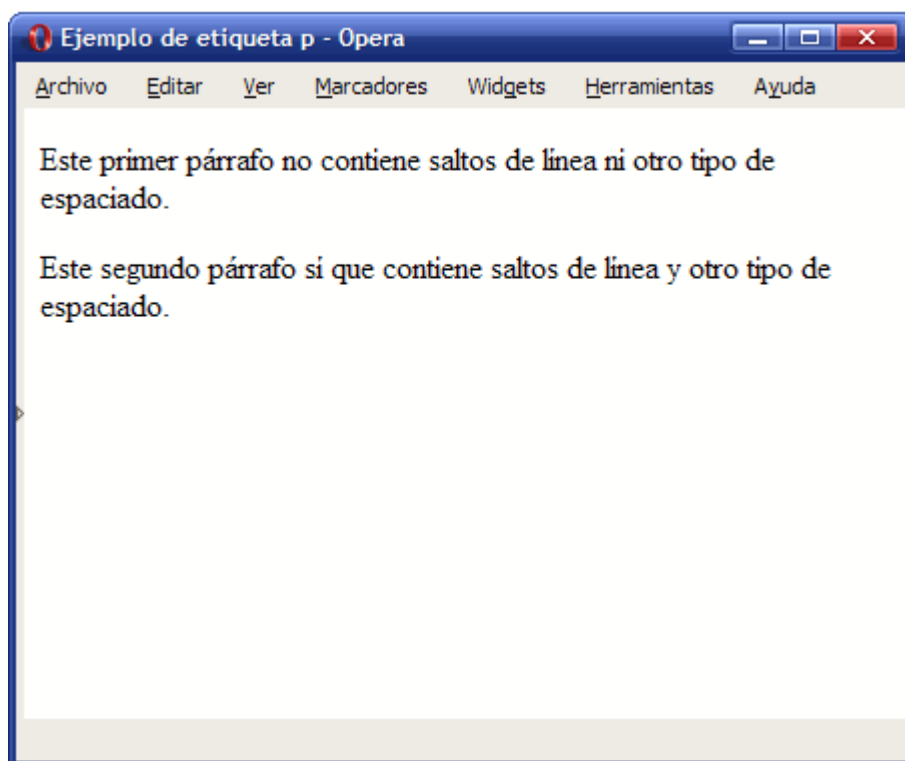


Figura 2.22. Ejemplo de comportamiento de HTML con los espacios en blanco

Los dos párrafos de la imagen anterior se ven idénticos, aunque el segundo párrafo incluye varios espacios en blanco y está escrito en varias líneas diferentes. La razón de este comportamiento es que HTML ignora todos los espacios en blanco *sobrantes*, es decir, todos los espacios en blanco que no son el espacio en blanco que separa las palabras.

No obstante, HTML proporciona varias alternativas para poder incluir tantos espacios en blanco y tantas nuevas líneas como sean necesarias dentro del contenido textual de las páginas.

3.6.1. Nuevas líneas

Para incluir una nueva línea en un punto y forzar a que el texto que sigue se muestre en la línea inferior, se utiliza la etiqueta `
`. En cierta manera, insertar la etiqueta `
` en un determinado punto del texto equivale a presionar la tecla **ENTER** (o **Intro**) en ese mismo punto.

La definición formal de `
` se muestra a continuación:

	Nueva línea
Atributos comunes	básicos
Atributos específicos	-
Tipo de elemento	En línea y etiqueta vacía
Descripción	Fuerza al navegador a insertar una nueva línea

La etiqueta `
` es una de las pocas *etiquetas especiales* de HTML. La particularidad de `
` es que es una etiqueta vacía, es decir, no encierra ningún texto. De esta forma, la etiqueta debe abrirse y cerrarse de forma consecutiva: `
</br>`.

En estos casos, HTML permite utilizar un atajo para indicar que una etiqueta se está abriendo y cerrando de forma consecutiva: `
` (también se puede escribir como `
`).

Utilizando la etiqueta `
` se puede rehacer el ejemplo anterior para que respete las líneas que forman el segundo párrafo:

```
<!DOCTYPE html>

<html>

<head>
<title>Ejemplo de etiqueta br</title>
</head>

<body>
<p>Este primer párrafo no contiene saltos de línea ni otro tipo
de espaciado.</p>

<p>Este segundo párrafo sí que contiene saltos <br/>
de <br/>
línea <br/>
y otro tipo de espaciado.</p>
</body>

</html>
```

El navegador ahora sí que muestra correctamente las nuevas líneas que se querían insertar:

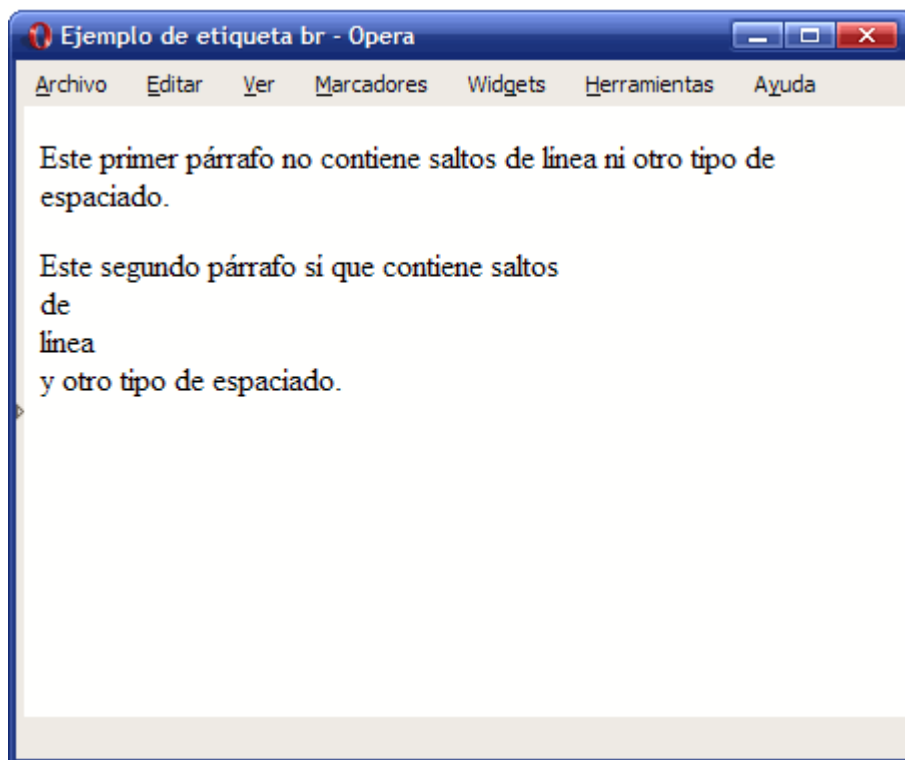


Figura 2.23. Ejemplo de uso de la etiqueta `br`

3.6.2. Espacios en blanco

La solución al problema de los espacios en blanco no es tan sencilla como el de las nuevas líneas. Para incluir espacios en blanco adicionales, se debe sustituir cada nuevo espacio en blanco por el texto ` ` (es importante incluir el símbolo `&` al principio y el símbolo `;` al final).

Así, el código HTML del ejemplo anterior se debe rehacer para incluir los espacios en blanco adicionales:

```
<!DOCTYPE html>

<html>

<head>
<title>Ejemplo de entidad &nbsp;</title>
</head>

<body>
<p>Este primer párrafo no contiene saltos de línea ni otro tipo
de espaciado.</p>
```

```
<p>Este segundo párrafo sí que contiene saltos <br/>  
de <br/>  
línea <br/>  
y &nbsp;&nbsp;&nbsp;&nbsp;&nbsp; otro &nbsp;&nbsp;&nbsp;&nbsp; tipo &nbsp;&nbsp;&nbsp;&nbsp; de &nbsp;&nbsp;&nbsp;&nbsp; espaciado.</p>  
</body>  
  
</html>
```

Ahora el navegador sí que muestra correctamente los espacios en blanco (y las nuevas líneas) del segundo párrafo:

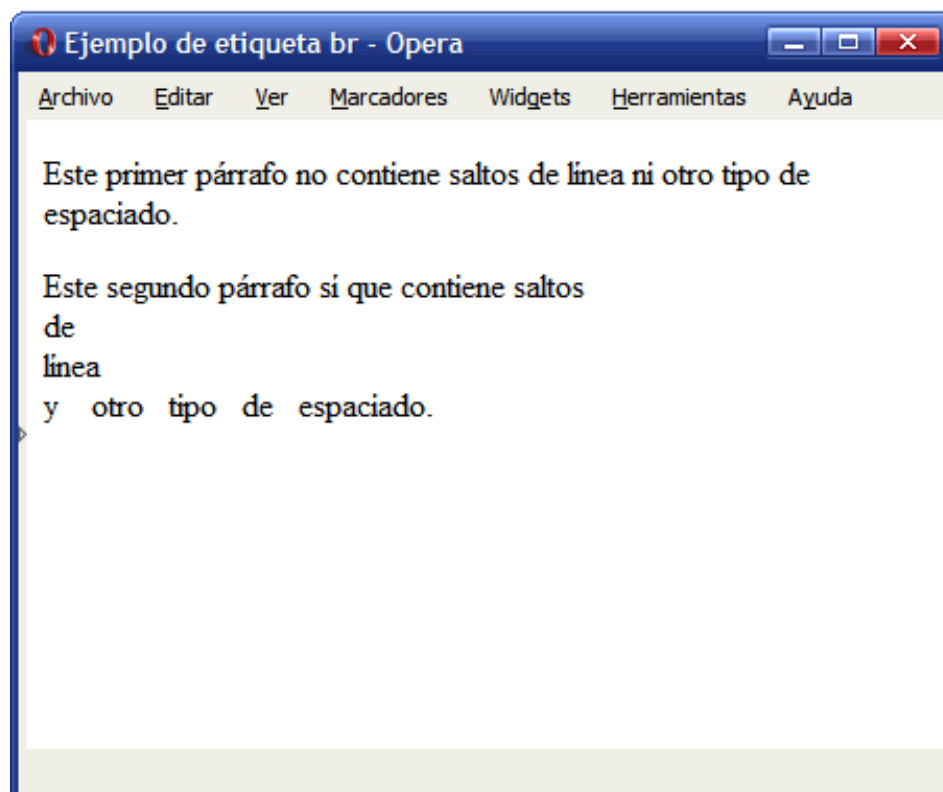


Figura 2.24. Ejemplo de uso de espacios en blanco en HTML

Cada texto ` ` solamente equivale a un espacio en blanco, por lo que se deben escribir tantos ` ` seguidos como espacios en blanco seguidos existan en el texto.

Más adelante se profundiza en el origen de ` ` y se comprenderá por qué es necesario incluir esa sucesión tan extraña de caracteres cada vez que se quiere incluir un espacio en blanco adicional.

Ejercicio 4: Determinar el código HTML que corresponde al siguiente documento:



Figura 2.25. Texto HTML con espacios en blanco y nuevas líneas

3.6.3. Texto preformateado

En ocasiones, es necesario mostrar los espacios en blanco de un texto que no se puede modificar. Se trata de un caso habitual cuando una página web debe mostrar directamente el texto generado por alguna aplicación.

En estos casos, se puede utilizar la etiqueta `<pre>`, que muestra el texto tal y como se ha escrito, respetando todos los espacios en blanco y todas las nuevas líneas. La definición formal de la etiqueta se muestra a continuación:

<pre>	Texto preformateado
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Muestra el texto que encierra tal y como está escrito (respetando los espacios en blanco)

El siguiente ejemplo muestra el uso de la etiqueta `<pre>`:

```
<!DOCTYPE html>
<html>
<head><title>Ejemplo de etiqueta pre</title></head>

<body>
<pre>
    La etiqueta pre
    respeta los espacios en blanco
    y
    muestra el texto
    tal y como
    está escrito
</pre>

<p>
    La etiqueta pre
    respeta los espacios en blanco
    y
    muestra el texto
    tal y como
    está escrito
</p>
</body>
</html>
```

El ejemplo anterior incluye el mismo texto (con espacios en blanco y varias líneas) dentro de una etiqueta `<pre>` y dentro de una etiqueta `<p>`. Las diferencias visuales en un navegador son muy evidentes:

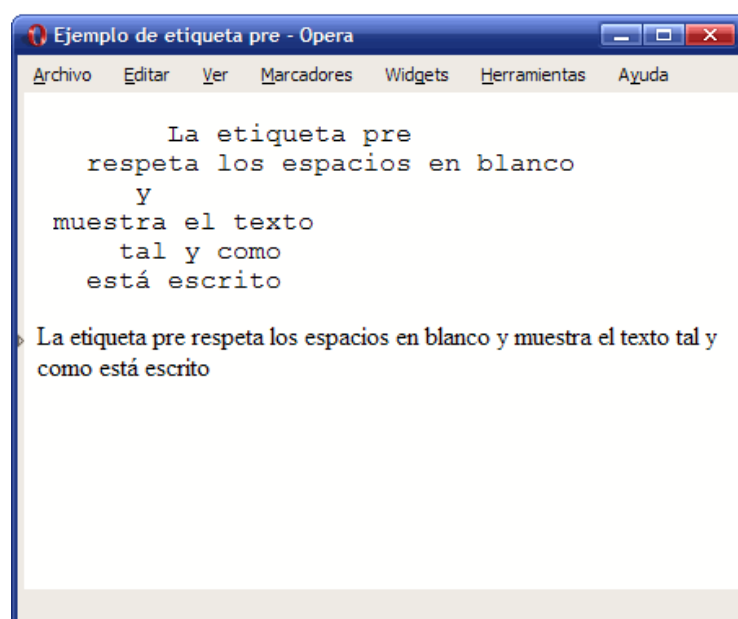


Figura 2.26. Ejemplo de uso de la etiqueta `pre`

El primer texto se ve en pantalla tal y como se ha escrito, respetando todos los espacios en blanco y todas las nuevas líneas. El segundo texto se ve como un párrafo normal, ya que HTML ha eliminado todos los espacios en blanco sobrantes. Los elementos `<pre>` son *especiales*, ya que los navegadores les aplican las siguientes reglas:

- Mantienen todos los espacios en blanco (tabuladores, espacios y nuevas líneas)
- Muestra el texto con un tipo de letra especial, denominado "*de ancho fijo*", ya que todas sus letras son de la misma anchura
- No se ajusta la longitud de las líneas (las líneas largas producen un *scroll* horizontal en la ventana del navegador)

Esta última característica diferencia por completo a los párrafos de los elementos `<pre>`. Como se ha visto, los navegadores ajustan la anchura de los párrafos de texto para que ocupen todo el tamaño de la ventana. Sin embargo, los elementos `<pre>` se muestran tal y como son originalmente, por lo que una línea muy larga dentro de un elemento `<pre>` provoca que la anchura de la página sea superior a la anchura de la ventana del navegador.

Si en el ejemplo anterior se añade más texto al final de la segunda línea (para producir una línea larga), el navegador muestra un *scroll* horizontal ya que el texto completo no cabe en el tamaño de la ventana y las líneas de los elementos `<pre>` nunca se ajustan.

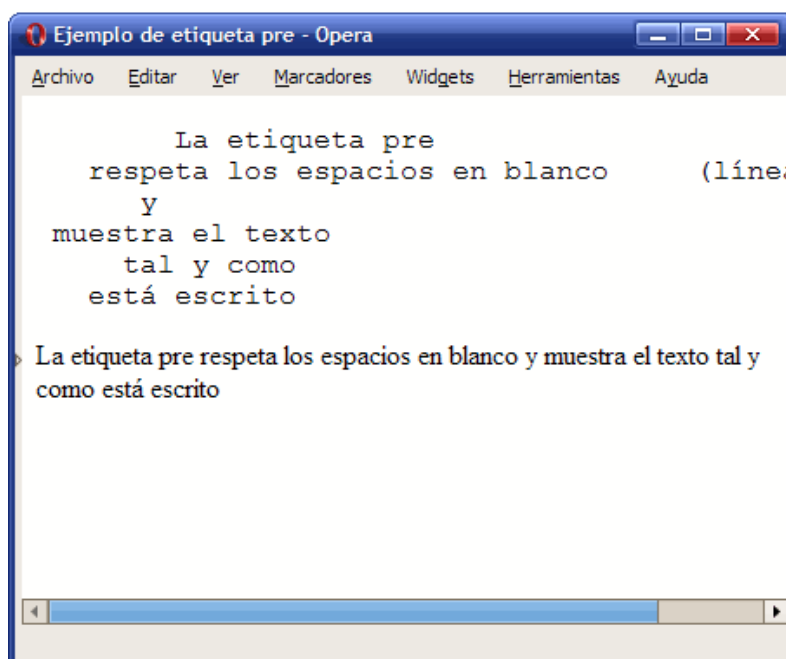


Figura 2.27. Ejemplo de aparición de scroll horizontal con la etiqueta `pre`

Otra etiqueta relacionada con `<pre>` es la etiqueta `<code>`, que se utiliza para mostrar código fuente de cualquier lenguaje de programación. La definición formal de `<code>` es la siguiente:

<code><code></code>	Código fuente
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Delimita el texto considerado un fragmento de código fuente

En la mayoría de páginas web, no tiene sentido utilizar la etiqueta `<code>`. Sin embargo, en muchas páginas web técnicas que incluyen listados de programas, trozos de código o etiquetas HTML, lo correcto es emplear la etiqueta `<code>`.

Ejemplo:

```
<!DOCTYPE html>
<html>
<head><title>Ejemplo de etiqueta code</title></head>

<body>
<code>
    La etiqueta code
    no respeta los espacios en blanco
</code>

<p>La etiqueta <code>code</code> es similar a la
etiqueta <code>pre</code>, sobre todo en el formato
del texto.</p>

</body>
</html>
```

El navegador muestra claramente el comportamiento de `<code>` y sus diferencias con `<pre>`:

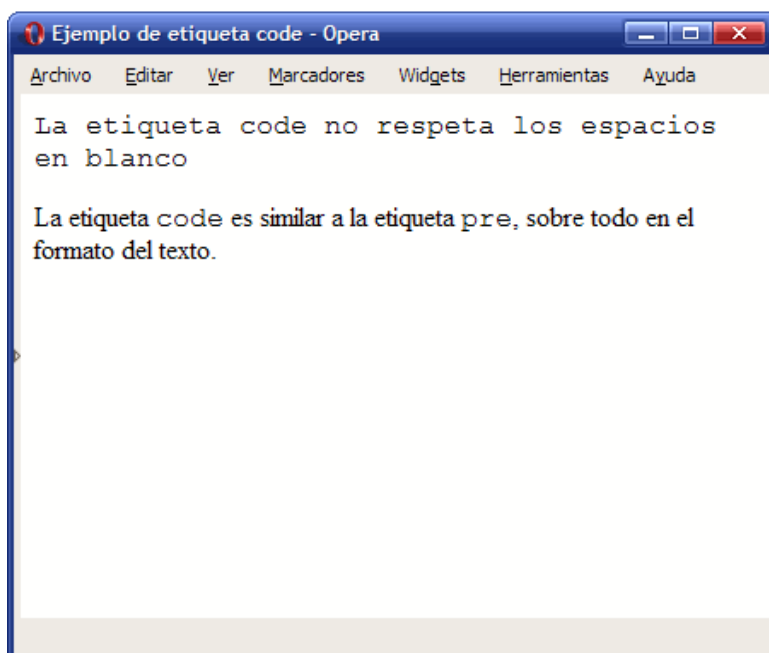


Figura 2.28. Ejemplo de uso de la etiqueta code

Al igual que sucede con los elementos `<pre>`, el texto encerrado por la etiqueta `<code>` se muestra con un tipo de letra especial de ancho fijo. Por el contrario, el elemento `<code>` no respeta los espacios en blanco ni las líneas, por lo que su comportamiento es similar a la etiqueta `<p>`. La última diferencia es que `<code>` es un elemento en línea, mientras que `<pre>` es un elemento de bloque.

3.7. Codificación de caracteres

Una consideración importante directamente relacionada con el texto de las páginas HTML es la codificación de los caracteres y la inserción de caracteres *especiales*. Algunos de los caracteres que se utilizan habitualmente en los textos no se pueden incluir directamente en las páginas web:

- Los caracteres que utiliza HTML para definir sus etiquetas (`<`, `>` y `"`) no se pueden utilizar libremente.
- Los caracteres propios de los idiomas que no son el inglés (ñ, á, ç, ¿, ¡, etc.) pueden ser problemáticos dependiendo de la codificación de caracteres utilizada.

La solución a la primera limitación consiste en sustituir los caracteres reservados de HTML por unas expresiones llamadas **entidades HTML** y que representan a cada carácter:

Entidad	Carácter	Descripción	Traducción
<	<	less than	signo de menor que
>	>	more than	signo de mayor que
&	&	ampersand	ampersand
"	"	quotation mark	comillas
 	(espacio en blanco)	non-breaking space	espacio en blanco
'	'	apostrophe	apóstrofo

De esta forma, si se considera el siguiente texto:

Los caracteres <, >, " y & pueden dar problemas con los textos en HTML

Para mostrar correctamente el texto anterior en una página HTML, se debe sustituir cada carácter especial por su entidad HTML:

<p>Los caracteres <, >, " y & pueden dar problemas con los textos en HTML</p>

Ejercicio 5: Determinar el código HTML que corresponde al siguiente documento:

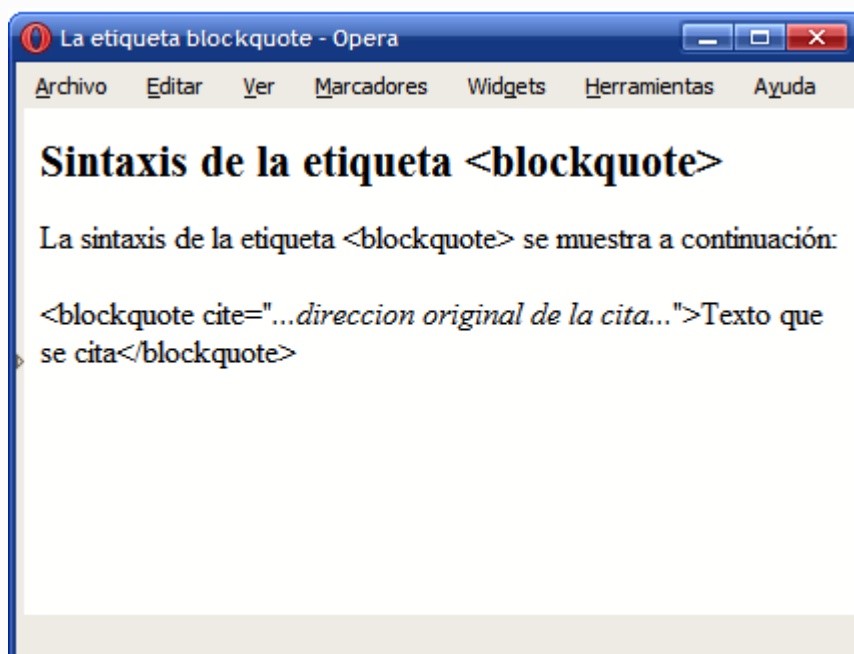


Figura 2.29. Texto HTML que incluye caracteres especiales

Por otra parte, los caracteres propios de los idiomas diferentes al inglés también pueden ser problemáticos. El motivo es que desde que se crea una página web hasta que llega al navegador del usuario, intervienen numerosos procesos:

El diseñador crea la página web con su editor HTML (por ejemplo Dreamweaver).

Si se trata de una aplicación dinámica, el programador recorta la página HTML del diseñador y la mezcla con el resto del código de la aplicación (por ejemplo PHP).

El servidor web almacena las páginas HTML estáticas o el código de la aplicación web y sirve las páginas solicitadas por los usuarios.

El usuario solicita y visualiza las páginas web a través de su navegador.

Si en todos los procesos anteriores se utiliza la misma codificación de caracteres, los caracteres propios de los idiomas se pueden escribir directamente:

```
<p>Este párrafo contiene caracteres acentuados y se almacena en  
formato UTF-8</p>
```

La palabra **párrafo** del ejemplo anterior incluye la letra á. Si el editor HTML del diseñador utiliza la codificación UTF-8, el entorno de desarrollo del programador también utiliza UTF-8, el servidor web sirve las páginas con esa codificación y el navegador del usuario es capaz de visualizar las páginas con formato UTF-8, el texto anterior se verá correctamente en el navegador del usuario.

Sin embargo, muchas veces no es posible que todos los procesos involucrados utilicen la misma codificación de caracteres. Por limitaciones técnicas o por decisiones de los diseñadores y programadores, los textos pueden pasar de codificación UTF-8 a codificación ISO-8859 en cualquier momento. Si se produce este cambio sin realizar una conversión correcta, el navegador del usuario mostrará caracteres extraños en todos los acentos y en todas las letras como la ñ.

La solución más sencilla para asegurar que todos estos caracteres potencialmente problemáticos se van a visualizar correctamente en el navegador del usuario consiste en sustituir cada carácter problemático por su entidad HTML:

Entidad	Carácter	Descripción oficial
ñ	ñ	latin letter n with tilde
Ñ	Ñ	latin capital n letter with tilde
á	á	a acute
é	é	e acute
í	í	i acute
ó	ó	o acute
ú	ú	u acute
Á	Á	A acute
É	É	E acute
Í	Í	I acute
Ó	Ó	O acute
Ú	Ú	U acute
€	€	euro

Así, el párrafo de texto del ejemplo anterior, se podría escribir de la siguiente manera:

```
<p>Este párrafo contiene caracteres acentuados y se almacena en  
formato UTF-8</p>
```

Si se utilizan las entidades HTML en vez de los caracteres problemáticos, es indiferente pasar de una codificación de caracteres a otra diferente. En la Wikipedia se puede consultar la lista completa de las 252 entidades HTML definidas (http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references).

4. ENLACES

El lenguaje de marcado HTML se definió teniendo en cuenta algunas de las características que existían en ese momento para la publicación digital de contenidos. Entre los conceptos utilizados en su creación, se encuentra el mecanismo de "*hipertexto*".

De hecho, las letras "HT" de la sigla HTML significan "*hipertexto*" (*hypertext* en inglés), por lo que el significado completo de HTML podría traducirse como "lenguaje de marcado para hipertexto".

La incorporación del *hipertexto* fue una de las claves del éxito del lenguaje HTML, ya que permitió crear documentos interactivos que proporcionan información adicional cuando se solicita. El elemento principal del hipertexto es el "*hiperenlace*", también llamado "enlace web" o simplemente "enlace".

Los enlaces se utilizan para establecer relaciones entre dos recursos. Aunque la mayoría de enlaces relacionan páginas web, también es posible enlazar otros recursos como imágenes, documentos y archivos.

Una característica que no se suele tener en cuenta en los enlaces es que están formados por dos extremos y un sentido. En otras palabras, el enlace comienza en un recurso y apunta hacia otro recurso. Cada uno de los dos extremos se llaman "*anchors*" en inglés, que se puede traducir literalmente como "anclas".

4.1. URL

Antes de empezar a crear enlaces, es necesario comprender y dominar el concepto de URL. El acrónimo **URL** (del inglés *Uniform Resource Locator*) hace referencia al identificador único de cada recurso disponible en Internet. Las URL son esenciales para crear los enlaces, pero también se utilizan en otros elementos HTML como las imágenes y los formularios.

La URL de un recurso tiene dos objetivos principales:

- Identificar de forma única a ese recurso
- Permitir localizar de forma eficiente ese recurso

En primer lugar, las URL permiten que cada página HTML publicada en Internet tenga un nombre único que permita diferenciarla de las demás. De esta forma es posible crear enlaces que apunten de forma inequívoca a una determinada página.

Si se accede a la página principal de Google, la dirección que muestra el navegador es:

`http://www.google.com`

La cadena de texto <http://www.google.com> es la URL completa de la página principal de Google. La URL de las páginas es imprescindible para crear los enlaces, ya que permite distinguir una página de otra.

El segundo objetivo de las URL es el de permitir la localización eficiente de cada recurso de Internet. Para ello es necesario comprender las diferentes partes que forman las URL. Una URL sencilla siempre está formada por las mismas tres partes. Si por ejemplo se considera la siguiente URL:

`http://www.daw.es/diw/tema2.html`

Las partes que componen la URL anterior son:

- Protocolo (<http://>): el mecanismo que debe utilizar el navegador para acceder a ese recurso. Todas las páginas web utilizan <http://>. Las páginas web seguras (por ejemplo las de los bancos y las de los servicios de email) utilizan <https://> (se añade una letra *s*).
- Servidor (www.daw.es): simplificando mucho su explicación, se trata del ordenador en el que se encuentra guardada la página que se quiere acceder. Los navegadores son capaces de obtener la dirección de cada servidor a partir de su nombre.
- Ruta ([/diw/tema2.html](http://www.daw.es/diw/tema2.html)): *camino* que se debe seguir, una vez que se ha llegado al servidor, para localizar el recurso específico que se quiere acceder.

Por tanto, las URL no sólo identifican de forma única a cada recurso de Internet, sino que también proporcionan a los navegadores la información necesaria para poder llegar hasta ese recurso.

La mayoría de URL son tan sencillas como la URL mostrada anteriormente. No obstante, existen URL complejas formadas por más partes.

<http://www.alistapart.com/comments/webstandards2008?page=5#42>

Las cinco partes que forman la URL anterior son:

- Protocolo ([http://](#))
- Servidor ([www.alistapart.com](#))
- Ruta ([/comments/webstandards2008](#))
- Consulta ([?page=5](#)): información adicional necesaria para que el servidor localice correctamente el recurso que se quiere acceder. Siempre comienza con el carácter ? y contiene una sucesión de palabras separadas por = y &
- Sección ([#42](#)): permite que el navegador se posicione automáticamente en una sección de la página web. Siempre comienza con el carácter #

Como las URL utilizan los caracteres :, =, & y / para separar sus partes, estos caracteres están reservados y no se pueden utilizar libremente. Además, algunos caracteres no están reservados pero pueden ser problemáticos si se utilizan en la propia URL.

Si es necesario incluir estos caracteres reservados y especiales en una URL, se sustituyen por combinaciones de caracteres seguros. Esta sustitución se denomina **codificación de caracteres** y el servidor realiza el proceso inverso (**decodificación**) cuando le llega una URL con los caracteres codificados.

A continuación se muestra la tabla para codificar los caracteres más comunes:

Carácter original	Carácter codificado	Carácter original	Carácter codificado
/	%2F	?	%3F
:	%3A	@	%40
=	%3D	&	%26
"	%22	\	%5C
'	%27	~	%7E
(espacio en blanco)	%20		%7C

Por otra parte, aunque desde hace tiempo ya es posible incluir en las URL caracteres de otros idiomas que no sean el inglés, aún no es completamente seguro utilizar estos caracteres en las URL. Si se utilizan letras como ñ, á, é o ç, es posible que algunos navegadores no las interpreten de forma correcta.

La solución consiste en codificar todos los caracteres que no existen en inglés. La siguiente tabla muestra la codificación de los caracteres más utilizados:

Carácter original	Carácter codificado	Carácter original	Carácter codificado
ñ	%F1	Ñ	%D1
á	%E1	Á	%C1
é	%E9	É	%C9
í	%ED	Í	%CD
ó	%F3	Ó	%D3
ú	%FA	Ú	%DA
ç	%E7	Ç	%C7

Teniendo en cuenta las dos tablas anteriores de codificación de caracteres, es fácil crear las URL correctas sin caracteres problemáticos:

```
<!-- URL problemática -->  
http://www.ejemplo.com/estaciones/otoño.html
```

```
<!-- URL correcta -->  
http://www.ejemplo.com/estaciones/oto%F1o.html
```

```
<!-- URL problemática -->  
http://www.ejemplo.com/ruta/nombre página.html
```

```
<!-- URL correcta -->  
http://www.ejemplo.com/ruta/nombre%20p%E1gina.html
```

4.2. Enlaces relativos y absolutos

Las páginas web habituales suelen contener decenas de enlaces de diferentes tipos. La siguiente imagen muestra algunos de los tipos de enlaces de la página principal del sitio web <http://www.456bereastreet.com/>:

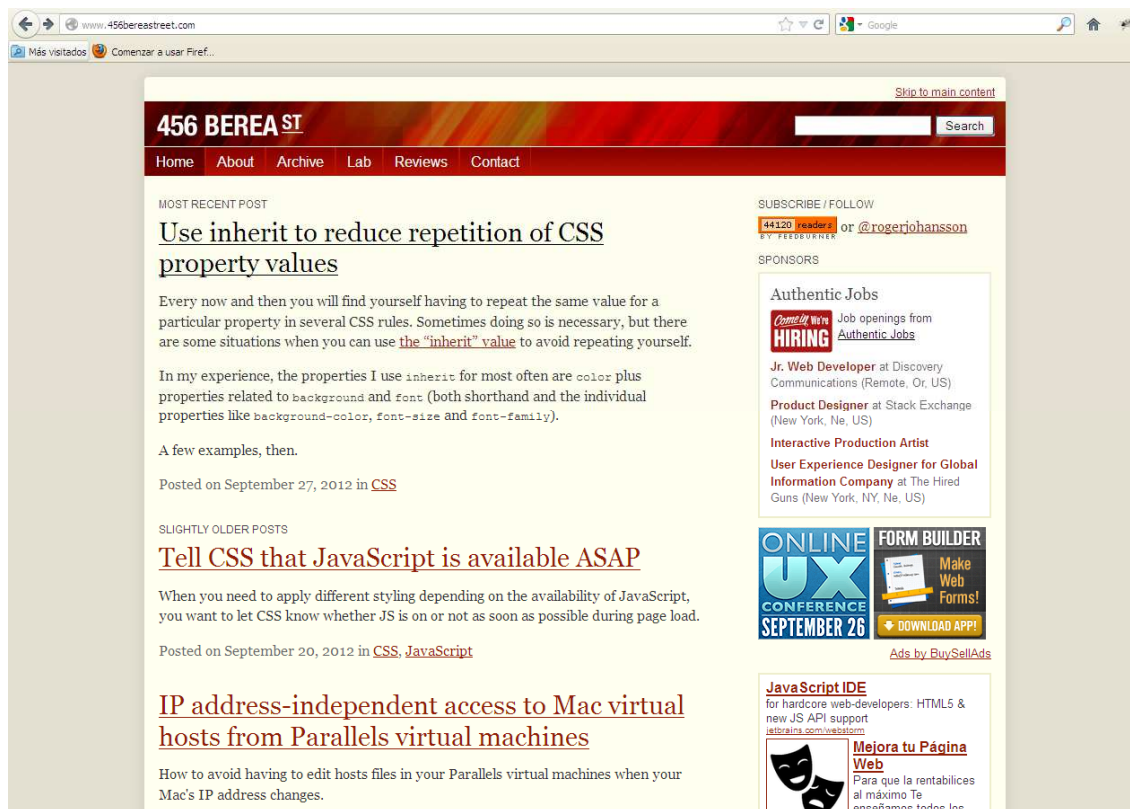


Figura 2.30. Ejemplo de diferentes tipos de enlaces en la página 456BereaStreet.com

En esa página, cuando se pincha sobre algunos enlaces, el navegador abandona el sitio web para acceder a páginas que se encuentran en otros sitios. Estos enlaces se conocen como "enlaces externos". Sin embargo, la mayoría de enlaces de un sitio web apuntan a páginas del propio sitio web, por lo que se denominan "enlaces internos".

Además de internos/externos, la otra característica que diferencia a los enlaces (y por tanto, también a las URL) es si el enlace es absoluto o relativo. Las **URL absolutas** incluyen todas las partes de la URL (protocolo, servidor y ruta) por lo que no se necesita más información para obtener el recurso enlazado.

Las **URL relativas** prescinden de algunas partes de las URL para hacerlas más breves. Como se trata de URL incompletas, es necesario disponer de información adicional para obtener el recurso enlazado. En concreto, para que una URL relativa sea útil es imprescindible conocer la URL del origen del enlace.

Las URL relativas se construyen a partir de las URL absolutas y prescinden de la parte del protocolo, del nombre del servidor e incluso de parte o toda la ruta del recurso enlazado. Aunque las URL relativas pueden ser difíciles de entender para los que comienzan con HTML, son tan útiles que todos los sitios web las utilizan.

Imagina que dispones de una página publicada en <http://www.ejemplo.com/ruta1/ruta2/pagina1.html> y quieres incluir en ella un enlace a otra página que se encuentra en <http://www.ejemplo.com/ruta1/ruta2/pagina2.html>. Como las URL identifican de forma única a los recursos de Internet y proporcionan la información necesaria para llegar hasta ellos, el enlace debería utilizar la URL completa de la segunda página.

Las URL completas también se llaman URL absolutas, ya que el navegador no necesita disponer de información adicional para localizar el recurso enlazado. Si se utilizan siempre las URL absolutas, los enlaces están completamente definidos.

Sin embargo, escribir siempre las URL completas es bastante aburrido, cuesta mucho tiempo y hace imposible cambiar la ubicación de los contenidos de un sitio web. Por ese motivo, casi todos los sitios web de Internet utilizan URL relativas siempre que es posible.

Una URL relativa es una versión abreviada de una URL absoluta. Su objetivo es eliminar todas las partes de la URL absoluta que se pueden *adivinar* a partir de la información de contexto de la página web. En otras palabras, las URL relativas aprovechan la *inteligencia* de los navegadores para crear URL incompletas que los navegadores pueden completar deduciendo la información que falta.

Considerando de nuevo el ejemplo anterior, la URL a la que se quiere enlazar utiliza el mismo protocolo y se encuentra en el mismo servidor que la página origen, por lo que la URL relativa puede prescindir de esas partes:

URL absoluta: http://www.ejemplo.com/ruta1/ruta2/pagina2.html URL relativa: /ruta1/ruta2/pagina2.html

En el ejemplo anterior, las dos URL son equivalentes porque cuando no se indica el protocolo y el servidor de una URL, los navegadores suponen que son los mismos que los de la página origen. Por lo tanto, cuando el navegador encuentra la URL </ruta1/ruta2/pagina2.html>, realiza el siguiente proceso:

1. La URL no es absoluta, por lo que se debe determinar la URL absoluta a partir de la URL relativa para poder cargar el recurso enlazado.
2. A la URL relativa le falta el protocolo y el servidor, por lo que se supone que son los mismos que los de la página origen (<http://> y www.ejemplo.com).
3. Se añaden las partes que faltan a la URL relativa para obtener la URL absoluta: <http://> + www.ejemplo.com + [/ruta1/ruta2/pagina2.html](http://www.ejemplo.com/ruta1/ruta2/pagina2.html) = <http://www.ejemplo.com/ruta1/ruta2/pagina2.html>.

Aunque el ejemplo mostrado es el caso más sencillo de URL relativa, existen otros casos más avanzados en los que se prescinde de parte o toda la ruta del recurso que se enlaza. A continuación se muestran los **cuatro tipos diferentes de URL relativas**:

1) El origen y el destino del enlace se encuentran en el mismo directorio

Si desde una página web se quiere enlazar un recurso que se encuentra en el mismo directorio del servidor, la URL relativa puede prescindir de todas las partes de la URL absoluta salvo el nombre del recurso enlazado.

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	Página web llamada pagina2.html y que se encuentra en el mismo directorio
URL absoluta	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina2.html
URL relativa	pagina2.html

Cuando el navegador encuentra una URL relativa que sólo consiste en el nombre de un recurso, supone que el protocolo, servidor y directorio del recurso enlazado son los mismos que los del origen del enlace.

2) El destino del enlace se encuentra cerca de su origen y en un nivel superior

En este caso, el recurso que se enlaza no está en el mismo directorio que el origen del enlace pero sí que está *cerca* y en algún directorio superior. La URL relativa debe indicar de alguna manera que es necesario *subir* un nivel en la jerarquía de directorios para llegar hasta el recurso.

Para indicar al navegador que debe subir un nivel, se incluyen dos puntos y una barra (../) en la ruta del recurso enlazado. De esta forma, cada vez que aparece ../ en una URL relativa, significa que se debe subir un nivel.

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	Página web llamada pagina2.html y que se encuentra en el directorio superior llamado ruta2
URL absoluta	http://www.ejemplo.com/ruta1/ruta2/pagina2.html
URL relativa	../pagina2.html

Cuando el navegador encuentra la URL relativa [../pagina2.html](#), sabe que para encontrar el recurso enlazado ([pagina2.html](#)) tiene que subir un nivel desde el lugar en el que se encuentra esa URL relativa. La página que incluye esa URL se encuentra en el directorio [ruta1/ruta2/ruta3](#), por lo que subir un nivel equivale entrar en el directorio [ruta1/ruta2](#).

De la misma forma, si el destino se encuentra un par de niveles por encima, se debe incluir ../ dos veces seguidas:

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	Página web llamada pagina2.html y que se encuentra en el directorio superior llamado ruta1
URL absoluta	http://www.ejemplo.com/ruta1/pagina2.html
URL relativa	../../pagina2.html

Además de subir niveles, también se puede entrar en otros directorios para obtener los recursos:

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	Página web llamada pagina2.html y que se encuentra en un directorio llamado ruta4 que se encuentra en la raíz del servidor
URL absoluta	http://www.ejemplo.com/ruta4/pagina2.html
URL relativa	../../../../ruta4/pagina2.html

Si se intentan subir más niveles de los que es posible, el navegador ignora todos los `../` sobrantes. Si la página que tiene el enlace es <http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html> y la URL relativa que se incluye es `../../../../../../../../pagina2.html`, el navegador realmente la interpreta como `../../../../pagina2.html`.

Como el objetivo de las URL relativas es crear URL más cortas y sencillas que las URL absolutas, este método sólo se puede utilizar cuando el origen y el destino se encuentran cerca, porque de otro modo la URL relativa se complica demasiado.

3) El destino del enlace se encuentra cerca de su origen y en un nivel inferior

Este caso es muy similar al anterior, pero más sencillo. Si el recurso enlazado se encuentra en algún directorio inferior al que se encuentra el origen, sólo es necesario indicar el nombre de los directorios a los que debe entrar el navegador.

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	Página web llamada pagina2.html y que se encuentra en un directorio inferior llamado ruta6 que está dentro del directorio ruta5 y que a su vez está dentro del directorio ruta4
URL absoluta	http://www.ejemplo.com/ruta1/ruta2/ruta3/ruta4/ruta5/ruta6/pagina2.html
URL relativa	ruta4/ruta5/ruta6/pagina2.html

De la misma forma, se pueden indicar varios directorios seguidos para que el navegador descienda jerárquicamente por la estructura de directorios:

4) El origen y el destino del enlace se encuentran muy alejados

Cuando el origen y el destino de un enlace se encuentran muy alejados (pero en el mismo servidor) las URL relativas se pueden complicar en exceso. Aunque es posible utilizar `../` para subir por la jerarquía de directorios y se puede entrar en cualquier directorio indicando su nombre, las URL relativas que se obtienen son demasiado largas y complicadas.

En estos casos, lo más sencillo es indicar la ruta completa hasta el recurso enlazado comenzando desde la raíz del servidor web. Por lo tanto, estas URL relativas sólo omiten el protocolo y el nombre del servidor.

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	Página web llamada <code>pagina2.html</code> y que se guarda en un directorio llamado <code>ruta7</code> que se encuentra en la raíz del servidor
URL absoluta	http://www.ejemplo.com/ruta7/pagina2.html
URL relativa	<code>/ruta7/pagina2.html</code>

Cuando la URL relativa comienza por `/`, el navegador considera que es la ruta completa comenzando desde la raíz del servidor, por lo que sólo le añade el protocolo y el nombre del servidor origen.

A continuación se resumen los cuatro posibles casos de URL relativas y el procedimiento que sigue el navegador para convertirlas en URL absolutas:

Si la URL relativa...	El navegador la transforma en URL absoluta...
...sólo consiste en el nombre de un recurso	...añadiendo el protocolo, servidor y ruta completa del origen del enlace
...comienza por <code>../</code>	...añadiendo el protocolo y servidor del origen del enlace, subiendo un nivel en la jerarquía de directorios y añadiendo el resto de la ruta incluida en la URL relativa
...comienza por <code>/</code>	...añadiendo el protocolo y servidor del origen del enlace
En cualquier otro caso	...añadiendo el protocolo, servidor y ruta completa del origen del enlace, a la que se añade la ruta incluida en la URL relativa

4.3. Enlaces básicos

Los enlaces en HTML se crean mediante la etiqueta `<a>` (su nombre viene del inglés "anchor", literalmente traducido como "ancla"). A continuación se muestra la definición simplificada de `<a>` y más adelante se muestra su definición completa. La etiqueta `<a>` ha sido cambiada en HTML5

<a>	Enlaces
Atributos comunes	básicos, i18n, eventos y foco
Atributos específicos	<code>name = "texto"</code> - Permite nombrar al enlace para que se pueda acceder desde otros enlaces. No está soportado en HTML5. <code>href = "url"</code> - Indica la URL del recurso que se quiere enlazar
Tipo de elemento	En línea
Descripción	Se emplea para enlazar todo tipo de recursos

El atributo más importante de la etiqueta `<a>` es `href`, que se utiliza para indicar la URL a la que apunta el enlace. Cuando el usuario pincha sobre un enlace, el navegador se dirige a la URL del recurso indicado mediante `href`. Las URL de los enlaces pueden ser absolutas, relativas, internas y externas.

Con la definición anterior, para crear un enlace que apunte a la página principal de Google solamente habría que incluir lo siguiente en un documento HTML:

```
<a href="http://www.google.com">Página principal de Google</a>
```

Como el atributo `href` indica una URL, un enlace puede apuntar a cualquier tipo de recurso al que pueda acceder el navegador. El siguiente enlace apunta a una imagen, que se mostrará en el navegador cuando el usuario pinche sobre el enlace:

```
<a href="http://www.ejemplo.com/fondo_escritorio.jpg">Imagen  
interesante para un fondo de escritorio</a>
```

De la misma forma, los enlaces pueden apuntar directamente a documentos PDF, Word, etc.

```
<a href="http://www.ejemplo.com/informe.pdf">Descargar informe  
completo [PDF]</a>
```

```
<a href="http://www.ejemplo.com/informe.doc">Descargar informe  
completo [DOC]</a>
```

Un truco muy útil con los enlaces es el uso de URL relativas para poder volver al inicio del sitio web desde cualquier página web interior:

```
<a href="/">Volver al inicio</a>
```

El enlace anterior utiliza una URL relativa con una ruta que apunta directamente a la raíz del servidor. Para obtener la URL absoluta, el navegador añade el mismo protocolo y el mismo nombre de servidor de la página en la que se encuentra el enlace. El resultado es que cuando se pincha ese enlace, siempre se vuelve al inicio del sitio web, independientemente de la página en la que se incluya el enlace.

El otro atributo básico de la etiqueta `<a>` es `name`, que permite definir enlaces dentro de una misma página web. Si una página es muy larga, puede ser útil mostrar enlaces de tipo "Saltar hasta la segunda sección", "Volver al principio de la página", etc.

Este tipo de enlaces son especiales, ya que la URL de la página siempre es la misma para todas las secciones y por tanto, debe añadirse otra parte a las URL para identificar cada sección.

```
<a name="primera_seccion"></a>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Mauris id ligula eu felis adipiscing ultrices. Duis gravida leo
ut lectus. Praesent condimentum mattis ligula.</p>

...

<a name="segunda_seccion"></a>

<p>Pellentesque malesuada. In in lacus. Phasellus erat erat,
lacinia a, convallis eu, nonummy et, odio. Aenean urna elit,
ultrices id, placerat varius, facilisis eget, dolor.</p>

...
```

El atributo `name` permite crear "enlaces vacíos" que hacen referencia a secciones dentro de una misma página. Una vez definidos los "enlaces vacíos", es posible crear un enlace que apunte directamente a una sección concreta de una página:

```
<!-- Enlace normal a La página -->  
<a href="http://www.ejemplo.com/pagina1.html">Enlace a la página 1</a>  
<!-- Enlace directo a La segunda sección de La página -->  
<a href="http://www.ejemplo.com/pagina1.html#segunda_seccion">Enlace a  
la sección 2 de la página 1</a>
```

La sintaxis que se utiliza con estos enlaces es la misma que con los enlaces normales, salvo que se añade el símbolo # seguido del nombre de la sección a la que se apunta. Cuando el usuario pincha sobre uno de estos enlaces, el navegador accede a la página apuntada por la URL y baja directamente a la sección cuyo nombre se indica después del símbolo #.

También es posible utilizar este tipo de enlaces con URL relativas en una misma página. El siguiente ejemplo añade enlaces de tipo "Volver al inicio de la página" en varias secciones:

```
<a name="inicio"></a>  
  
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Mauris id ligula eu felis adipiscing ultrices. Duis gravida leo  
ut lectus. Praesent condimentum mattis ligula.</p>  
  
<a href="#inicio">Volver al inicio de la página</a>  
  
...  
  
<p>Pellentesque malesuada. In in lacus. Phasellus erat erat,  
lacinia a, convallis eu, nonummy et, odio. Aenean urna elit,  
ultrices id, placerat varius, facilisis eget, dolor.</p>  
  
<a href="#inicio">Volver al inicio de la página</a>  
  
...
```

Los enlaces directos a secciones también funcionan con el atributo `id` de cualquier elemento. El siguiente ejemplo es equivalente al ejemplo anterior:


```
<h1 id="inicio">Título de la página</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Mauris id ligula eu felis adipiscing ultrices. Duis gravida leo
ut lectus. Praesent condimentum mattis ligula.</p>

<a href="#inicio">Volver al inicio de la página</a>

...

<p>Pellentesque malesuada. In in lacus. Phasellus erat erat,
lacinia a, convallis eu, nonummy et, odio. Aenean urna elit,
ultrices id, placerat varius, facilisis eget, dolor.</p>

<a href="#inicio">Volver al inicio de la página</a>

...
```

El nombre de la sección que se indica después del símbolo # puede utilizar el valor de los atributos `id` de cualquier elemento. De hecho, se recomienda utilizar los atributos `id` de los elementos ya existentes en la página en vez de crear "enlaces vacíos" de tipo ``.

Nota: El atributo `name` de la etiqueta `<a>` está obsoleto y se eliminará. Se debe usar el atributo `id` en su lugar.

Ejercicio 6: A partir de la estructura de directorios y archivos indicada en la siguiente imagen:

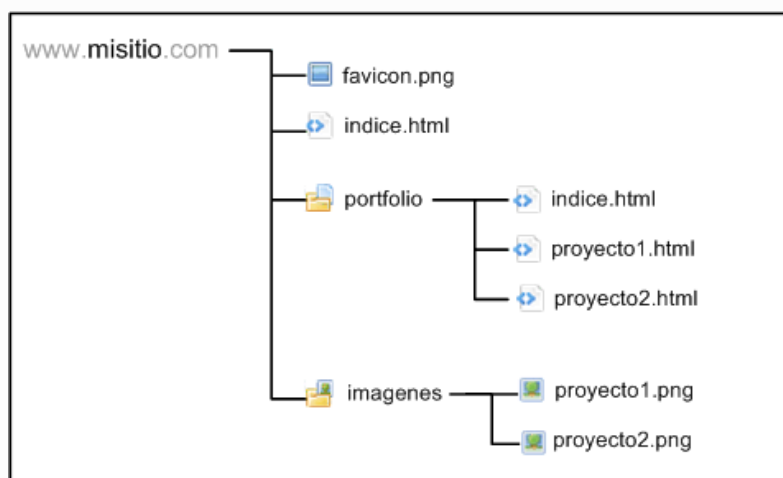


Figura 2.31. Estructura de archivos y directorios de un sitio web de ejemplo

- 1) Crear la siguiente página llamada `indice.html` que sirva como página principal del sitio:



Figura 2.32. Página principal del sitio web de ejemplo

- 2) Crear la página de índice del portfolio:

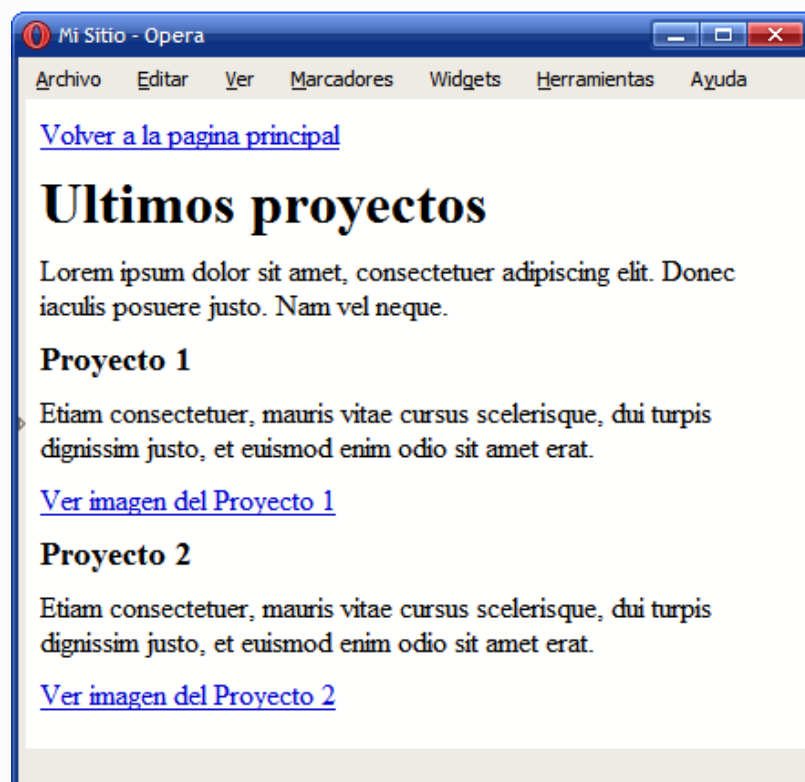


Figura 2.33. Página con la información sobre los proyectos realizados

4.4. Enlaces avanzados

Incluir enlaces básicos mediante la etiqueta `<a>` es muy sencillo. Sin embargo, la definición completa de `<a>` es muy compleja, ya que dispone de varios atributos específicos importantes. A continuación se muestra la definición completa de `<a>`:

<code><a></code>	Enlaces
Atributos comunes	básicos, i18n, eventos y foco
Atributos específicos	<p><code>href</code> = "url" - Indica la URL del recurso que se quiere enlazar</p> <p><code>hreflang</code> = "codigo_idioma" - Idioma del recurso enlazado</p> <p><code>type</code> = "tipo_de_contenido" - Permite "avisar" al navegador sobre el tipo de contenido que se enlaza (imágenes, archivos, etc.) para que pueda prepararse en caso de que no entienda ese contenido</p> <p><code>rel</code> = "tipo_de_relacion" - Describe la relación del documento actual con el recurso enlazado</p> <p><code>rev</code> = "tipo_de_relacion" - Describe la relación del recurso enlazado con el documento actual. No está soportado en HTML5.</p> <p><code>charset</code> = "tipo_de_charset" - Describe la codificación del recurso enlazado. No está soportado en HTML5.</p> <p><code>target</code> = "tipo_de_target" - Indica dónde se abrirá el recurso enlazado.</p> <p><code>media</code> = "tipo_de_medio_y_valor" - Indica para qué dispositivo está optimizado el recurso enlazado.</p>
Tipo de elemento	En línea
Descripción	Se emplea para enlazar todo tipo de recursos

4.4.1. Idioma del enlace (hreflang)

El enlace puede indicar al navegador el idioma del recurso que se enlaza. Para establecer el valor del idioma, se utiliza un código estandarizado de dos letras. Además del idioma genérico, también se puede indicar una variación idiomática. Ejemplo de códigos de idioma más utilizados:

Código	Idioma	Variación idiomática
<code>en</code>	Inglés	-
<code>en-US</code>	Inglés	Estados Unidos

es	Español	-
es-ES	Español	España
es-AR	Español	Argentina

Otros códigos utilizados son: [fr](#) (francés), [de](#) (alemán), [it](#) (italiano), [nl](#) (holandés), [el](#) (griego), [pt](#) (portugués), [ar](#) (árabe), [he](#) (hebreo), [ru](#) (ruso), [zh](#) (chino), [ja](#) (japonés).

La lista completa de códigos de idioma está definida en [el estándar ISO 639](#).

4.4.2. Tipo de contenido (type)

Se utiliza para notificar al navegador sobre el tipo de contenido que se enlaza. Se indica mediante una cadena de texto cuyos posibles valores también están estandarizados. Los valores de los contenidos más utilizados son los siguientes: ["text/html"](#) (páginas HTML), ["image/png"](#) (imágenes con formato PNG), ["image/gif"](#) (imágenes con formato GIF), ["text/css"](#) (hojas de estilo CSS), ["application/rss+xml"](#) (archivos RSS).

La lista completa de tipos de contenido se define en [los estándares RFC 2045 y RFC 2046](#).

4.4.3. Tipo de relación (rel y rev)

Los enlaces pueden proporcionar información adicional muy útil para los navegadores y para los motores de búsqueda como Google. Los atributos [rel](#) y [rev](#) permiten indicar la relación que la página actual tiene con la página a la que se enlaza (atributo [rel](#)) y la relación que tiene la página enlazada con la página actual (atributo [rev](#)). Este último no está soportado en HTML5.

Los tipos de relación definidos son los siguientes:

- [alternate](#) – Indica que es una versión alternativa al documento actual (puede ser una versión en otro idioma o una versión preparada para otro medio, como una impresora o un dispositivo móvil)
- [author](#) – Indica el autor de documento al que enlaza.

- **bookmark** – Establece el enlace actual como un "marcador" o "favorito". Un marcador es un enlace que constituye un punto de entrada muy importante dentro del documento.
- **help** – Indica que es un documento de ayuda.
- **license** – Indica que enlaza con la información de copyright del documento.
- **stylesheet** – Indica que se ha enlazado una hoja de estilos
- **start** – Indica que se trata del primer documento de una colección de documentos (por ejemplo el primer capítulo de un libro). **No soportado en HTML5.**
- **next** – Indica que es el documento que sigue al actual dentro de una secuencia lógica de documentos (por ejemplo, los capítulos de un libro)
- **prev** – Indica que es el documento que precede al actual dentro de una secuencia lógica de documentos (por ejemplo, los capítulos de un libro)
- **contents** – Indica que el recurso enlazado es el documento que contiene la tabla de contenidos de la colección de documentos (por ejemplo, el índice de un libro). **No soportado en HTML5**
- **nofollow** – Indica que enlaza a un documento desaprobado, como un enlace pagado. ("nofollow" es usado por Google, para especificar que el buscador jerárquico de Google no debería seguir ese enlace).
- **prefetch** – Indica que el documento de destino debería ser almacenado en caché.
- **search** – Indica que enlaza a una herramienta de búsqueda para el documento.
- **tag** – Una etiqueta (palabra clave) para el documento actual.

La especificación oficial de HTML5 define la [lista completa de tipos de relaciones](#) que se pueden utilizar.

4.4.4. Tipo de target

Cuando queremos indicar dónde se abrirá el recurso con el que estamos enlazando utilizamos el atributo **target**, donde los posibles valores son: **_blank**, **_parent**, **_self** y **_top**.

- **_blank** – Indica que el hipervínculo direccionado se abrirá en una nueva ventana o pestaña

- `_parent` – Indica que el recurso de destino se abrirá en el frame padre.
- `_self` – Es el parámetro por defecto. Si no se coloca el atributo `target` el recurso será abierto reemplazando a la página actual.
- `_top` – Indica que el recurso de destino se abrirá en el cuerpo completo de la ventana.

4.4.5. Tipo de medio (`media`). Nuevo en HTML5

El atributo `media` especifica para qué dispositivo está optimizado el recurso enlazado. Este atributo se usa para especificar que la URL enlazada está diseñada para dispositivos especiales (como el iPhone), medios de comunicación de habla o dispositivos de impresión. Este atributo acepta varios valores. Sólo se utiliza si el atributo `media` está presente. Este atributo es meramente consultivo y es nuevo en HTML5.

Posibles operadores

Value	Description
and	Specifies an AND operator
not	Specifies a NOT operator
,	Specifies an OR operator

Dispositivos

Value	Description
all	Default. Suitable for all devices
aural	Speech synthesizers
braille	Braille feedback devices
handheld	Handheld devices (small screen, limited bandwidth)
projection	Projectors
print	Print preview mode/printed pages
screen	Computer screens
tty	Teletypes and similar media using a fixed-pitch character grid
tv	Television type devices (low resolution, limited scroll ability)

Valores

Value	Description
width	Specifies the width of the targeted display area. "min-" and "max-" prefixes can be used. Example: <code>media="screen and (min-width:500px)"</code>
height	Specifies the height of the targeted display area. "min-" and "max-" prefixes can be used. Example: <code>media="screen and (max-height:700px)"</code>
device-width	Specifies the width of the target display/paper.

	"min-" and "max-" prefixes can be used. Example: media="screen and (device-width:500px)"
device-height	Specifies the height of the target display/paper. "min-" and "max-" prefixes can be used. Example: media="screen and (device-height:500px)"
orientation	Specifies the orientation of the target display/paper. Possible values: "portrait" or "landscape" Example: media="all and (orientation: landscape)"
aspect-ratio	Specifies the width/height ratio of the targeted display area. "min-" and "max-" prefixes can be used. Example: media="screen and (aspect-ratio:16/9)"
device-aspect-ratio	Specifies the device-width/device-height ratio of the target display/paper. "min-" and "max-" prefixes can be used. Example: media="screen and (aspect-ratio:16/9)"
color	Specifies the bits per color of target display. "min-" and "max-" prefixes can be used. Example: media="screen and (color:3)"
color-index	Specifies the number of colors the target display can handle. "min-" and "max-" prefixes can be used. Example: media="screen and (min-color-index:256)"
monochrome	Specifies the bits per pixel in a monochrome frame buffer. "min-" and "max-" prefixes can be used. Example: media="screen and (monochrome:2)"
resolution	Specifies the pixel density (dpi or dpcm) of the target display/paper. "min-" and "max-" prefixes can be used. Example: media="print and (resolution:300dpi)"
scan	Specifies scanning method of a tv display. Possible values are "progressive" and "interlace". Example: media="tv and (scan:interlace)"
grid	Specifies if the output device is grid or bitmap. Possible values are "1" for grid, and "0" otherwise. Example: media="handheld and (grid:1)"

Ejemplo:

```
<a href="att_a_media.asp?output=print"
media="print and (resolution:300dpi)">
Open media attribute page for print.</a>
```

4.5. Otros tipos de enlaces

Los enlaces mostrados en las secciones anteriores son los más utilizados por las páginas web. Los enlaces creados con la etiqueta `<a>` permiten enlazar cualquier tipo de recurso desde cualquier página. La característica más importante de estos enlaces es que el usuario debe activar la carga de los recursos. En otras palabras, el navegador no carga ningún recurso enlazado con la etiqueta `<a>` a menos que el usuario pinche sobre el enlace.

Además de estos enlaces, las páginas HTML pueden incluir otro tipo de enlaces que cargan los recursos automáticamente. Si una página HTML utiliza archivos CSS para aplicar estilos a sus contenidos, no es lógico que los enlace con la etiqueta `<a>` y espere a que el usuario pinche sobre el enlace para así cargar los archivos CSS. De la misma forma, muchas páginas web dinámicas necesitan que el navegador cargue varios archivos JavaScript para funcionar correctamente.

HTML define las etiquetas `<script>` y `<link>` para enlazar recursos que se deben cargar automáticamente. Cuando el navegador encuentra alguna de estas dos etiquetas, descarga los recursos enlazados y los aplica a la página web.

La etiqueta `<script>` tiene dos modos de funcionamiento, ya que se emplea tanto para insertar un bloque de código JavaScript en la página como para enlazar un archivo JavaScript externo.

<script>	Código ejecutable
Atributos comunes	-
Atributos específicos	<p><code>src = "url"</code> - Indica la dirección del archivo que contiene el código</p> <p><code>type = "tipo_de_contenido"</code> - Permite "avisar" al navegador sobre el tipo de código que se incluye (normalmente JavaScript)</p> <p><code>defer = "defer"</code> - El código no va a modificar el contenido de la página web</p> <p><code>charset = "tipo_de_charset"</code> - Describe la codificación del código enlazado</p>
Tipo de elemento	Bloque y en línea (también puede ser una etiqueta vacía)
Descripción	Se emplea para enlazar o definir un bloque de código (normalmente JavaScript)

Aunque la etiqueta `<script>` permite enlazar código de varios lenguajes de programación, el uso habitual de `<script>` consiste en enlazar un archivo JavaScript externo:

```
<head>
  <script type="text/javascript"
src="http://www.ejemplo.com/js/inicializar.js"></script>
</head>
```

El atributo `type` utilizado habitualmente para los archivos JavaScript es `"text/javascript"`. El atributo `src` es equivalente al atributo `href` de los enlaces creados con la etiqueta `<a>`. La URL indicada en el atributo `src` puede ser absoluta o relativa y externa o interna.

Además de enlazar un archivo JavaScript externo, la misma etiqueta `<script>` también permite incluir en la página web un bloque de código JavaScript:

```
<html>
<head>
  <script type="text/javascript">
    //
      window.onload = function() { alert("La página se ha cargado
completamente"); }
    //]]&gt;
  &lt;/script&gt;
&lt;/head&gt;
&lt;body&gt;
  ...
&lt;/body&gt;
&lt;/html&gt;</pre></div><div data-bbox="138 645 863 754" data-label="Text"><p>Cuando se incluye código JavaScript en la propia página XHTML, se debe insertar dentro de una sección especial llamada <code>CDATA</code>. Para ello, el código JavaScript se debe encerrar entre <code>&lt;![CDATA[</code> y <code>]]&gt;</code>. Cuando el navegador encuentra una sección de este tipo, no procesa su contenido como si fuera XHTML y por tanto no tiene en cuenta los posibles errores de validación de XHTML.</p></div><div data-bbox="138 773 863 881" data-label="Text"><p>De esta forma, se pueden construir páginas XHTML válidas y código JavaScript correcto. En los capítulos posteriores se profundiza en el concepto de validación de páginas XHTML. Los caracteres <code>//</code> al comienzo y al final de la sección <code>CDATA</code> son necesarios para los navegadores que no son capaces de procesar correctamente estas secciones.</p></div><div data-bbox="484 938 511 955" data-label="Page-Footer"><p>88</p></div>
```

La etiqueta `<script>` (tanto cuando enlaza, como cuando incluye directamente el código) puede aparecer en cualquier parte del documento HTML, aunque normalmente se incluye dentro de la cabecera de la página (`<head>...</head>`).

La segunda etiqueta de XHTML para enlazar recursos es `<link>`, que permite enlazar y relacionar la página con otros recursos externos.

<link>	Enlazar recursos
Atributos comunes	básicos, i18n y eventos
Atributos específicos	Los siguientes con el mismo significado que para la etiqueta "a": charset, href, hreflang, type, rel y rev <code>media = "tipo_de_medio"</code> - Indica el medio para el que debe aplicarse la relación
Tipo de elemento	Etiqueta vacía
Descripción	Se emplea para enlazar y establecer relaciones entre el documento y otros recursos

Al contrario que `<script>`, la etiqueta `<link>` solamente se puede incluir dentro de la cabecera del documento. Se pueden añadir tantas etiquetas `<link>` como sean necesarias, pero siempre dentro de `<head>...</head>`.

El atributo `media` hace referencia al medio para el que es válida la relación con el recurso enlazado. Los medios disponibles también están estandarizados, siendo los más comunes `screen` para los contenidos mostrados en pantalla, `print` para las impresoras y `handheld` para los dispositivos móviles.

El uso habitual de la etiqueta `<link>` es el de enlazar las hojas de estilos CSS utilizadas por las páginas web:

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

En este caso, es habitual establecer los atributos `rel` y `type` para indicar el tipo de recurso enlazado y su relación con la página web. La URL del recurso enlazado se indica en el atributo `href`, que admite tanto URL absolutas como relativas.

4.6. Ejemplos de enlaces habituales

4.6.1. Enlace al inicio del sitio web

```
<a href="/">Inicio</a>
```

Al pulsar el enlace anterior desde cualquier página web, se vuelve directamente a la página de inicio, *home* o página principal del sitio web.

4.6.2. Enlace a un email

```
<a href="mailto:nombre@direccion.com" title="Dirección de email  
para solicitar más información">  
Solicita más información  
</a>
```

Al pinchar sobre el enlace anterior, se abre automáticamente el programa de correo electrónico del ordenador del usuario y se establece la dirección de envío al valor indicado después de `mailto:`. La sintaxis es la misma que la de un enlace normal, salvo que se cambia el prefijo `http://` por `mailto:`.

La sintaxis de `mailto:` permite utilizarlo para otros ejemplos más complejos:

```
<!-- Envío del correo electrónico a varias direcciones a la vez  
-->  
<a  
href="mailto:nombre@direccion.com,otro_nombre@direccion.com">Sol  
icita más información</a>  
  
<!-- Añadir un "asunto" inicial al correo electrónico -->  
<a href="mailto:nombre@direccion.com?subject=Solicitud de más  
información">Solicita más información</a>  
  
<!-- Añadir un texto inicial en el cuerpo del correo electrónico  
-->  
<a href="mailto:nombre@direccion.com?body=Estaría interesado en  
solicitar más información sobre sus productos">Solicita más  
información</a>
```

Todas las opciones anteriores se pueden combinar entre sí para realizar ejemplos más avanzados. Aunque el uso de `mailto:` puede parecer una ventaja, su uso está desaconsejado. Si se incluye una dirección de correo electrónico directamente en una página web, es muy probable que en poco tiempo esa dirección de email se encuentre llena de correo electrónico basura o "*spam*", ya que existen programas automáticos encargados de rastrear sistemáticamente todas las páginas web de Internet para encontrar direcciones de correo electrónico válidas.

La forma de mostrar las direcciones de correo electrónico en las páginas web consiste en incluir la dirección en una imagen o indicarla de forma que solamente los usuarios puedan entenderlo:

```
<p>La dirección de correo es <strong>nombre (arroba)
direccion.com</strong></p>
<p>La dirección de correo es
<strong>nombre_arroba_direccion.com</strong></p>
<p>La dirección de correo es
<strong>nombreQUITAESTO@direccion.com</strong></p>
<p>La dirección de correo es
<strong>nombre(ARROBA)direccion.com</strong></p>
<p>La dirección de correo es <strong>nombre @ direccion .
com</strong></p>
```

4.6.3. Enlace a un archivo FTP

Para enlazar un archivo almacenado en un servidor FTP, la parte del protocolo de la URL debe cambiar de `http://` a `ftp://`:

```
<a href="ftp://ftp.ejemplo.com/ruta/archivo.zip" title="Archivo
comprimido de los contenidos">
Descarga un ZIP con todos los contenidos
</a>
```

4.6.4. Enlazar varias hojas de estilos CSS

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

4.6.5. Enlazar hojas de estilos CSS para diferentes medios

```
<link rel="stylesheet" type="text/css" href="/css/comun.css"
media="screen, projection" />
<link rel="stylesheet" type="text/css" href="/css/impresora.css"
media="print" />
<link rel="stylesheet" type="text/css" href="/css/movil.css"
media="handheld" />
```

4.6.6. Enlazar el favicon

El *favicon* o icono para favoritos es el pequeño icono que muestran las páginas en varias partes del navegador. Dependiendo del navegador que se utilice, este icono se muestra en la barra de direcciones, en la barra de título del navegador y/o en el menú de favoritos/marcadores.

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

Aunque en principio la imagen debería ser de tipo **.ICO** (formato gráfico de los iconos), algunos navegadores soportan favicons en otros formatos gráficos más habituales (como por ejemplo **.PNG**).

4.6.7. Enlazar un archivo RSS

```
<link rel="alternate" type="application/rss+xml" title="Resumen de todos los artículos del blog" href="/feed.xml" />
```

4.6.8. Enlazar hojas de estilos, favicon y RSS

En una misma página se pueden incluir varias etiquetas `<link>`, por lo que es habitual que las páginas enlacen hojas de estilos, favicon y archivos RSS de forma conjunta:

```
<head>
...
<link rel="stylesheet" type="text/css" href="/css/impresora.css"
media="print" />
<link rel="stylesheet" type="text/css" href="/css/movil.css"
media="handheld" />
<style type="text/css" media="screen,projection">
  @import '/css/main.css';
</style>
<link rel="shortcut icon" href="/favicon.ico" type="image/ico"
/>
<link rel="alternate" type="application/rss+xml" title="Resumen
de todos los artículos del blog" href="/feed.xml" />
...
</head>
```

4.6.9. Indicar que existe una versión de la página en otro idioma

```
<head>
<title>English tutorial</title>
<link lang="es" xml:lang="es" title="El tutorial en español"
type="text/html" rel="alternate" hreflang="es"
href="http://www.ejemplo.com/tutorial/espanol.html" />
</head>
```

4.6.10. Indicar que existe una versión de la página preparada para imprimir

```
<head>
<link media="print" title="El tutorial en PDF"
type="application/pdf" rel="alternate"
href="http://www.ejemplo.com/tutorial/documento.pdf" />
</head>
```

4.6.11. Indicar que existe una página que es índice de la página actual

```
<head>
<title>Tutorial - Capítulo 5</title>
<link rel="start" title="El índice del tutorial"
type="text/html"
href="http://www.ejemplo.com/tutorial/indice.html" />
</head>
```

Ejercicio 7: Enlazar el favicon en todas las páginas del ejercicio 6 y añadir todos los atributos posibles a los enlaces.

5. LISTAS

En ocasiones, es posible agrupar determinadas palabras o frases en un conjunto de elementos que tienen más significado de forma conjunta. El menú de navegación de un sitio web por ejemplo está formado por un grupo de palabras. Aunque cada palabra por separado tiene sentido, de forma conjunta constituyen el menú de navegación de la página, por lo que su significado conjunto es mayor que por separado.

El lenguaje HTML define tres tipos diferentes de listas para agrupar los elementos: listas no ordenadas (se trata de una colección simple de elementos en la que no importa su orden), listas ordenadas (similar a la anterior, pero los elementos están numerados y por tanto, importa su orden) y listas de definición (un conjunto de términos y definiciones similar a un diccionario).

5.1. Listas no ordenadas

Las listas no ordenadas son las más sencillas y las que más se utilizan. Una lista no ordenada es un conjunto de elementos relacionados entre sí pero para los que no se indica un orden o secuencia determinados. La etiqueta `` encierra todos los elementos de la lista y la etiqueta `` cada uno de sus elementos.

	Lista no ordenada
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir listas no ordenadas

	Elemento de una lista
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir los elementos de las listas (ordenadas y no ordenadas)

El siguiente código HTML muestra un ejemplo sencillo de lista no ordenada:

```
<!DOCTYPE html>

<html>
<head><title>Ejemplo de etiqueta UL</title></head>
<body>

<h1>Menú</h1>

<ul>
  <li>Inicio</li>
  <li>Noticias</li>
  <li>Artículos</li>
  <li>Contacto</li>
</ul>

</body>
</html>
```

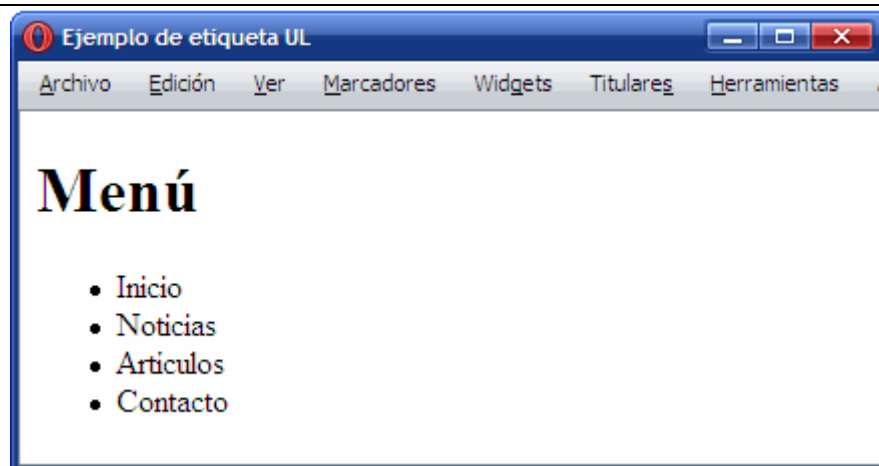


Figura 2.34. Ejemplo de uso de la etiqueta ul

El navegador por defecto muestra los elementos de la lista tabulados y con una pequeña viñeta formada por un círculo negro. Como ya se sabe, el aspecto con el que se muestran los elementos de las listas se puede modificar mediante las hojas de estilos CSS.

5.2. Listas ordenadas

Las listas ordenadas son casi idénticas a las listas no ordenadas, salvo que en este caso los elementos relacionados se muestran siguiendo un orden determinado. Cuando se crea por ejemplo una lista con las instrucciones de un producto, es importante el orden en el que se realiza cada paso. Cuando se muestra un índice o tabla de contenidos en un libro, es importante el orden de cada elemento del índice.

En todos estos casos, la lista más adecuada es la lista ordenada, que se define mediante la etiqueta ``. Los elementos de la lista se definen mediante la etiqueta ``, la misma que se utiliza en las listas no ordenadas.

<code></code>	Lista ordenada
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir listas ordenadas

El siguiente código HTML muestra un ejemplo sencillo de lista ordenada:


```
<!DOCTYPE html>

<html>
<head><title>Ejemplo de etiqueta OL</title></head>
<body>

<h1>Instrucciones</h1>

<ol>
  <li>Enchufar correctamente</li>
  <li>Comprobar conexiones</li>
  <li>Encender el aparato</li>
</ol>

</body>
</html>
```

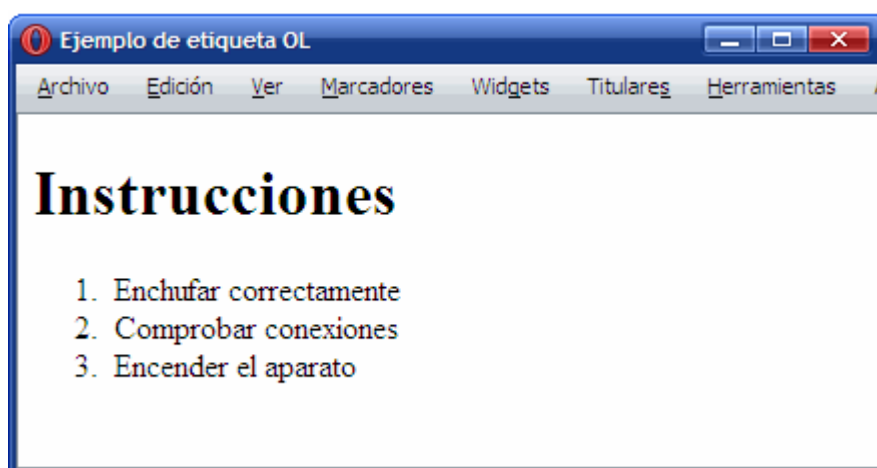


Figura 2.35. Ejemplo de uso de la etiqueta ol

El navegador muestra la lista de forma muy parecida a las listas no ordenadas, salvo que en este caso no se emplean viñetas gráficas en los elementos, sino que se numeran de forma consecutiva. El tipo de numeración empleada también se puede modificar aplicando hojas de estilos CSS a los elementos de la lista.

5.3. Listas de definición

Las listas de definición apenas se utilizan en la mayoría de páginas HTML. Su funcionamiento es similar al de un diccionario, ya que cada elemento de la lista está formado por términos y definiciones. La etiqueta `<dl>` crea la lista de definición y las etiquetas `<dt>` y `<dd>` definen respectivamente el término y la descripción de cada elemento de la lista.

<dl>	Lista de definición
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir listas de definición

<dt>	Término de una definición
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir los términos de los elementos de una lista de definición

<dd>	Descripción de una definición
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para indicar las definiciones de los elementos de una lista de definición

El siguiente código HTML muestra un ejemplo sencillo de lista de definición:

```
<!DOCTYPE html>
<html>
<head><title>Ejemplo de etiqueta DL</title></head>
<body>
<h1>Metalenguajes</h1>

<dl>
  <dt>SGML</dt>
```

```
<dd>Metalenguaje para la definición de otros lenguajes de  
marcado</dd>  
  
<dt>XML</dt>  
<dd>Lenguaje basado en SGML y que se emplea para describir  
datos</dd>  
  
<dt>RSS</dt>  
<dt>GML</dt>  
<dt>XHTML</dt>  
<dt>SVG</dt>  
<dt>XUL</dt>  
<dd>Lenguajes derivados de XML para determinadas  
aplicaciones</dd>  
</dl>  
</body>  
</html>
```

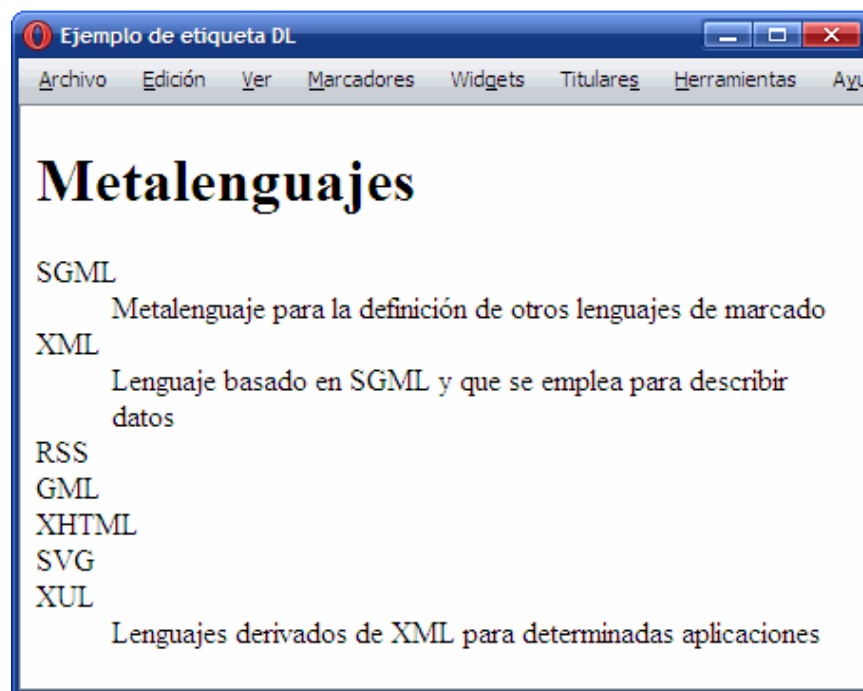


Figura 2.36. Ejemplo de uso de la etiqueta dl

Los navegadores formatean las listas de definición de forma similar a las otras listas, tabulando la definición y alineando a la izquierda los términos. Aunque no es habitual, cada término puede tener asociada más de una definición y cada definición puede tener asociada varios términos.

Ejercicio 8: Determinar el código HTML que corresponde a la siguiente lista anidada simple



Figura 2.37. Ejemplo de lista anidada simple de dos niveles

Ejercicio 9: Determinar el código HTML que corresponde a la siguiente lista anidada compleja



Figura 2.38. Ejemplo de lista anidada compleja de dos niveles

6. IMÁGENES Y OBJETOS

6.1. Imágenes

Las imágenes son uno de los elementos más importantes de las páginas web. De hecho, prácticamente todas las páginas web contienen alguna imagen y la mayoría incluyen decenas de imágenes. Dentro de las imágenes que se pueden incluir en una página HTML se deben distinguir dos tipos: las imágenes de contenido y las imágenes *de adorno*.

Las **imágenes de contenido** son las que proporcionan información y complementan la información textual. Las **imágenes de adorno** son las que se utilizan para hacer bordes redondeados, para mostrar pequeños iconos en las listas de elementos, para mostrar fondos de página, etc. Las imágenes de contenido se incluyen directamente en el código HTML mediante la etiqueta `` y las imágenes de adorno no se deberían incluir en el código HTML, sino que deberían emplearse hojas de estilos CSS para mostrarlas.

A continuación se muestra la definición de la etiqueta ``, utilizada para incluir las imágenes en las páginas HTML:

	Imagen
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<code>src = "url"</code> - Indica la URL de la imagen que se muestra <code>alt = "texto"</code> - Descripción corta de la imagen <code>longdesc = "url"</code> - Indica una URL en la que puede encontrarse una descripción más detallada de la imagen <code>name = "texto"</code> - Nombre del elemento imagen <code>height = "unidad_de_medida"</code> - Indica la altura con la que se debe mostrar la imagen (no es obligatorio que coincida con la altura original de la imagen) <code>width = "unidad_de_medida"</code> - Indica la anchura con la que se debe mostrar la imagen (no es obligatorio que coincida con la anchura original de la imagen)
Tipo de elemento	En línea y etiqueta vacía
Descripción	Se emplea para incluir imágenes en los documentos

Los dos atributos requeridos son `src` y `alt`. El atributo `src` es similar al atributo `href` de los enlaces, ya que establece la URL de la imagen que se va a mostrar en la página. Las URL indicadas pueden ser absolutas o relativas. El atributo `alt` permite describir el contenido de la imagen mediante un texto breve. Las descripciones deben tener una longitud inferior a 1024 caracteres y son útiles para las personas y dispositivos discapacitados que no pueden acceder a las imágenes.

Ejemplo sencillo para incluir una imagen:

```

```

Como `` es una etiqueta vacía, no tiene etiqueta de cierre. No obstante, para que la página XHTML sea válida, todas las etiquetas deben estar cerradas. Como ya se explicó anteriormente, para cerrar una etiqueta vacía se incluyen los caracteres `</>` al final de la etiqueta.

HTML no impone ninguna restricción sobre el formato gráfico que se puede utilizar en las imágenes, por lo que en principio la etiqueta `` puede incluir cualquier formato gráfico existente. Sin embargo, si la imagen utiliza un formato poco habitual, todos o algunos navegadores no serán capaces de mostrar esa imagen.

La recomendación es utilizar uno de los tres siguientes formatos gráficos que entienden todos los navegadores modernos: GIF, JPG y PNG. El formato PNG presenta el inconveniente de que los navegadores obsoletos como Internet Explorer 6 no muestran correctamente las imágenes con transparencias de 24 bits.

El atributo `longdesc` no se utiliza de forma habitual, pero permite indicar la URL en la que se puede encontrar más información sobre la imagen. Como el atributo `alt` sólo permite incluir descripciones de hasta 1024 caracteres, el atributo `longdesc` se emplea con las imágenes complejas que necesitan mucha información para ser descritas:

```



```

En el ejemplo anterior, las dos imágenes se encuentran en el mismo directorio del servidor (`/imagenes/`). Se trata de una estrategia habitual en la mayoría de sitios web: guardar todas las imágenes de contenido en un directorio especial independiente del resto de contenidos HTML. Además, el directorio siempre suele llamarse de la misma manera: `"imagenes"` o `"images"` en inglés.

Los atributos `width` y `height` se utilizan para indicar la anchura y altura con la que se muestran las imágenes, por lo que son los más contradictorios. Como ya se ha comentado, HTML estructura de forma correcta los contenidos de la página y CSS define el aspecto gráfico con el que se muestran los contenidos. En principio, la anchura y la altura con la que se muestra una imagen es parte de su aspecto gráfico, por lo que debería ser propio de CSS y no de XHTML.

Sin embargo, en la práctica no es viable establecer la anchura y altura de todas las imágenes de contenidos mediante CSS. Si el sitio web dispone de muchas imágenes, la sobrecarga de estilos diferentes que debería definir CSS sería contraproducente. Por este motivo, los atributos `width` y `height` son la excepción a la norma de que el código HTML no haga referencia al aspecto de los contenidos.

```



```

Si el valor del atributo `width` o `height` se indica mediante un número entero, el navegador supone que hace referencia a la unidad de medida píxel. Por tanto, en el ejemplo anterior, la primera foto se muestra con una anchura de 200 píxel y una altura de 350 píxel.

También es posible indicar la anchura y altura en forma de porcentaje. En este caso, el porcentaje hace referencia a la altura/anchura del elemento en el que está contenida la imagen. Si la imagen no se encuentra dentro de ningún otro elemento, hace referencia a la anchura/altura total de la página.

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

El ejemplo anterior mezcla los dos tipos de medidas que se pueden utilizar, para indicar que la foto tiene una anchura igual al 30% de la anchura del elemento `<div>` que la contiene y una altura de 350 píxel.

La anchura/altura con la que se muestra una imagen no tiene que coincidir obligatoriamente con la anchura/altura real de la imagen. Sin embargo, cuando estos valores no coinciden, las imágenes se muestran deformadas y el aspecto final es muy desagradable.

Si solamente se establece la altura de la imagen, el navegador calcula la anchura necesaria para que se mantenga la proporción de la imagen. De la misma forma, si sólo se establece la anchura de la imagen, el navegador calcula la altura que hace que la imagen se siga viendo con las mismas proporciones.

Ejercicio 10: Modificar la página de índice del portfolio de los ejercicios 6 y 7 para mostrar directamente las imágenes de los proyectos.



Figura 2.39. Nueva página del portfolio que muestra la imagen de cada uno de los proyectos

6.2. Etiquetas figure y figcaption

Estos nuevos elementos de HTML5 permiten anotar o resaltar ilustraciones, diagramas o fotos que son referenciados directamente desde el contenido principal.

La etiqueta `<figure>` especifica contenido independiente, como ilustraciones, diagramas, fotografías, listados de código, etc. Mientras que el contenido del elemento `<figure>` está relacionado con el contenedor principal, su posición es independiente del contenedor principal y si se elimina no debería afectar al documento.

La etiqueta `<figcaption>` se utiliza para añadir una leyenda o título para el elemento `<figure>`.

Ambas etiquetas soportan los atributos básicos y de eventos.

Ejemplo:

```
<figure>
  
  <figcaption>Fig1. - Una vista de las cataratas del
Niágara.</figcaption>
</figure>
```

Ejercicio 11: Modificar la página de índice del portfolio del ejercicio 10 para mostrar las imágenes de los proyectos con sus respectivas leyendas al pie de las mismas.

[Volver a la pagina principal](#)

Ultimos proyectos

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec iaculis posuere ju

Proyecto 1

Etiam consectetur, mauris vitae cursus scelerisque, dui turpis dignissim justo, et



Imagen del Proyecto 1

Proyecto 2

Etiam consectetur, mauris vitae cursus scelerisque, dui turpis dignissim justo, et



Imagen del Proyecto 2

Figura 2.40. Nueva página del portfolio que muestra la imagen con leyenda de cada uno de los proyectos

6.3. Mapas de imagen

Aunque el uso de los mapas de imagen se ha reducido drásticamente en los últimos años, aún se utilizan en algunos sitios especializados. Muchas agencias de viaje y sitios relacionados utilizan mapas geográficos para seleccionar el destino del viaje. La mayoría de mapas se realiza hoy en día mediante Flash, aunque algunos sitios siguen recurriendo a los mapas de imagen y con HTML5 puede que se vuelva a ellos.

Un mapa de imagen permite definir diferentes zonas *"pinchables"* dentro de una imagen. El usuario puede pinchar sobre cada una de las zonas definidas y cada una de ellas puede apuntar a una URL diferente. Siguiendo el ejemplo anterior, una sola imagen que muestre un mapa de todos los continentes puede definir una zona diferente para cada continente. De esta forma, el usuario puede pinchar sobre la zona correspondiente a cada continente para que el navegador muestre la página que contiene los viajes disponibles a ese destino.

Las zonas o regiones que se pueden definir en una imagen se crean mediante rectángulos, círculos y polígonos. Para crear un mapa de imagen, en primer lugar se inserta la imagen original mediante la etiqueta ``. A continuación, se utiliza la

etiqueta `<map>` para definir las zonas o regiones de la imagen. Cada zona se define mediante la etiqueta `<area>`.

<map>	Mapa de imagen
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<code>name = "texto"</code> - Nombre que identifica de forma única al mapa definido (es obligatorio indicar un nombre único)
Tipo de elemento	Bloque y en línea
Descripción	Se emplea para definir mapas de imagen

<area>	Area de un mapa de imagen
Atributos comunes	básicos, i18n, eventos y foco
Atributos específicos	<p><code>href = "url"</code> - URL a la que se accede al pinchar sobre el área</p> <p><code>nohref = "nohref"</code> - Se emplea para las áreas que no son seleccionables</p> <p><code>shape = "default rect circle poly"</code> - Indica el tipo de área que se define (toda la imagen, rectangular, circular o poligonal)</p> <p><code>coords = "lista de números"</code> - Se trata de una lista de números separados por comas que representan las coordenadas del área. Rectangular = X1,Y1,X2,Y2 (coordenadas X e Y del vértice superior izquierdo y coordenadas X e Y del vértice inferior derecho). Circular = X1,Y1,R (coordenadas X e Y del centro y radio del círculo). Poligonal = X1,Y1,X2,Y2,...,XnYn (coordenadas de los vértices del polígono. Si las últimas coordenadas no son iguales que las primeras, se cierra automáticamente el polígono uniendo ambos vértices)</p>
Tipo de elemento	Etiqueta vacía
Descripción	Se emplea para definir las distintas áreas que forman un mapa de imagen

Si una imagen utiliza un mapa de imagen, debe indicarlo mediante el atributo usemap. El valor del atributo debe ser el nombre del mapa de imagen definido en otra parte del mismo documento HTML:

```

...
<map name="continentes">
  ...
</map>
```

Las áreas se definen mediante el atributo `shape` que indica el tipo de área y `coords` que es una lista de coordenadas cuyo significado depende del tipo de área definido. El enlace de cada área se define mediante el atributo `href`, con la misma sintaxis y significado que para los enlaces normales.

El siguiente ejemplo muestra una imagen sencilla que contiene cuatro figuras geométricas:

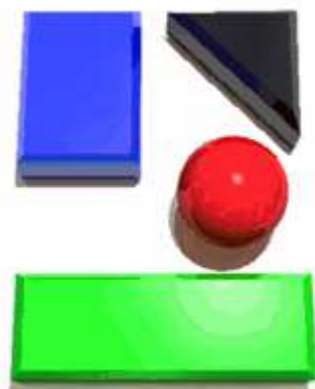


Figura 2.41. Ejemplo de imagen que incluye un mapa de imagen

Utilizando un círculo, dos rectángulos y un polígono se pueden definir fácilmente cuatro zonas *pinchables* en la imagen mediante el siguiente código HTML:

```


<map name="mapa_zonas">
  <area shape="rect" coords="20,25,84,113"
href="rectangulo.html" />
  <area shape="polygon" coords="90,25,162,26,163,96,89,25,90,24"
href="triangulo.html" />
  <area shape="circle" coords="130,114,29" href="circulo.html"
/>
  <area shape="rect" coords="19,156,170,211"
href="mailto:rectangulo@direccion.com" />
  <area shape="default" nohref="nohref" />
</map>
```

6.3. Objetos

Además de las imágenes, HTML permite incluir en las páginas web otros elementos mucho más complejos, como *applets* de Java y vídeos en formato QuickTime o Flash. La mayoría de este tipo de contenidos no los interpreta el navegador directamente, sino que hace uso de pequeños programas llamados *plugins* y que se encargan de tratar con este tipo de elementos complejos.

La etiqueta `<object>` es la que permite "embeber" o incluir en las páginas HTML cualquier tipo de contenido complejo:

<object>	Objeto
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<p><code>data = "url"</code> - Indica la URL de los datos que utiliza el objeto</p> <p><code>classid</code>, <code>codebase</code>, <code>codetype</code> - Información específica que depende del tipo de objeto</p> <p><code>type</code> - Indica el tipo de contenido de los datos</p> <p><code>height = "unidad_de_medida"</code> - Indica la altura con la que se debe mostrar el objeto</p> <p><code>width = "unidad_de_medida"</code> - Indica la anchura con la que se debe mostrar el objeto</p>
Tipo de elemento	Bloque y en línea
Descripción	Se emplea para embeber objetos en los documentos

El atributo `data` se emplea para indicar la URL del recurso que se va a incluir. El atributo `type` indica el tipo de contenido de los datos del objeto. Los posibles valores de `type` están estandarizados y coinciden con los del atributo `type` de la etiqueta `<a>` que se explicó anteriormente.

El propio estándar de HTML incluye ejemplos de uso de esta etiqueta. Incluir un vídeo en formato MPEG:

```
<object data="PlanetaTierra.mpeg" type="application/mpeg" />
```

También se pueden incluir varias versiones alternativas de un mismo contenido. Así, si el navegador no es capaz de interpretar el formato por defecto, puede optar por cualquiera de los otros formatos alternativos:

```
<object title="La Tierra vista desde el espacio"
classid="http://www.observer.mars/TheEarth.py">
  <!-- Formato alternativo en forma de vídeo -->
  <object data="PlanetaTierra.mpeg" type="application/mpeg">
    <!-- Otro formato alternativo mediante una imagen GIF -->
    <object data="PlanetaTierra.gif" type="image/gif">
      <!-- Si el navegador no soporta ningún formato, se muestra
el siguiente texto -->
      La <strong>Tierra</strong> vista desde el espacio.
    </object>
  </object>
</object>
```

A los objetos también se les puede pasar información adicional en forma de parámetros mediante la etiqueta `<param>`:

<param>	Parámetros de un objeto
Atributos comunes	id
Atributos específicos	name = "texto" - Indica el nombre del parámetro value = "texto" - Indica el valor del parámetro
Tipo de elemento	Etiqueta vacía
Descripción	Se emplea para indicar el valor de los parámetros del objeto

Las etiquetas `<param>` siempre se incluyen en el interior de las etiquetas `<object>`:

```
<object data="..." type="...">
  <param name="parametro1" value="40" />
  <param name="parametro2" value="20" />
  <param name="parametro3" value="texto de prueba" />
</object>
```

Uno de los principales inconvenientes de `<object>` es la forma de incluir vídeos en formato Flash en las páginas HTML. Si se utiliza el siguiente código:

```
<object data="nombre_video.swf" type="application/x-shockwave-
flash"></object>
```

El elemento anterior es correcto desde el punto de vista técnico, pero provoca que algunos navegadores como Internet Explorer no visualicen el vídeo hasta que se ha descargado completamente. Si se trata de un vídeo largo, esta solución no es válida para el usuario.

Por este motivo, se utiliza una solución alternativa para incluir vídeos Flash en las páginas HTML: el uso de la etiqueta `<embed>`. Aunque esta solución funciona correctamente, no se trata de una solución válida desde el punto de vista del estándar de XHTML, por lo que las páginas que incluyan esta solución no pasarán correctamente el proceso de validación.

<embed>	Embeber objetos
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<p><code>src = "url"</code> - Indica la URL del archivo u objeto que se incluye en la página</p> <p><code>type = "tipo_de_contenido"</code> - Indica el tipo de contenido del objeto (flash, quicktime, java, etc.)</p> <p><code>height = "unidad_de_medida"</code> - Indica la altura con la que se debe mostrar el objeto</p> <p><code>width = "unidad_de_medida"</code> - Indica la anchura con la que se debe mostrar el objeto</p>
Tipo de elemento	Bloque
Descripción	Se emplea para embeber objetos en los documentos

Este es el motivo por el que los sitios web más populares de vídeos en formato Flash proporcionan un código similar al siguiente para incluir sus vídeos en las páginas HTML:

```
<object width="425" height="350">
  <param name="movie"
value="http://www.youtube.com/v/MsH0rBWCYjs"></param>
  <param name="wmode" value="transparent"></param>
  <embed src="http://www.youtube.com/v/MsH0rBWCYjs"
type="application/x-shockwave-flash" wmode="transparent"
width="425" height="350"></embed>
</object>
```

Una vez más, se debe tener en cuenta que la solución anterior de utilizar la etiqueta `<embed>` es correcta desde el punto de vista del usuario (no tiene que esperar a que el vídeo se descargue completamente para poder verlo) pero no es una solución técnicamente válida, ya que la etiqueta `<embed>` no es parte del estándar XHTML.

7. TABLAS

Desde sus primeras versiones, HTML incluyó el soporte para crear tablas de datos en las páginas web. Además de ser sencillo, el modelo definido por HTML es muy flexible y bastante completo.

Las tablas en HTML utilizan los mismos conceptos de filas, columnas, cabeceras y títulos que los que se utilizan en cualquier otro entorno de publicación de documentos:

El diagrama muestra una tabla HTML con el título "Cursos de diseño gráfico". La tabla tiene 4 columnas: "Nombre", "Horas", "Plazas" y "Horario". Las filas de datos son: "Introducción a XHTML", "CSS avanzado", "Taller de usabilidad" y "Introducción a AJAX". Las etiquetas de partes son: "título de tabla" (punto a la cabecera de la tabla), "cabecera de columna" (punto a una columna), "cabecera de tabla" (punto a la fila de encabezados), "fila" (punto a una fila de datos), "cabecera de fila" (punto a una celda de la fila de encabezados) y "columna" (punto a una celda de una fila de datos).

Nombre	Horas	Plazas	Horario
Introducción a XHTML	20	20	09:00 – 13:00
CSS avanzado	40	15	16:00 – 20:00
Taller de usabilidad	40	10	16:00 – 20:00
Introducción a AJAX	60	20	08:30 – 12:30

Figura 2.42. Partes que componen una tabla compleja

Las tablas de HTML pueden contener elementos simples, agrupaciones de filas y de columnas, cabeceras y pies de tabla, subdivisiones, cabeceras múltiples y otros elementos complejos.

A pesar de que las tablas HTML son fáciles de comprender y utilizar, son uno de los elementos más polémicos de HTML. El problema de las tablas es que no siempre se utilizan adecuadamente. Aunque parezca obvio, las tablas se deben utilizar para mostrar información tabular.

Hasta hace unos años, las tablas también se utilizaban para definir la estructura de las páginas web. La cabecera de la página era una fila de una gran tabla, el pie de página era otra fila de esta tabla y la zona de contenidos estaba formada por varias columnas dentro de esa gran tabla.

Aunque algunos malos diseñadores siguen utilizando hoy en día las tablas para definir la estructura completa de las páginas web, se trata de una técnica obsoleta y nada recomendable. El motivo es que se complica en exceso el código HTML y su mantenimiento es muy complejo. La solución correcta para definir la estructura de las páginas consiste en la utilización de hojas de estilos CSS.

7.1. Tablas básicas

Las tablas más sencillas de HTML se definen con tres etiquetas: `<table>` para crear la tabla, `<tr>` para crear cada fila y `<td>` para crear cada columna.

A continuación se muestra el código HTML de una tabla sencilla:

```
<html>
<head><title>Ejemplo de tabla sencilla</title></head>
<body>

<h1>Listado de cursos</h1>

<table>
<tr>
  <td><strong>Curso</strong></td>
  <td><strong>Horas</strong></td>
  <td><strong>Horario</strong></td>
</tr>

<tr>
  <td>CSS</td>
  <td>20</td>
  <td>16:00 - 20:00</td>
</tr>

<tr>
  <td>HTML</td>
  <td>20</td>
  <td>16:00 - 20:00</td>
</tr>

<tr>
  <td>Dreamweaver</td>
  <td>60</td>
  <td>16:00 - 20:00</td>
</tr>
</table>

</body>
</html>
```

Si se visualiza el código anterior en cualquier navegador, se obtiene una tabla como la que muestra la siguiente imagen:

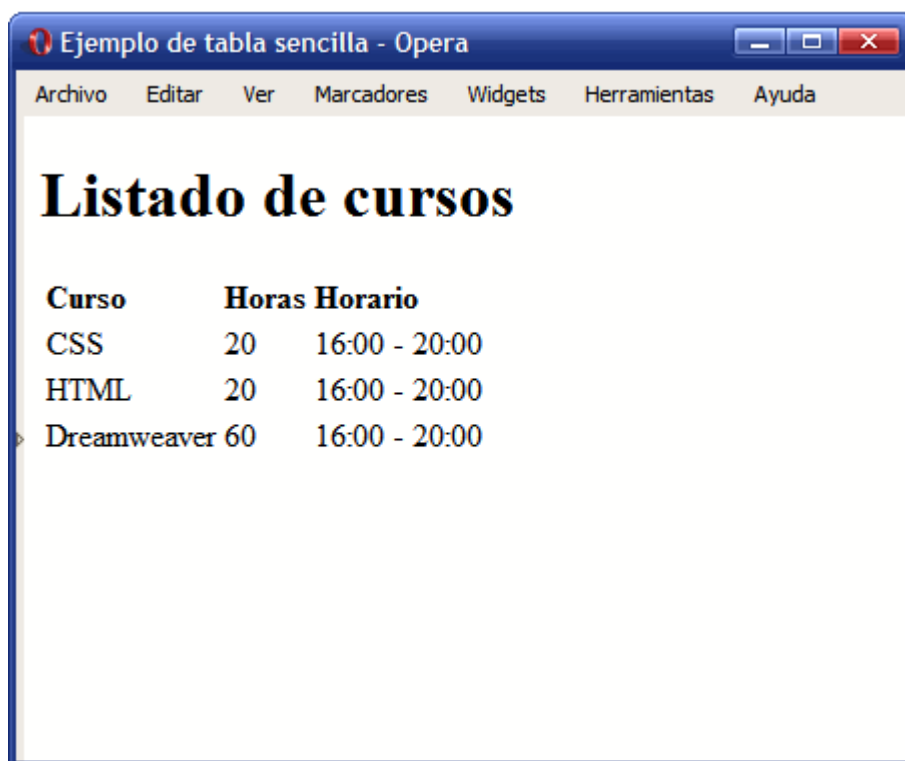


Figura 2.43. Ejemplo de tabla sencilla creada con las etiquetas `table`, `tr` y `td`

La etiqueta `<table>` encierra todas las filas y columnas de la tabla. Las etiquetas `<tr>` (del inglés "table row") definen cada fila de la tabla y encierran todas las columnas. Por último, la etiqueta `<td>` (del inglés "table data cell") define cada una de las columnas de las filas, aunque realmente HTML no define columnas sino celdas de datos.

Al definir una tabla, se debe pensar en primer lugar en las filas que la forman y a continuación en las columnas. El motivo es que HTML procesa primero las filas y por eso las etiquetas `<tr>` aparecen antes que las etiquetas `<td>`.

<code><table></code>	Tabla
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<code>summary = "texto"</code> - Permite describir el contenido de la tabla (lo utilizan los buscadores y las personas discapacitadas)
Tipo de elemento	Bloque
Descripción	Se emplea para definir tablas de datos

<tr>	Fila de tabla
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir cada fila de las tablas de datos

<td>	Celda de tabla
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<p>abbr = "texto" - Permite describir el contenido de la celda (se emplea sobre todo con los navegadores de voz utilizados por personas discapacitadas)</p> <p>headers = "lista_de_id" - Indica las celdas que actúan como cabeceras para esta celda (los títulos de las columnas y filas). Se indica como una lista de valores del atributo "id" de celdas</p> <p>scope = "col, row, colgroup, rowgroup" - Indica las celdas para las que esta celda será su cabecera. Ej: scope="col" indica que esta celda es la cabecera de todas las demás celdas que están en la misma columna</p> <p>colspan = "numero" - Número de columnas que ocupa esta celda</p> <p>rowspan = "numero" - Número de filas que ocupa esta celda</p>
Tipo de elemento	Bloque
Descripción	Se emplea para definir cada una de las celdas que forman las filas de una tabla, es decir, las columnas de la tabla

De todos los atributos disponibles para las celdas, los más utilizados son **rowspan** y **colspan**, que se emplean para construir tablas complejas como las que se ven más

adelante. Entre los demás atributos, sólo se utiliza de forma habitual el atributo `scope`, sobre todo con las celdas de cabecera que se ven a continuación.

Normalmente, algunas de las celdas de la tabla se utilizan como cabecera de las demás celdas de la fila o de la columna. En este caso, HTML define la etiqueta `<th>` (del inglés "table header cell") para indicar que una celda es cabecera de otras celdas.

<th>	Celda cabecera de tabla
Atributos comunes	básicos, <code>id</code> y eventos
Atributos específicos	<p><code>abbr</code> = "texto" - Permite describir el contenido de la celda (se emplea sobre todo con los navegadores de voz utilizados por personas discapacitadas)</p> <p><code>headers</code> = "lista_de_id" - Indica las celdas que actúan como cabeceras para esta celda (los títulos de las columnas y filas). Se indica como una lista de ID de celdas</p> <p><code>scope</code> = "col, row, colgroup, rowgroup" - Indica las celdas para las que esta celda será su cabecera. Ej: <code>scope="col"</code> indica que esta celda es la cabecera de todas las demás celdas que están en la misma columna</p> <p><code>colspan</code> = "numero" - Número de columnas que ocupa esta celda</p> <p><code>rowspan</code> = "numero" - Número de filas que ocupa esta celda</p>
Tipo de elemento	Bloque
Descripción	Se emplea para definir las celdas que son cabecera de una fila o de una columna de la tabla

Los atributos de la etiqueta `<th>` son idénticos que los atributos definidos para la etiqueta `<td>`. En este caso, el atributo más utilizado es `scope`, que permite indicar si la celda es cabecera de la fila o de la columna (`<th scope="row">` y `<th scope="col">` respectivamente).

Por otra parte, HTML define la etiqueta `<caption>` para establecer la leyenda o título de una tabla. La etiqueta debe colocarse inmediatamente después de la etiqueta `<table>` y cada tabla sólo puede incluir una etiqueta `<caption>`.

<caption>	Leyenda o título de tabla
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Se emplea para definir la leyenda o título de una tabla

Ejercicio 12: Determinar el código HTML necesario para crear la tabla que se muestra en la siguiente imagen:

Nombre producto	Precio unitario	Unidades	Subtotal
Reproductor MP3 (80 GB)	192.02	1	192.02
Fundas de colores	2.50	5	12.50
Reproductor de radio & control remoto	12.99	1	12.99
TOTAL	-	7	207.51

Figura 2.44. Tabla sencilla con celdas de cabecera

Utilizar las celdas de cabecera donde sea necesario y añadir todos los atributos posibles.

Ejercicio 13: Determinar el código HTML necesario para crear la tabla que se muestra en la siguiente imagen. Utilizar las celdas de cabecera donde sea necesario y añadir todos los atributos posibles.



Imagen	Datos
	Portátil - 3 GHz - 4 GB RAM Comprar: 2.990 € 2.599 €
	Videocámara - Alta definición 1080p - 60 GB Comprar: 1.099 € 999 €
	Televisor - 46" - Full HD Comprar: 1.999 € 1.799 €
	Móvil - 3G - Wi-Fi - 8 GB Comprar: 399 € 349 €

Figura 2.45. Tabla con los resultados de una búsqueda

Las tablas complejas suelen disponer de una estructura irregular que junta varias columnas para formar una columna ancha o une varias filas para formar una fila más alta que las demás. Para fusionar filas o columnas, se utilizan los atributos `rowspan` y `colspan` respectivamente.

La siguiente imagen muestra una tabla compleja que ha fusionado dos columnas simples para formar una columna más ancha:

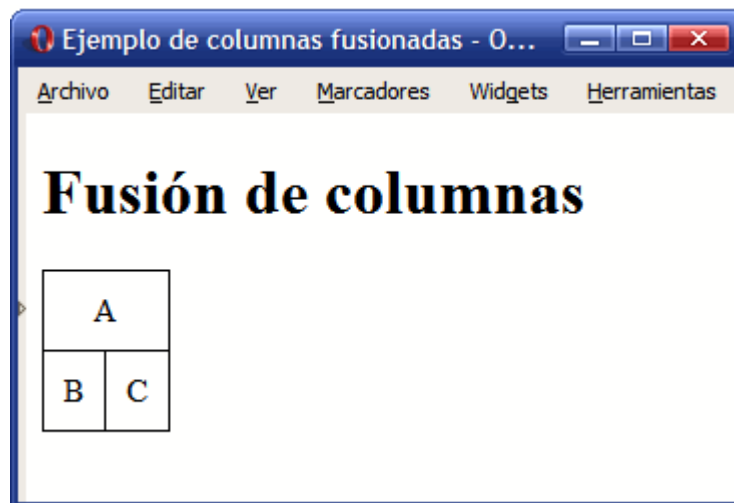


Figura 2.46. Ejemplo sencillo de fusión de columnas

Para obtener una tabla como la de la imagen anterior, se debe utilizar el siguiente código:

```
<table>
<tr>
  <td colspan="2">A</td>
</tr>

<tr>
  <td>B</td>
  <td>C</td>
</tr>
</table>
```

La primera fila de la tabla está formada sólo por una columna, mientras que la segunda fila está formada por dos columnas. En principio, podría pensarse en utilizar el siguiente código HTML para definir la tabla:

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

<tr>
  <td>B</td>
  <td>C</td>
</tr>
</table>
```

Sin embargo, si se utiliza el código anterior, el navegador visualiza de forma incorrecta la tabla, ya que las tablas en HTML deben disponer de una estructura regular. En otras palabras, todas las filas de una tabla HTML deben tener el mismo número de columnas. Por lo tanto, si se quieren mostrar menos columnas en una fila, se fusionan mediante el atributo `colspan`, que indica el número de columnas simples que va a ocupar una determinada celda.

En el ejemplo anterior, la celda de la primera fila debe ocupar el espacio de dos columnas simples, por lo que el código HTML debe ser `<td colspan="2">A</td>`.

De forma equivalente, si se quiere diseñar una tabla HTML que fusiona filas como la de la siguiente imagen:

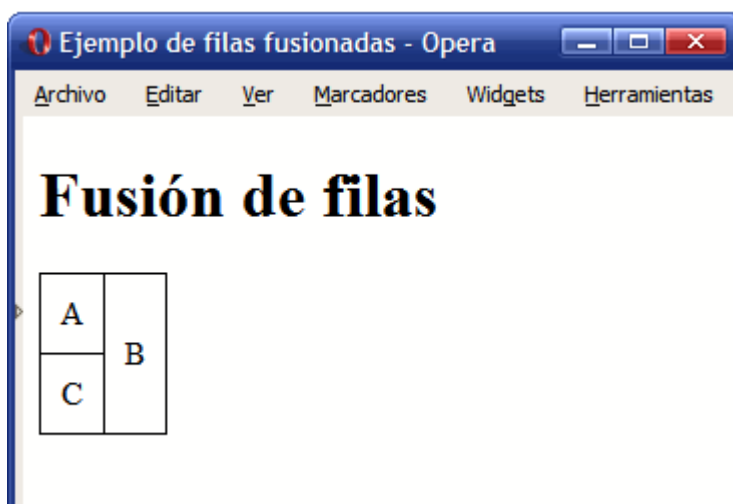


Figura 2.47. Ejemplo sencillo de fusión de filas

El código HTML que se debe utilizar para obtener la tabla de la imagen anterior es:

```
<table>
<tr>
  <td>A</td>
  <td rowspan="2">B</td>
</tr>

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

De forma análoga a la fusión de columnas del ejemplo anterior, la fusión de filas debe indicarse de forma especial. Como las tablas HTML tienen que ser regulares, todas las columnas deben tener el mismo número de filas. Así, si en el ejemplo anterior se utilizara el siguiente código:

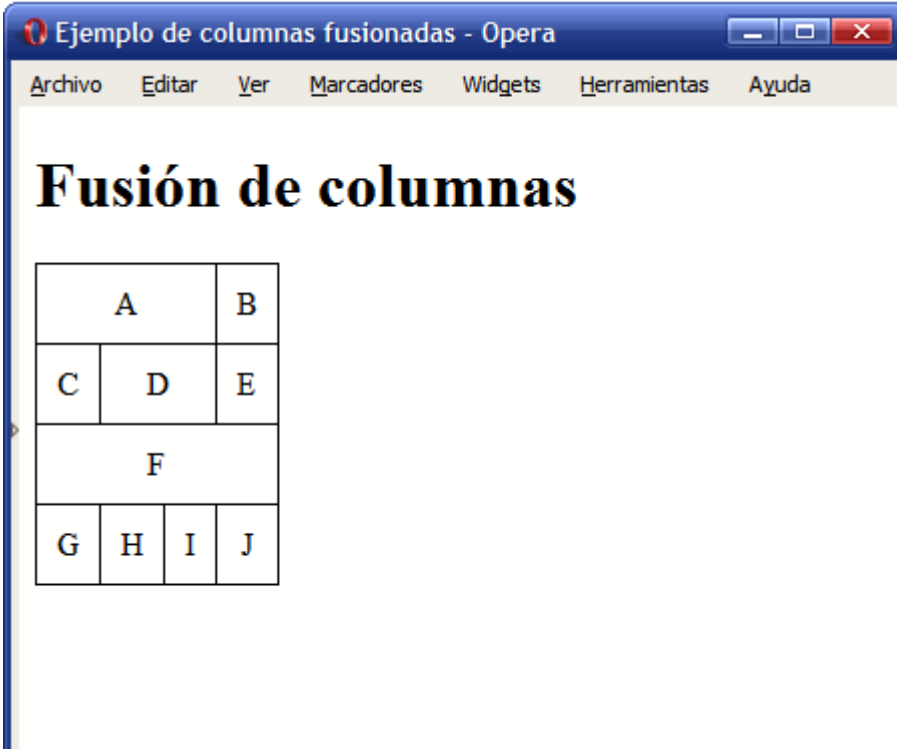

```
<table>
<tr>
  <td>A</td>
  <td>B</td>
</tr>

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

La tabla anterior no se visualizaría correctamente. Como la segunda columna de la tabla ocupa el espacio de las dos filas, el código HTML debe indicar claramente que esa celda va a ocupar dos filas, de manera que todas las columnas de la tabla cuenten con el mismo número de filas.

Utilizando los atributos `rowspan` y `colspan`, es posible diseñar tablas tan complejas como las que se muestran en los siguientes ejemplos.

Ejemplo de fusión de múltiples columnas:



A		B	
C	D	E	
F			
G	H	I	J

Figura 2.48. Ejemplo complejo de fusión de columnas

El código HTML necesario para fusionar las columnas de la tabla anterior se muestra a continuación:

```
<!DOCTYPE html>

<html>
<head><title>Ejemplo de columnas fusionadas</title></head>
<body>

<h1>Fusión de columnas</h1>

<table>
<tr>
  <td colspan="3">A</td>
  <td>B</td>
</tr>

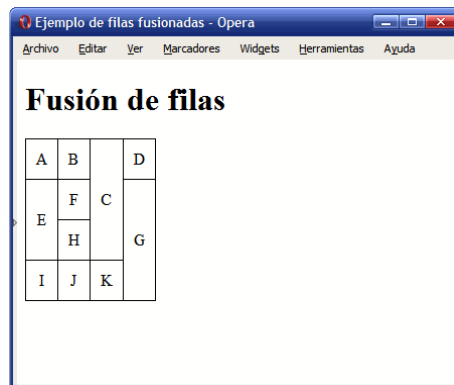
<tr>
  <td>C</td>
  <td colspan="2">D</td>
  <td>E</td>
</tr>

<tr>
  <td colspan="4">F</td>
</tr>

<tr>
  <td>G</td>
  <td>H</td>
  <td>I</td>
  <td>J</td>
</tr>
</table>

</body>
</html>
```

Ejemplo de fusión de múltiples filas:



A	B	C	D
E	F	C	G
H	G	G	
I	J	K	G

Figura 2.49. Ejemplo complejo de fusión de filas

El código HTML necesario para fusionar las filas de la tabla anterior se muestra a continuación:

```
<!DOCTYPE html>

<html>
<head><title>Ejemplo de filas fusionadas</title></head>
<body>

<h1>Fusión de filas</h1>

<table>
  <tr>
    <td>A</td>
    <td>B</td>
    <td rowspan="3">C</td>
    <td>D</td>
  </tr>
  <tr>
    <td rowspan="2">E</td>
    <td>F</td>
    <td rowspan="3">G</td>
  </tr>
  <tr>
    <td>H</td>
  </tr>
  <tr>
    <td>I</td>
    <td>J</td>
    <td>K</td>
  </tr>
</table>
</body>
</html>
```

Ejercicio 14: Determinar el código HTML necesario para crear la tabla que se muestra en la siguiente imagen:

Comparativa de reproductores MP3

Tabla comparativa de las características técnicas de los reproductores MP3

	MP3 mini			MP3 grande	
Capacidad de almacenamiento	4GB (1.000 canciones)	8GB (2.000 canciones)	16GB (4.000 canciones)	30GB (7.500 canciones)	80GB (20.000 canciones)
Colores					
Pantalla	LCD de 3 cm (diagonal) con retroiluminación			LCD de 6 cm (diagonal) con retroiluminación	
Tiempo de carga	Unas 3 horas			Unas 4 horas	
				Unas 2 horas para alcanzar el 80% de la capacidad	

Figura 2.50. Ejemplo de tabla con una estructura compleja

Emplear las etiquetas <table>, <tr>, <td>, <th>, <caption> y los atributos colspan, rowspan, abbr, scope, summary.

7.2. Tablas avanzadas

Algunas tablas complejas están formadas por más elementos que filas y celdas de datos. Así, es común que las tablas más avanzadas dispongan de una sección de cabecera, una sección de pie y varias secciones de datos. Además, también es posible agrupar varias columnas de forma lógica para poder aplicar estilos similares a un determinado grupo de columnas.

Un ejemplo clásico de tablas avanzadas es el de las tablas utilizadas en contabilidad, como por ejemplo la tabla que muestra el balance de una empresa:

CONSOLIDATED BALANCE SHEET	2008			
	March	June	September	December
Non-current assets	87.249	87.126	90.426	91.269
Intangible assets	21.810	21.145	20.986	20.758
Goodwill	17.914	19.660	21.828	21.739
Property, plant and equipment and Investment property	33.245	32.332	33.428	33.888
Long-term financial assets and other non-current assets	5.723	5.687	5.981	6.183
Deferred tax assets	8.557	8.303	8.202	8.702
Current assets	18.042	17.979	19.128	17.713
Inventories	1.154	1.134	1.052	1.012
Trade and other receivables	9.244	9.495	9.709	9.666
Current tax receivable	1.288	1.565	1.468	1.555
Short-term financial investments	1.877	1.803	1.788	1.679
Cash and cash equivalents	4.468	3.557	5.101	3.792
Non-current assets classified as held for sale	11	425	9	9
Total Assets = Total Equity and Liabilities	105.291	105.106	109.554	108.982
Equity	15.714	15.072	19.185	20.001
Equity attributable to equity holders of the parent	11.932	12.085	16.397	17.178
Minority interest	3.782	2.987	2.788	2.823
Non-current liabilities	54.053	66.406	63.908	62.644,0
Long-term financial debt	41.665	54.263	51.647	50.675
Deferred tax liabilities	4.868	4.617	4.727	4.700
Long-term provisions	6.466	6.507	6.545	6.287
Other long-term liabilities	1.054	1.020	988	982
Current liabilities	35.523	23.628	26.462	26.337,0
Short-term financial debt	19.507	7.466	8.975	8.382
Trade and other payables	8.792	8.259	8.782	8.533
Current tax payable	2.007	2.324	2.529	2.841
Short-term provisions and other liabilities	5.218	5.212	6.176	6.580
Liabilities associated with non-current assets classified as held for sale	0	367	0	0
Financial Data				
Net Financial Debt (1)	53.510	54.922	52.239	52.145

Figura 2.51. Ejemplo de tabla compleja correspondiente al balance de una empresa

Las partes que componen las tablas complejas se definen mediante las etiquetas `<thead>`, `<tbody>` y `<tfoot>`. La cabecera de la tabla se define con la etiqueta `<thead>`, el pie de la tabla se define mediante `<tfoot>` y cada sección de datos se define con una etiqueta `<tbody>`.

<code><thead><tbody><tfoot></code>	Cabecera de tabla Sección de una tabla Pie de tabla
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplean para agrupar varias filas en una cabecera (thead) un pie (tfoot) o una sección (tbody) de una tabla

Cada tabla puede contener solamente una cabecera y un pie, pero puede incluir un número ilimitado de secciones. Si se define una cabecera y/o un pie, las etiquetas `<thead>` y/o `<tfoot>` deben colocarse inmediatamente antes que cualquier etiqueta `<tbody>`.

La siguiente imagen muestra una tabla avanzada con cabecera, pie y una sección de datos:

AÑO	Expansión de ventas			
	Producto A	Producto B	Producto C	Producto D
N-3	-	-	-	-
N-2	3	5	8	4
N-1	4	4	7	3
N	5	7	6	2

AÑO	Producto A	Producto B	Producto C	Producto D
Expansión de ventas				

Figura 2.52. Ejemplo de tabla avanzada con cabecera, pie y secciones

El código HTML necesario para crear la tabla de la imagen anterior hace uso de las etiquetas `<thead>`, `<tbody>` y `<tfoot>`:

```
<!DOCTYPE html>
<html>
<head><title>Ejemplo de tabla avanzada</title></head>
<body>
<h3>Análisis de ventas</h3>
```

```
<table summary="Análisis de ventas anuales">
  <caption>Análisis de ventas anuales</caption>
  <thead>
    <tr>
      <th rowspan="2" scope="col">AÑO</th>
      <th colspan="4" scope="col">Expansión de ventas</th>
    </tr>
    <tr>
      <th scope="col">Producto A</th>
      <th scope="col">Producto B</th>
      <th scope="col">Producto C</th>
      <th scope="col">Producto D</th>
    </tr>
  </thead>

  <tfoot>
    <tr>
      <th rowspan="2" scope="col">AÑO</th>
      <th scope="col">Producto A</th>
      <th scope="col">Producto B</th>
      <th scope="col">Producto C</th>
      <th scope="col">Producto D</th>
    </tr>
    <tr>
      <th colspan="4" scope="col">Expansión de ventas</th>
    </tr>
  </tfoot>

  <tbody>
    <tr>
      <th scope="row">N-3</th><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td>
    </tr>
    <tr>
      <th scope="row">N-2</th><td>3</td><td>5</td><td>8</td><td>4</td>
    </tr>
    <tr>
      <th scope="row">N-1</th><td>4</td><td>4</td><td>7</td><td>3</td>
    </tr>
    <tr>
      <th scope="row">N</th><td>5</td><td>7</td><td>6</td><td>2</td>
    </tr>
  </tbody>
</table>
</body>
</html>
```

Aunque al principio resulta extraño, el elemento `<tfoot>` siempre se escribe antes que cualquier elemento `<tbody>` en el código HTML. De hecho, si la etiqueta `<tfoot>` aparece después de un elemento `<tbody>`, la página no se considera válida.

La etiqueta `<tbody>` permite realizar agrupaciones de filas, pero en ocasiones se necesitan agrupar columnas. Aunque su uso no es muy común, HTML define dos etiquetas similares para agrupar columnas: `<col>` y `<colgroup>`.

La etiqueta `<col>` se utiliza para asignar los mismos atributos a varias columnas de forma simultánea. De esta forma, la etiqueta `<col>` no agrupa columnas, sino que sólo asigna atributos comunes a varias columnas.

La siguiente imagen muestra una tabla que hace uso de la etiqueta `<col>`:



Análisis de ventas anuales				
AÑO	Producto A	Producto B	Producto C	Producto D
N-3	-	-	-	-
N-2	3	5	8	4
N-1	4	4	7	3
N	5	7	6	2

Figura 2.53. Ejemplo de tabla avanzada que usa la etiqueta `col`

El código HTML necesario para crear la tabla anterior se muestra a continuación:

```
<table summary="Análisis de ventas anuales">
  <caption>Análisis de ventas anuales</caption>

  <col style="width:10%;" />
  <col style="width:30%;" />

  <thead>
    <tr>
      <th scope="col">AÑO</th>
      <th scope="col">Producto A</th>
```



```

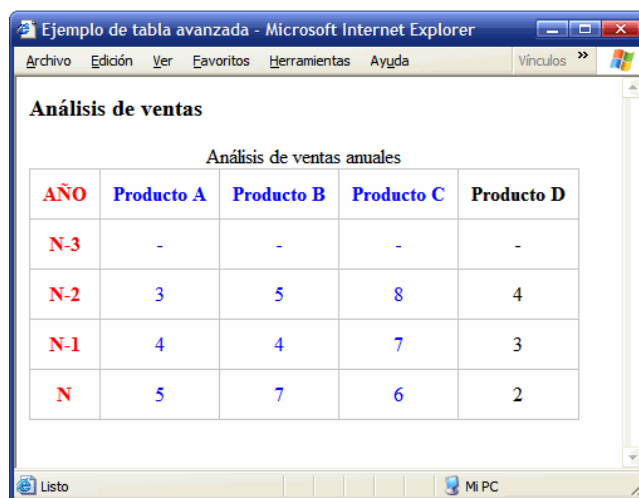
        <th scope="col">Producto B</th>
        <th scope="col">Producto C</th>
        <th scope="col">Producto D</th>
    </tr>
</thead>

<tbody>
    <tr>
        <th scope="row">N-3</th><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td>
    </tr>
    <tr>
        <th scope="row">N-2</th><td>3</td><td>5</td><td>8</td><td>4</td><td>4</td><td>4</td>
    </tr>
    <tr>
        <th scope="row">N-1</th><td>4</td><td>4</td><td>7</td><td>3</td><td>3</td><td>3</td>
    </tr>
    <tr>
        <th scope="row">N</th><td>5</td><td>7</td><td>6</td><td>2</td><td>2</td><td>2</td>
    </tr>
</tbody>
</table>

```

Por otra parte, la etiqueta `<colgroup>` se emplea para agrupar de forma estructural varias columnas de la tabla. La forma habitual de indicar el número de columnas que abarca la agrupación es utilizar el atributo `span`, que establece el número de columnas de cada agrupación.

La siguiente imagen muestra una tabla avanzada con una agrupación de columnas realizada con la etiqueta `<colgroup>`:



Análisis de ventas anuales				
AÑO	Producto A	Producto B	Producto C	Producto D
N-3	-	-	-	-
N-2	3	5	8	4
N-1	4	4	7	3
N	5	7	6	2

Figura 2.54. Ejemplo de tabla avanzada que usa la etiqueta `colgroup`

El código HTML necesario para crear la tabla anterior se muestra a continuación:

```
<table summary="Análisis de ventas anuales">
  <caption>Análisis de ventas anuales</caption>

  <colgroup span="1" style="color:red;" />
  <colgroup span="3" style="color:blue;" />

  <thead>
    <tr>
      <th scope="col">AÑO</th>
      <th scope="col">Producto A</th>
      <th scope="col">Producto B</th>
      <th scope="col">Producto C</th>
      <th scope="col">Producto D</th>
    </tr>
  </thead>

  <tbody>
    <tr>
      <th scope="row">N-3</th><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td>
    </tr>
    <tr>
      <th scope="row">N-2</th><td>3</td><td>5</td><td>8</td><td>4</td>
    </tr>
    <tr>
      <th scope="row">N-1</th><td>4</td><td>4</td><td>7</td><td>3</td>
    </tr>
    <tr>
      <th scope="row">N</th><td>5</td><td>7</td><td>6</td><td>2</td>
    </tr>
  </tbody>
</table>
```

El uso de las etiquetas `<col>` y `<colgroup>` no está muy extendido, debido a que la mayoría de navegadores no soportan muchas de sus funcionalidades.

8. FORMULARIOS

HTML es un lenguaje de marcado cuyo propósito principal consiste en estructurar los contenidos de los documentos y páginas web. Sin embargo, HTML también incluye elementos para crear aplicaciones web. El estándar HTML/XHTML permite crear formularios para que los usuarios interactúen con las aplicaciones web.

Los años transcurridos desde la publicación de los estándares de HTML y XHTML ha provocado que no estén disponibles todos los elementos utilizados por los formularios más avanzados y modernos. No obstante, HTML/XHTML incluye los suficientes elementos de formulario para crear desde los formularios sencillos que utilizan los buscadores hasta los formularios complejos de las aplicaciones más avanzadas.

8.1. Formularios básicos

Los formularios más sencillos se pueden crear utilizando solamente dos etiquetas: `<form>` y `<input>`. Si se considera el formulario que muestra la siguiente imagen:



Figura 2.55. Formulario sencillo definido con las etiquetas `form` e `input`

El código HTML necesario para definir el formulario anterior se muestra a continuación:

```
<!DOCTYPE html>
<html>
<head><title>Ejemplo de formulario sencillo</title></head>
<body>
<h3>Formulario muy sencillo</h3>
<form action="maneja_formulario.php" method="post">
  Escribe tu nombre:
  <input type="text" name="nombre" value="" />
  <br/>
  <input type="submit" value="Enviar" />
</form>
</body>
</html>
```

La etiqueta `<form>` encierra todos los contenidos del formulario (botones, cuadros de texto, listas desplegables) y la etiqueta `<input>` permite definir varios tipos diferentes de elementos (botones y cuadros de texto).

<form>	Formulario
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<p><code>action = "url"</code> - Indica la URL que se encarga de procesar los datos del formulario</p> <p><code>method = "POST o GET"</code> - Método HTTP empleado al enviar el formulario</p> <p><code>enctype = "application/x-www-form-urlencoded o multipart/form-data o text/plain"</code> - Tipo de codificación empleada al enviar el formulario al servidor (sólo se indica de forma explícita en los formularios que permiten adjuntar archivos)</p> <p><code>accept = "tipo_de_contenido"</code> - Lista separada por comas de todos los tipos de archivos aceptados por el servidor (sólo para los formularios que permiten adjuntar archivos). No soportado en HTML5</p> <p><code>accept-charset = "juego de caracteres"</code> - Lista separada por espacios de las codificaciones de caracteres que se usarán para el envío del formulario. Valores comunes: UTF-8 e ISO-8859-1</p> <p><code>autocomplete = "on o off"</code> - Especifica si un formulario tiene autocompletar a on o a off. Nuevo en HTML5</p> <p><code>novalidate = "novalidate"</code> - Es booleano. Cuando está presente especifica que los campos input del formulario no deberían ser validados cuando se envían. Nuevo en HTML5</p>
Tipo de elemento	Bloque
Descripción	Se emplea para insertar un formulario en la página

La mayoría de formularios utilizan sólo los atributos `action` y `method`. El atributo `action` indica la URL de la aplicación del servidor que se encarga de procesar los datos introducidos por los usuarios. Esta aplicación también se encarga de generar la respuesta que muestra el navegador.

El atributo `method` establece la forma en la que se envían los datos del formulario al servidor. Este atributo hace referencia al método HTTP, por lo que no es algo propio de HTML. Los dos valores que se utilizan en los formularios son `GET` y `POST`. De esta forma, casi todos los formularios incluyen el atributo `method="get"` o el atributo `method="post"`.

Al margen de otras diferencias técnicas, el método `POST` permite el envío de mucha más información que el método `GET`. En general, el método `GET` admite como máximo el envío de unos 500 bytes de información. La otra gran limitación del método `GET` es que no permite el envío de archivos adjuntos con el formulario. Además, los datos enviados mediante `GET` se ven en la barra de direcciones del navegador (se añaden al final de la URL de la página), mientras que los datos enviados mediante `POST` no se pueden ver tan fácilmente.

Existe una regla general que dice que el método `GET` se debe utilizar en los formularios que no modifican la información (por ejemplo en un formulario de búsqueda). Por su parte, el método `POST` se debería utilizar cuando el formulario modifica la información original (insertar, modificar o borrar alguna información).

El ejemplo más común de formulario con método `GET` es el de los buscadores. Si realizas una búsqueda con un buscador, verás que las palabras que has introducido en tu búsqueda aparecen como parte de la URL de la página de resultados.

El atributo `enctype` de la etiqueta `<form>`, como se explica más adelante, es imprescindible en los formularios que permiten adjuntar archivos.

Respecto a los atributos nuevos en HTML5. El atributo `autocomplete` especifica si un formulario tiene autocompletar a `on` o a `off`. Cuando `autocomplete` está a `on`, el navegador automáticamente completa los valores basándose en los valores que el usuario haya introducido antes. El valor por defecto es `on`. Si el valor es `off`, el usuario debe introducir un valor en cada campo en cada uso. El navegador no completa automáticamente las entradas. Es posible tener autocompletar a `on` para el formulario y a `off` para campos input específicos o viceversa.

El atributo `novalidate` es booleano. Cuando está presente especifica que los campos input del formulario no deberían ser validados cuando se envían. Su valor se puede establecer de las siguientes formas: `<form novalidate>`, `<form novalidate="novalidate">` y `<form novalidate="">`

8.2. Elementos de formulario

Los elementos de formulario como botones y cuadros de texto también se denominan "campos de formulario" y "controles de formulario". La mayoría de controles se crean con la etiqueta `<input>`, por lo que su definición formal y su lista de atributos es muy extensa:

<input>	Control de un formulario
Atributos comunes	básicos, i18n, eventos y foco
Atributos específicos	<p><code>type = "text password checkbox radio submit reset file hidden image button color date datetime datetime-local month week time email number range search tel url "</code> - Indica el tipo de control que se incluye en el formulario</p> <p><code>name = "texto"</code> - Asigna un nombre al control (es imprescindible para que el servidor pueda procesar el formulario)</p> <p><code>value = "texto"</code> - Valor inicial del control</p> <p><code>size = "unidad_de_medida"</code> – Especifica el ancho, en caracteres, del control</p> <p><code>maxlength = "numero"</code> - Máximo número de caracteres para los controles.</p> <p><code>checked = "checked"</code> - Para los controles checkbox y radiobutton permite indicar qué opción aparece preseleccionada</p> <p><code>disabled = "disabled"</code> - El control aparece deshabilitado y su valor no se envía al servidor junto con el resto de datos</p> <p><code>readonly = "readonly"</code> - El contenido del control no se puede modificar</p> <p><code>src = "url"</code> - Para el control que permite crear botones con imágenes, indica la URL de la imagen que se emplea como botón de formulario</p> <p><code>alt = "texto"</code> – Texto alternativo para imágenes. (Sólo <code>image</code>)</p>
Tipo de elemento	En línea y etiqueta vacía
Descripción	Se emplean para insertar un control en un formulario

A continuación se muestran ejemplos para los veintitres controles que se pueden crear con la etiqueta `<input>`. Los trece últimos han sido añadidos en HTML5.

8.2.1. Cuadro de texto

Se trata del elemento más utilizado en los formularios. Es el tipo por defecto. Define un cuadro de texto de una línea (por defecto de ancho 20 caracteres). En el caso más sencillo, se muestra un cuadro de texto vacío en el que el usuario puede escribir cualquier texto:

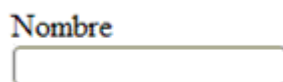


Figura 2.56. Ejemplo de etiqueta input (type="text")

A continuación se muestra el código HTML correspondiente al ejemplo anterior:

```
Nombre <br/>  
<input type="text" name="nombre" value="" />
```

El atributo `type` diferencia a cada uno de los veintitres controles que se pueden crear con la etiqueta `<input>`. Para los cuadros de texto, su valor es `text`. El atributo `name` es el más importante en los campos del formulario. De hecho, si un campo no incluye el atributo `name`, sus datos no se envían al servidor. El valor que se indica en el atributo `name` es el nombre que utiliza la aplicación del servidor para obtener el valor de este campo de formulario.

Cuando el usuario pulsa el botón de envío del formulario, el navegador envía los datos a una aplicación del servidor para que procese la información y genere una respuesta adecuada. En el servidor, la aplicación que procesa los datos debe obtener en primer lugar toda la información introducida por el usuario. Para ello, utiliza el valor del atributo `name` para obtener los datos de cada control del formulario.

Como el valor del atributo `name` se utiliza en aplicaciones programadas, es esencial ponerse de acuerdo con el programador de la aplicación, no se debe modificar su valor sin modificar la aplicación y no se deben utilizar caracteres problemáticos en programación (espacios en blanco, acentos y caracteres como ñ o ç).

El atributo `value` se emplea para establecer el valor inicial del cuadro de texto. Si se crea un formulario para insertar datos, los cuadros de texto deberían estar vacíos. Por lo tanto, o no se añade el atributo `value` o se incluye con un valor vacío `value=""`. Si por el contrario se crea un formulario para modificar datos, lo lógico es que se muestren inicialmente los datos guardados en el sistema. En este caso, el atributo `value` incluirá el valor que se desea mostrar: `<input type="text" name="nombre" value="Juan Pérez" />`

Si no se especifica un tamaño, el navegador muestra el cuadro de texto con un tamaño predeterminado. El atributo `size` permite establecer el tamaño, en caracteres, con el que se muestra el cuadro de texto. Su uso es imprescindible en muchos formularios, en los que algunos campos como la dirección deben mostrar más caracteres de lo normal (`<input size="100" ...`) y otros campos como el código postal deben mostrar menos caracteres de lo normal (`<input size="5"...`).

Además de controlar el tamaño con el que se muestra un cuadro de texto, también se puede limitar el tamaño del texto introducido. El atributo `maxlength` permite establecer el máximo número de caracteres que el usuario puede introducir en un cuadro de

texto. Su uso es imprescindible para campos como el código postal, el número de la Seguridad Social y cualquier otro dato con formato predefinido y limitado.

Por último, el atributo `readonly` permite que el usuario pueda ver los contenidos del cuadro de texto pero no pueda modificarlos y el atributo `disabled` deshabilita un cuadro de texto de forma que el usuario no pueda modificarlo y además, el navegador no envía sus datos al servidor.

8.2.2. Cuadro de contraseña

La única diferencia entre este control y el cuadro de texto normal es que el texto que el usuario escribe en un cuadro de contraseña no se ve en la pantalla. En su lugar, los navegadores ocultan el texto utilizando asteriscos o círculos, por lo que es ideal para escribir contraseñas y otros datos sensibles.

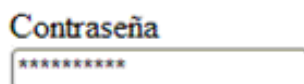


Figura 2.57. Ejemplo de etiqueta `input` (`type="password"`)

```
Contraseña <br/>  
<input type="password" name="contrasena" value="" />
```

Cambiando el valor del atributo `type` por `password` se transforma el cuadro de texto normal en un cuadro de contraseña. Todos los demás atributos se utilizan de la misma forma y tienen el mismo significado.

8.2.3. Checkbox

Los checkbox o "casillas de verificación" son controles de formulario que permiten al usuario seleccionar y deseleccionar opciones individualmente. Aunque en ocasiones se muestran varios checkbox juntos, cada uno de ellos es completamente independiente del resto. Por este motivo, se utilizan cuando el usuario puede activar y desactivar varias opciones relacionadas pero no excluyentes.

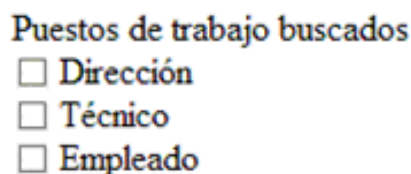


Figura 2.58. Ejemplo de etiqueta `input` (`type="checkbox"`)


```
Puestos de trabajo buscados <br/>
<input name="puesto_directivo" type="checkbox"
value="direccion"/> Dirección
<input name="puesto_tecnico" type="checkbox" value="tecnico"/>
Técnico
<input name="puesto_empleado" type="checkbox" value="empleado"/>
Empleado
```

El valor del atributo `type` para estos controles de formulario es `checkbox`. Como se muestra en el ejemplo anterior, el texto que se encuentra al lado de cada checkbox no se puede establecer mediante ningún atributo, por lo que es necesario añadirlo manualmente fuera del control del formulario. Si no se añade un texto al lado de la etiqueta `<input />` del checkbox, el usuario sólo ve un pequeño cuadrado sin ninguna información relativa a la finalidad de ese checkbox.

El valor del atributo `value`, junto con el valor del atributo `name`, es la información que llega al servidor cuando el usuario envía el formulario.

Si se quiere mostrar un checkbox seleccionado por defecto, se utiliza el atributo `checked`. Si el valor del atributo es `checked`, el checkbox se muestra seleccionado. En cualquier otro caso, el checkbox permanece sin seleccionar. Aunque resulta redundante que el nombre y el valor del atributo sean idénticos, es obligatorio indicarlo de esta forma porque los atributos en XHTML no pueden tener valores vacíos:

```
<input type="checkbox" checked="checked" ... /> Checkbox
seleccionado por defecto
```

8.2.4. Radiobutton

Los controles de tipo `radiobutton` son similares a los controles de tipo `checkbox`, pero presentan una diferencia muy importante: son mutuamente excluyentes. Los `radiobutton` se utilizan cuando el usuario solamente puede escoger una opción entre las distintas opciones relacionadas que se le presentan. Cada vez que se selecciona una opción, automáticamente se deselecta la otra opción que estaba seleccionada.

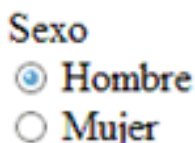


Figura 2.59. Ejemplo de etiqueta `input` (`type="radio"`)

```
Sexo <br/>
<input type="radio" name="sexo" value="hombre" checked="checked"
/> Hombre
<input type="radio" name="sexo" value="mujer" /> Mujer
```

El valor del atributo `type` para estos controles de formulario es `radio`. El atributo `name` se emplea para indicar los radiobutton que están relacionados. Por lo tanto, cuando varios radiobutton tienen el mismo valor en su atributo `name`, el navegador sabe que están relacionados y puede deseleccionar una opción del grupo de radiobutton cuando se seleccione otra opción.

8.2.5. Botón de envío de formulario

La mayoría de formularios dispone de un botón para enviar al servidor los datos introducidos por el usuario:

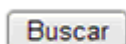


Figura 2.60. Ejemplo de etiqueta input (type="submit")

```
<input type="submit" name="buscar" value="Buscar" />
```

El valor del atributo `type` para este control de formulario es `submit`. El navegador se encarga de enviar automáticamente los datos cuando el usuario pincha sobre este tipo de botón. El valor del atributo `value` es el texto que muestra el botón. Si no se establece el atributo `value`, el navegador muestra el texto predefinido `Enviar consulta`.

8.2.6. Botón de reseteo del formulario

Aunque su uso era muy popular hace unos años, la mayoría de formularios modernos ya no utilizan este tipo de botón. Se trata de un botón especial que borra todos los datos introducidos por el usuario y devuelve el formulario a su estado original:

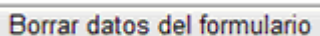


Figura 2.61. Ejemplo de etiqueta input (type="reset")

```
<input type="reset" name="limpiar" value="Borrar datos del
formulario" />
```

El valor del atributo `type` para este control de formulario es `reset`. Cuando el usuario pulsa este botón, el navegador borra toda la información introducida y muestra el formulario en su estado original. Si el formulario no contenía originalmente ningún valor, el botón de `reset` lo vuelve a mostrar vacío. Si el formulario contenía información, el botón `reset` vuelve a mostrar la misma información original.

Como es habitual en los botones de formulario, el atributo `value` permite establecer el texto que muestra el botón. Si no se utiliza este atributo, el navegador muestra el texto predefinido del botón, que en este caso es `Restablecer`.

8.2.7. Ficheros adjuntos

Los formularios también permiten adjuntar archivos para subirlos al servidor. Aunque desde el punto de vista de HTML y del navegador no existe ninguna limitación sobre el número, tipo o tamaño total de los archivos que se pueden adjuntar, todos los servidores añaden restricciones por motivos de seguridad.

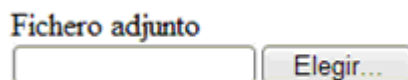


Figura 2.62. Ejemplo de etiqueta `input (type="file")`

```
Fichero adjunto  
<input type="file" name="adjunto" />
```

El valor del atributo `type` para este control de formulario es `file`. El navegador se encarga de mostrar un cuadro de texto donde aparece el nombre del archivo seleccionado y un botón que permite navegar por los directorios y archivos del ordenador del usuario.

Si se incluye un control para adjuntar archivos, es obligatorio añadir el atributo `enctype` en la etiqueta `<form>` del formulario. El valor del atributo `enctype` debe ser `multipart/form-data`, por lo que la etiqueta `<form>` de los formularios que permiten adjuntar archivos siempre es:

```
<form action="..." method="post" enctype="multipart/form-data">  
  ...  
</form>
```

8.2.8. Campos ocultos

Los campos ocultos se emplean para añadir información oculta en el formulario:

**Los campos ocultos
no se ven en pantalla*

Figura 2.63. Ejemplo de etiqueta input (type="hidden")

```
<input type="hidden" name="url_previa"  
value="/articulo/primer.html" />
```

El valor del atributo `type` para este control de formulario es `hidden`. Los campos ocultos no se muestran por pantalla, de forma que el usuario desconoce que el formulario los incluye. Normalmente los campos ocultos se utilizan para incluir información que necesita el servidor pero que no es necesario o no es posible que la establezca el usuario.

8.2.9. Botón de imagen

El aspecto de los botones de formulario se puede personalizar por completo, ya que incluso es posible utilizar una imagen como botón:



Figura 2.64. Ejemplo de etiqueta input (type="image")

```
<input type="image" name="enviar" src="accept.png" alt="Enviar"  
/>
```

El valor del atributo `type` para este control de formulario es `image`. El atributo `src` indica la URL de la imagen que debe mostrar el navegador en lugar del botón normal.

Su principal ventaja es que permite personalizar por completo la estética de los botones y mostrarlos con un aspecto homogéneo en todos los navegadores. El principal inconveniente es que ralentiza la carga del formulario y que si se quiere modificar su aspecto, es necesario crear una nueva imagen.

8.2.10. Botón

Algunos formularios complejos necesitan botones más avanzados que los de enviar datos (`type="submit"`) y resetear el formulario (`type="reset"`). Por ese motivo, el estándar HTML/XHTML define un botón de tipo genérico:

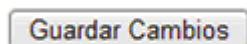


Figura 2.65. Ejemplo de etiqueta input (`type="button"`)

```
<input type="button" name="guardar" value="Guardar Cambios"
onclick="msg()" />
```

El valor del atributo `type` para este control de formulario es `button`. Si pruebas a pulsar un botón de este tipo, verás que el navegador no hace nada: no envía los datos al servidor y no borra los datos introducidos. Este tipo de botones sólo son útiles si se utilizan junto con el lenguaje de programación JavaScript. Si la página incluye código JavaScript, los botones de este tipo se pueden programar para que realicen cualquier tarea compleja cuando se pulsa sobre ellos.

8.2.11. Color

Se trata de un nuevo valor para el atributo `type` para los formularios definido en HTML5. Los visitantes pueden seleccionar un color en el sistema de selección de colores de su sistema operativo. El aspecto de este control en *Chrome* es el siguiente:

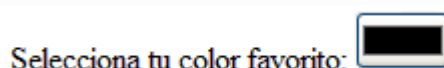


Figura 2.66. Ejemplo de etiqueta input (`type="color"`)

```
Selecciona tu color favorito: <input type="color" name="colorfav"/>
```

8.2.12. Campos para las fechas y las horas

El campo de tipo `date` permite insertar una fecha (año, mes y día). Pero en lugar de dejar que sean los propios usuarios quienes inserten las fechas, con los riesgos de error que eso conlleva, los navegadores propondrán de forma nativa una interfaz de selección de fechas que sea fácil de usar.

- `type="date"`: Define un control para la fecha solamente:

Escoja una fecha de entrega:

Figura 2.67. Ejemplo de etiqueta input (`type="date"`)

Escoja una fecha de entrega: `<input type="date" name="fechaentrega"/>`

- `type="datetime"`: Define un control para el día y hora con la zona horaria:

Fecha de entrega (fecha y hora):

Figura 2.68. Ejemplo de etiqueta input (`type="datetime"`)

Fecha de entrega (fecha y hora): `<input type="datetime" name="fechahoraentrega"/>`

- `type="datetime-local"` Define un control para el día y hora sin la zona horaria:

Fecha de entrega (fecha and hora):

Figura 2.69. Ejemplo de etiqueta input (`type="datetime-local"`)

Fecha de entrega (fecha y hora): `<input type="datetime-local" name="fechahoraentrega"/>`

- `type="month"` Define un control para un mes y un año (sin zona horaria):

Fecha de aniversario (mes y año):

Figura 2.70. Ejemplo de etiqueta input (`type="month"`)

Fecha de aniversario (mes y año): `<input type="month" name="fechaaniv"/>`

- `type="week"` Define un control para una semana y año (sin zona horaria):

Selecciona una semana:

Figura 2.71. Ejemplo de etiqueta input (`type="week"`)

Selecciona una semana: `<input type="week" name="semana"/>`

- `type="time"` Define un control para una hora (sin zona horaria):

Selecciona una hora:

Figura 2.71. Ejemplo de etiqueta input (`type="time"`)

```
Selecciona una hora: <input type="time" name="hora"/>
```

Nota: Es conveniente probar los ejemplos con un input de tipo submit.

```
<input type="submit" name="enviar" value="Enviar" />
```

8.2.13. Direcciones de e-mail

El nuevo campo de introducción de datos de tipo `email` permite especificar que el contenido insertado deberá ser una dirección de e-mail, es decir, que deberá incluir el carácter @. A continuación serán los navegadores quienes comprobarán de forma nativa si la arroba ha sido insertada o no.

Otra ventaja innegable, cuando se use con un smartphone que reconozca este tipo de campo: el teclado se adaptará automáticamente para la inserción de la dirección de e-mail.

En resumen, el tipo `email` define un campo para una dirección e-mail, que será automáticamente validada cuando se envíe. Veamos un ejemplo con Chrome:

E-mail:

Figura 2.72. Ejemplo de etiqueta input (`type="email"`)

```
E-mail: <input type="email" name="emailusuario"/><br/>  
<input type="submit" name="enviar" value="Enviar" />
```

8.2.14. Valores numéricos

Con el campo de inserción de datos de tipo `number` solamente se pueden introducir valores numéricos. La validación de este tipo de campo se deja una vez más a la libre elección de los navegadores.

Este tipo de campo admite varios atributos:

- **min**: el valor mínimo aceptado.
- **max**: el valor máximo aceptado.
- **step**: el valor del incremento que deberá aplicarse en la interfaz.
- **value**: el valor que se haya facilitado.

Veamos un ejemplo de sintaxis en el que podemos elegir un valor numérico, de 18 como mínimo, hasta un máximo de 100, mediante intervalo de 2 en 2 y con un valor inicial de 48:

Indique su edad (entre 18 y 100):

Figura 2.73. Ejemplo de etiqueta input (type="number")

```
Indique su edad (entre 18 y 100): <input type="number" name="edad"
min="18" max="100" step="2" value="48"/>
<input type="submit" name="enviar" value="Enviar" />
```

8.2.15. Barras de selección con cursor

El campo de introducción de datos **range**, que en realidad no permite introducir datos, mostrará una barra de selección con un cursor en la que podremos elegir un valor. También se pueden establecer restricciones sobre qué números se aceptan.

Este tipo de campo admite varios atributos:

- **min**: el valor mínimo aceptado.
- **max**: el valor máximo aceptado.
- **step**: el valor del incremento que deberá aplicarse en la interfaz.
- **value**: el valor que se haya facilitado.

Así se mostrará la barra de selección en Chrome:

Puntos:

Figura 2.74. Ejemplo de etiqueta input (type="range")

```
Puntos: <input type="range" name="puntos" min="1" max="10"
value="5"/>
<input type="submit" name="enviar" value="Enviar" />
```

8.2.16. Campos de búsqueda

El tipo **search** permite insertar un campo de introducción de datos específico para las búsquedas. Su visualización dependerá de los parámetros de la interfaz del navegador que esté utilizando el visitante.

Define un campo de búsqueda (como un sitio de búsqueda, o búsqueda de Google).

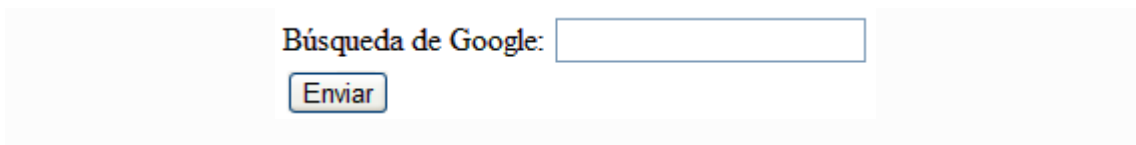


Figura 2.75. Ejemplo de etiqueta input (type="search")

```
Búsqueda de Google: <input type="search" name="busquedaGoogle"/>
<br/>
<input type="submit" name="enviar" value="Enviar" />
```

8.2.16. Números de teléfono

El campo de inserción de datos de tipo `tel` ha sido creado para recuperar números de teléfono.

No se aplicará ninguna restricción, ya que los números de teléfono son muy diferentes de un país a otro, e incluso pueden contener caracteres que no sean numéricos. Una vez más los smartphones podrán adaptar el teclado.

Por tanto, el tipo `tel` define un campo para introducir un número de telefono:

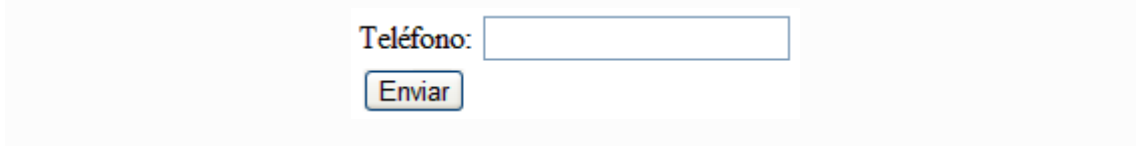


Figura 2.76. Ejemplo de etiqueta input (type="tel")

```
Teléfono: <input type="tel" name="telefono"/> <br/>
<input type="submit" name="enviar" value="Enviar" />
```

8.2.17. Campo para las URL

El campo para la introducción de datos de tipo `url` permite recuperar una URL. La comprobación de dicha URL para ver si es válida dependerá totalmente del navegador.

La última versión del navegador Opera añade automáticamente "http" cuando se introduce una URL que comience por "www". Una vez más, los smartphones podrán adaptar el teclado para facilitar la inserción de la URL.

Por tanto, el tipo `url` define un campo para introducir una URL.



Figura 2.77. Ejemplo de etiqueta input (type="url")

```
Tu sitio web: <input type="url" name="misitoweb"/> <br/>
<input type="submit" name="enviar" value="Enviar"/>
```

8.2.18. Campo de introducción de datos con sugerencias. Etiqueta <datalist>

La etiqueta `datalist` permite disponer de un campo de inserción libre y, al mismo tiempo de las funcionalidades de un menú desplegable. De este modo el usuario podrá insertar un valor personalizado o bien elegir un valor entre las sugerencias propuestas.

La elemento `datalist` utiliza los elementos habituales `option` para las sugerencias. Tendremos que usar simultáneamente un campo de tipo `text` y un elemento `datalist`. La relación se establece con el valor del atributo `list` de `input` y el `id` de `datalist`.

Veamos un ejemplo de aplicación:



Figura 2.77. Ejemplo de etiqueta `datalist`

```
<input list="navegadores" name="navegador">
<datalist id="navegadores">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
<input type="submit" name="enviar" value="Enviar" />
```

Ejercicio 15: Determinar el código HTML necesario para crear el formulario que se muestra en la siguiente imagen:

The screenshot shows a web browser window titled 'Opera' with a single tab 'Rellena tu CV'. The address bar shows 'localhost/F:/Curs'. The form itself is titled 'Rellena tu CV' and contains the following elements:

- Nombre:** A single-line text input field.
- Apellidos:** A single-line text input field.
- Fecha de nacimiento:** A date picker control.
- Indique su edad (entre 18 y 100):** A spin box with the value '18' displayed.
- Sexo:** Two radio buttons labeled 'Hombre' (selected) and 'Mujer'.
- Incluir mi foto:** A text input field followed by a button labeled 'Elegir...'. The text 'Incluir mi foto' is partially obscured by the input field.
- E-mail:** A single-line text input field.
- Teléfono:** A single-line text input field.
- Tu sitio web:** A single-line text input field.
- Valora del 1 al 10 cuánto te gusta esta página web:** A range slider control.
- Contraseña:** A single-line text input field.
- DNI:** A single-line text input field.
- Suscribirse al boletín de novedades:** A checked checkbox.
- Buttons:** Two buttons at the bottom: 'Guardar cambios' and 'Borrar los datos introducidos'.

Figura 2.78. Formulario con controles de varios tipos

1. Elegir el método más adecuado para el formulario (GET o POST) y cualquier otro atributo necesario.
2. La aplicación que se encarga de procesar el formulario se encuentra en la raíz del servidor, carpeta "php" y archivo "insertar_cv.php".

3. El nombre puede tener 30 caracteres como máximo, los apellidos 80 caracteres y la contraseña 10 caracteres como máximo.
4. Asignar los atributos adecuados al campo del DNI.
5. Por defecto, debe estar marcada la casilla de suscripción al boletín de novedades.

8.3. Formularios avanzados

Utilizando solamente las etiquetas `<form>` y `<input>` es posible diseñar la mayoría de formularios de las aplicaciones web. No obstante, HTML define algunos elementos adicionales para mejorar la estructura de los formularios creados.

La siguiente imagen muestra un formulario que agrupa sus elementos y añade etiquetas a cada campo para mejorar su estructura:

Ejemplo de etiqueta fieldset y legend - Opera

Archivo Editar Ver Marcadores Widgets Herramientas Ayuda

Formulario estructurado

Datos personales

Nombre

Apellidos

DNI

Datos de conexión

Nombre de usuario

Contraseña

Repita la contraseña

Figura 2.79. Ejemplo de uso de las etiquetas `fieldset` y `legend`

La etiqueta `<fieldset>` agrupa campos del formulario y la etiqueta `<legend>` asigna un nombre a cada grupo.

<fieldset>	Agrupación de campos
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para agrupar de forma lógica varios campos de un formulario

<legend>	Título o leyenda de un fieldset
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Se emplea para definir el título o leyenda de un conjunto de campos de formulario agrupados con la etiqueta fieldset

A continuación se muestra el código HTML del formulario correspondiente a la imagen anterior y que hace uso de `<fieldset>` y `<legend>` para agrupar los campos del formulario:

```
<!DOCTYPE html>
<html>
<head><title>Ejemplo de formulario avanzado 1</title></head>
<body>

<form action="manejo_formulario.php" method="post">
  <fieldset>
    <legend>Datos personales</legend>
    Nombre <br/>
    <input type="text" name="nombre" value="" />
    <br/>
    Apellidos <br/>
```

```
<input type="text" name="apellidos" value="" />
<br/>
DNI <br/>
<input type="text" name="dni" value="" size="10"
maxlength="9" />
</fieldset>

<fieldset>
  <legend>Datos de conexión</legend>
  Nombre de usuario<br/>
  <input type="text" name="nombre" value="" maxlength="10" />
  <br/>
  Contraseña<br/>
  <input type="password" name="password" value=""
maxlength="10" />
  <br/>
  Repite la contraseña<br/>
  <input type="password" name="password2" value=""
maxlength="10" />
</fieldset>
</form>
</body>
</html>
```

La etiqueta `<fieldset>` agrupa todos los controles de formulario a los que encierra. El navegador muestra por defecto un borde resaltado para cada agrupación. La etiqueta `<legend>` se incluye dentro de cada etiqueta `<fieldset>` y establece el título que muestra el navegador para cada agrupación de elementos.

Por otra parte, todos los controles de formulario salvo los botones presentan una carencia muy importante: no disponen de la opción de establecer el título o texto que se muestra junto al control. En el código HTML del ejemplo anterior, el nombre de cada campo se incluye en forma de texto normal, sin ninguna relación con el campo al que hace referencia.

Afortunadamente, el lenguaje HTML incluye una etiqueta denominada `<label>` y que se utiliza para establecer el título de cada campo del formulario. Su definición formal es la siguiente:

<code><label></code>	Título o leyenda de un campo de formulario
----------------------------	--------------------------------------------

<label>	Título o leyenda de un campo de formulario
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<code>for = "id_de_elemento"</code> - Indica el ID del campo del formulario para el que este elemento es su título Otros: <code>accesskey</code> , <code>onfocus</code> y <code>onblur</code>
Tipo de elemento	En línea
Descripción	Se emplea para definir el título o leyenda de los campos definidos en un formulario

El único atributo que suele utilizarse con la etiqueta `<label>` es `for`, que indica el identificador (atributo `id`) del campo de formulario para el que esta etiqueta hace de título.

En el anterior ejemplo, el nombre de los campos de formulario se incluía mediante un texto normal:

```
Nombre <br/>
<input type="text" name="nombre" value="" />

Apellidos <br/>
<input type="text" name="apellidos" value="" />

DNI <br/>
<input type="text" name="dni" value="" size="10" maxlength="9" />
```

Utilizando la etiqueta `<label>`, cada campo de formulario puede disponer de su propio título:

```
<label for="nombre">Nombre</label> <br/>
<input type="text" id="nombre" name="nombre" value="" />

<label for="apellidos">Apellidos</label> <br/>
<input type="text" id="apellidos" name="apellidos" value="" />

<label for="dni">DNI</label> <br/>
<input type="text" id="dni" name="dni" value="" size="10"
maxlength="9" />
```

La principal ventaja de utilizar `<label>` es que el código HTML está mejor estructurado y se mejora su accesibilidad. Además, al pinchar sobre el texto del `<label>`, el puntero

del ratón se posiciona automáticamente para poder escribir sobre el campo de formulario asociado. Este comportamiento es especialmente útil para los campos de tipo `radiobutton` y `checkbox`.

8.4. Otros elementos de formulario

La etiqueta `<input>` permite crear veintitrés tipos diferentes de controles de formulario. Sin embargo, algunas aplicaciones web utilizan otros elementos de formulario que no se pueden crear con `<input>`. Las listas desplegadas y las áreas de texto disponen de sus propias etiquetas (`<select>` y `<textarea>` respectivamente).

Las áreas de texto son útiles cuando se debe introducir una gran cantidad de texto, ya que es mucho más cómodo de introducir que en un campo de texto normal:

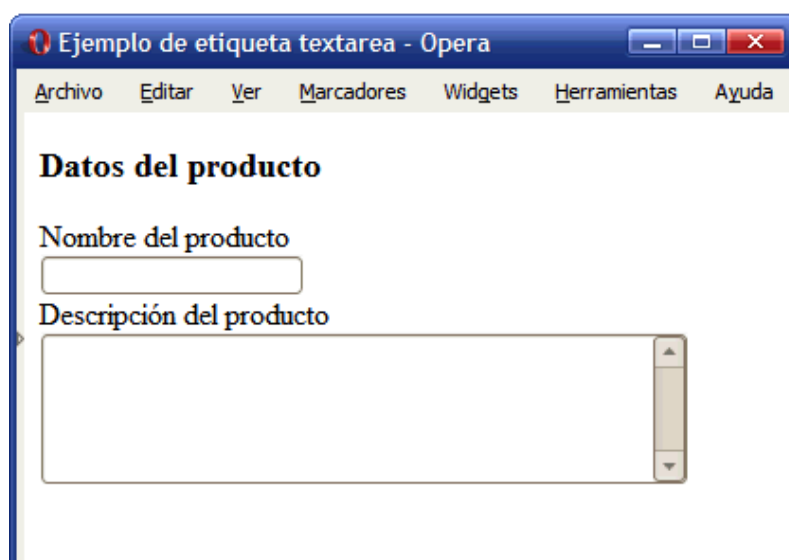


Figura 2.80. Ejemplo de uso de la etiqueta `textarea`

El código HTML del ejemplo anterior se muestra a continuación:

```
<form action="insertar_producto.php" method="post">

<label for="nombre">Nombre del producto</label> <br/>
<input type="text" id="nombre" name="nombre" value="" />

<label for="descripcion">Descripción del producto</label> <br/>
<textarea id="descripcion" name="descripcion" cols="40"
rows="5"></textarea>

</form>
```

La definición formal de la etiqueta `<textarea>` es:

<textarea>	Área de texto
Atributos comunes	básicos, i18n, eventos y foco
Atributos específicos	<p><code>rows = "numero"</code> - Número de filas de texto que mostrará el textarea</p> <p><code>cols = "numero"</code> - Número de caracteres que se muestran en cada fila del textarea</p> <p>Otros: name, disabled, readonly, onselect, onchange, onfocus, onblur</p> <p>HTML5 ha añadido varios atributos nuevos: autofocus, form, maxlength, placeholder, required, wrap</p>
Tipo de elemento	En línea
Descripción	Se emplea para incluir un área de texto en un formulario

Los atributos más utilizados en las etiquetas `<textarea>` son los que controlan su anchura y altura. La anchura del área de texto se controla mediante el atributo `cols`, que indica las columnas o número de caracteres que se podrán escribir como máximo en cada fila. La altura del área de texto se controla mediante `rows`, que indica directamente las filas de texto que serán visibles.

Hasta HTML5, el principal inconveniente de los elementos `<textarea>` es que no se podía limitar el número máximo de caracteres que se podían introducir mediante HTML, sólo se podía hacer mediante Javascript. Este problema se ha solucionado con el atributo `maxlength`, que también lo poseen los elementos `<input type="text">`. Con este atributo es posible limitar el número de caracteres permitidos en el área de texto.

Por otra parte, el otro control disponible para los formularios es el de las listas desplegables:



Figura 2.81. Ejemplo de uso de la etiqueta select

La imagen anterior muestra los tres tipos de listas desplegables disponibles. El primero es el de las listas más utilizadas que sólo muestran un valor cada vez y sólo permiten seleccionar un valor. El segundo tipo de lista es el que sólo permite seleccionar un valor pero muestra varios a la vez. Por último, el tercer tipo de lista desplegable es aquella que muestra varios valores y permite realizar selecciones múltiples.

El código HTML del ejemplo anterior se muestra a continuación:

```
<label for="so">Sistema operativo</label> <br/>
<select id="so" name="so">
  <option value="" selected="selected">- selecciona -</option>
  <option value="windows">Windows</option>
  <option value="mac">Mac</option>
  <option value="linux">Linux</option>
  <option value="otro">Otro</option>
</select>

<label for="so2">Sistema operativo</label> <br/>
<select id="so2" name="so2" size="5">
  <option value="windows" selected="selected">Windows</option>
  <option value="mac">Mac</option>
```

```
<option value="linux">Linux</option>
<option value="otro">Otro</option>
</select>

<label for="so3">Sistema operativo</label> <br/>
<select id="so3" name="so3" size="5" multiple="multiple">
  <option value="windows" selected="selected">Windows</option>
  <option value="mac">Mac</option>
  <option value="linux">Linux</option>
  <option value="otro">Otro</option>
</select>
```

Los tres tipos de listas desplegables se definen con la misma etiqueta `<select>` y cada elemento de la lista se define mediante la etiqueta `<option>`:

<select>	Lista desplegable
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<p><code>size = "numero"</code> - Número de filas que se muestran de la lista (por defecto sólo se muestra una)</p> <p><code>multiple = "multiple"</code> - Si se incluye, se permite seleccionar más de un elemento</p> <p>Otros: name, disabled, onchange, onfocus, onblur</p> <p>HTML5 ha añadido varios atributos nuevos: autofocus, form.</p>
Tipo de elemento	En línea
Descripción	Se emplea para incluir una lista desplegable en un formulario

<option>	Elemento de una lista desplegable
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<code>selected = "selected"</code> - Indica si el elemento aparece seleccionado por defecto al cargarse la página <code>value = "texto"</code> - El valor que se envía al servidor cuando el usuario elige esa opción Otros: label, disabled
Tipo de elemento	-
Descripción	Se emplea para definir cada elemento de una lista desplegable

La inmensa mayoría de listas desplegadas que utilizan las aplicaciones web son simples, por lo que el código HTML habitual de las listas desplegadas es:

```
<label for="so">Sistema operativo</label> <br/>
<select id="so" name="so">
  <option value="" selected="selected">- selecciona -</option>
  <option value="windows">Windows</option>
  <option value="mac">Mac</option>
  <option value="linux">Linux</option>
  <option value="otro">Otro</option>
</select>
```

La etiqueta `<select>` define la lista y encierra todas las opciones que muestra la lista. Cada una de las opciones de la lista se define mediante una etiqueta `<option>`. El atributo `value` de cada opción es obligatorio, ya que es el dato que se envía al servidor cuando el usuario envía el formulario. Para seleccionar por defecto una opción al mostrar la lista, se añade el atributo `selected` a la opción deseada.

Por otra parte, las listas desplegadas permiten agrupar sus opciones de forma que el usuario pueda encontrar fácilmente las opciones cuando la lista es muy larga:

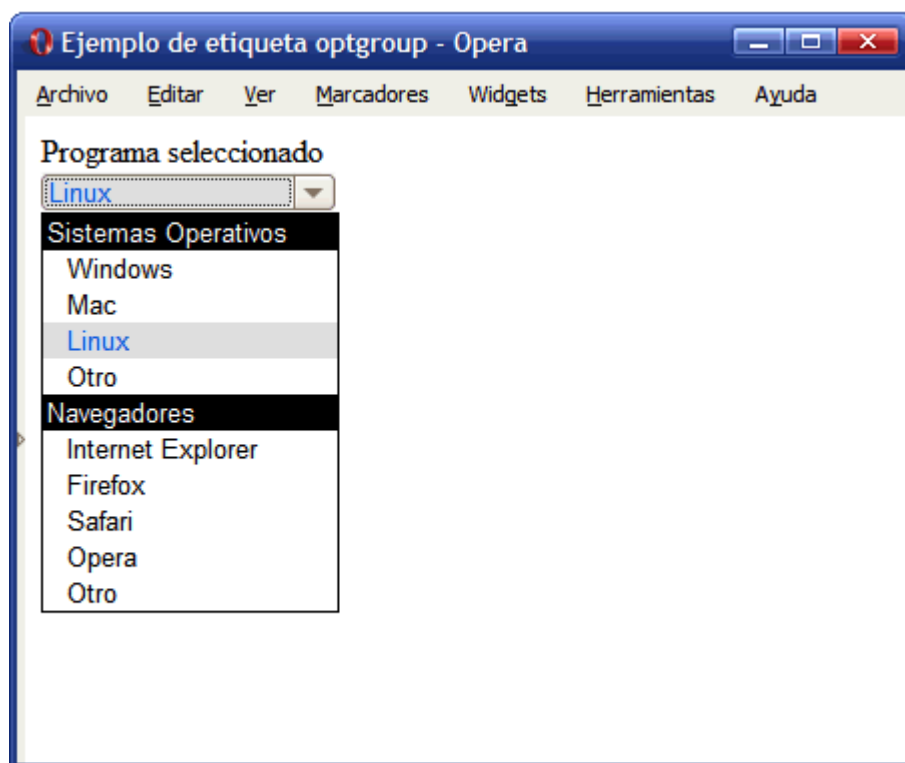


Figura 2.82. Ejemplo de uso de la etiqueta optgroup

El código HTML correspondiente a la imagen anterior se muestra a continuación:

```
<form id="formulario" method="post" action="">
<label for="programa">Programa seleccionado</label> <br/>
<select id="programa" name="programa">
  <optgroup label="Sistemas Operativos">
    <option value="Windows" selected="selected">Windows</option>
    <option value="Mac">Mac</option>
    <option value="Linux">Linux</option>
    <option value="Other">Otro</option>
  </optgroup>
  <optgroup label="Navegadores">
    <option value="Internet Explorer"
selected="selected">Internet Explorer</option>
    <option value="Firefox">Firefox</option>
    <option value="Safari">Safari</option>
    <option value="Opera">Opera</option>
    <option value="Other">Otro</option>
  </optgroup>
</select>

</form>
```

La etiqueta `<optgroup>` permite agrupar opciones relacionadas dentro de una lista desplegable. Su definición formal se muestra a continuación:

<optgroup>	Agrupación de elementos de una lista desplegable
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<code>label = "texto"</code> - Texto que se muestra como título de la agrupación de opciones Otros: disabled, selected
Tipo de elemento	-
Descripción	Se emplea para definir una agrupación lógica de opciones de una lista desplegable

El único atributo que suele utilizarse con la etiqueta `<optgroup>` es `label`, que indica el nombre de cada agrupación. Los navegadores muestran de forma destacada el título de cada agrupación, de forma que el usuario pueda localizar más fácilmente la opción deseada.

Ejercicio 16: Determinar el código HTML necesario para crear el formulario que se muestra en la siguiente imagen:

Figura 2.83. Formulario con controles de tipo lista desplegable

Ejercicio 17: Determinar el código HTML necesario para crear el formulario que se muestra en la siguiente imagen:

The screenshot shows a web browser window titled "Información sobre el producto - Opera". The browser's menu bar includes "Archivo", "Editar", "Ver", "Marcadores", "Widgets", "Herramientas", and "Ayuda". The main content area is titled "Información sobre el producto" and contains two sections:

- Datos básicos:**
 - A text input field labeled "Nombre".
 - A text area labeled "Descripción".
 - A "Foto" label next to a small image input field and an "Elegir..." button.
 - A checkbox labeled "Añadir contador de visitas".
- Datos económicos:**
 - A "Precio" label next to a text input field and a "€" symbol.
 - An "Impuestos" label next to a dropdown menu currently showing "4%". The dropdown menu is open, showing options: "4%", "7%", "16%", and "25%".
 - A "Promoción" label next to three radio buttons: "Ninguno" (selected), "Transporte gratuito", and "Descuento 5%".

Figura 2.84. Formulario complejo

8.5. La validación de los formularios

8.5.1. La validación del lado del cliente

Con HTML5 podremos realizar la validación de los datos introducidos en el navegador del lado del cliente. Esto no impide que luego se valide el formulario del lado del servidor, para una mayor seguridad y eficacia. La validación nativa del lado del cliente va a permitir que los diseñadores web no tengan que utilizar JavaScript.

Recuerda que cada navegador podrá elegir con total libertad cómo se va a indicar en pantalla que se producido un error.

8.5.2. Desactivar la validación

Si así se desea se puede desactivar sin ningún problema la validación nativa de los formularios en HTML5. Simplemente se deberá utilizar el atributo booleano `novalidate` en el elemento `<form>`.

```
<form action="maneja_formulario.php" method="post"
novalidate="novalidate">
...
</form>
```

También se puede indicar en el botón de envío del formulario, añadiendo el atributo `formnovalidate`.

```
<input type="submit" formnovalidate="formnovalidate"
name="enviar" value="Enviar sin validación" />
```

8.5.3. Insertar un campo obligatorio

Para hacer que un campo de introducción de datos sea obligatorio, hay que usar el atributo booleano `required`. El navegador no enviará el formulario hasta que el campo obligatorio no haya sido completado.

```
<input type="text" name="nombre" required="required" />
```

Ejercicio 18: Haz que los campos para el nombre, apellidos, dni, email y contraseña del formulario del ejercicio 15 sean requeridos.

8.5.4. Los valores autorizados

Se puede indicar al navegador cuáles son los valores autorizados para los campos de inserción de datos. Para ello usaremos el atributo `pattern`, que tendrá como valor una expresión regular. Con la expresión regular se debe indicar cuál es el valor autorizado y el navegador mostrará un mensaje de advertencia en caso de error.

Supongamos que, en un campo determinado, el usuario debe introducir una contraseña con un formato muy estricto: tres cifras, un guión y tres letras en mayúscula solamente.

Esta es la sintaxis:

```
<label for="clave"> Su contraseña: </label>
<input type="text" pattern="[0-9]{3}-[A-Z]{3}" name="clave"
id="clave" />
```


- [0-9]: Indica que se admiten todas las cifras de 0 a 9.
- {3}: Indica que es obligatorio introducir 3 cifras.
- -: Indica que se debe insertar el carácter –
- [A-Z]: Indica que se admiten todas las letras en mayúsculas de A a Z.
- {3}: Indica que es obligatorio introducir 3 letras.

8.5.5. Las expresiones regulares

En el ejemplo anterior hemos visto cómo usar una expresión regular en el atributo `pattern`. Veamos más detenidamente estas expresiones regulares.

1. **Letras autorizadas.** Esta es la sintaxis para indicar que el usuario solamente podrá insertar una letra `a` o `g` o `r` y que ésta deberá estar en minúsculas: `[agr]`. Cualquier otro valor no será válido.

La sintaxis sería:

```
<input type="text" pattern="[agr]" name="test" />
```

Ejemplos de valores no válidos: A ar

2. **Un intervalo de letras autorizadas.** Con esta sintaxis se puede indicar un intervalo de letras autorizadas: `[G-M]`. El usuario podrá indicar una letra mayúscula comprendida entre `G` y `M`, ambas incluidas.

Ejemplos de valores no válidos: k B

3. **Otras letras a excepción de.** Esta es la sintaxis para indicar que todas las letras están autorizadas, a excepción de las indicadas: `[^abc]`. El usuario podrá usar todos los caracteres salvo `a`, `b` y `c`.

Esta es la sintaxis para indicar que todas las letras están autorizadas, a excepción de las indicadas en el intervalo: `[^d-m]`. El usuario podrá usar todas las letras a excepción de las comprendidas entre `d` y `m`, ambas incluidas.

Ejemplos de valores no válidos: k

4. **Las mayúsculas y minúsculas.** Se puede exigir que las letras autorizadas estén en mayúsculas o en minúsculas.

Esta es la sintaxis para indicar que se puede introducir un único carácter en mayúscula: `[A-Z]`.

Esta es la sintaxis para indicar que se puede introducir un único carácter en minúscula: `[a-z]`.

Esta es la sintaxis para indicar que se puede introducir un único carácter en mayúscula o en minúscula: `[A-z]`.

5. **La condición O.** Es posible expresar varias posibilidades con el carácter `|`, que representa la función lógica O. Esta es la sintaxis para indicar que se puede introducir un carácter comprendido entre `a` y `e`, o `w`, y `z`: `[a-e|w-z]`.

6. **Las palabras autorizadas.** También se puede indicar qué palabras están autorizadas. Esta es la sintaxis para indicar que se pueden introducir las palabras `cosa` o `chisme` o `cacharro`: `(cosa|chisme|cacharro)`. Se puede exigir que las letras autorizadas estén en mayúsculas o en minúsculas.

Ejemplos de valores no autorizados: aparato

7. **Autorizar cifras.** Se sigue exactamente el mismo principio para las cifras. Esta es la sintaxis para indicar que se puede introducir una cifra comprendida entre `0` y `9`: `[0-9]`.

Ejemplos de valores no autorizados: 23

8. **Un número limitado de caracteres.** Se pueden indicar cuántas letras o cifras se admiten para que el valor introducido sea válido. Esta es la sintaxis para indicar que se pueden introducir 5 letras en mayúsculas: `[A-Z]{5}`.

Ejemplos de valores no válidos: AXE qwert HTML5

9. **Los valores múltiples.** Esta es la sintaxis para indicar que se puede introducir un valor que incluya dos letras en mayúsculas seguidas de tres cifras: `[A-Z]{2}[0-9]{3}`.

Ejemplos de inserción no válida: 123DE

10. **Los parámetros especiales.** Podemos usar algunos caracteres especiales para definir determinadas expresiones regulares.

- `\d` hace referencia a las cifras. `\d{3}` admite la introducción de 3 cifras.
- `\w` hace referencia a los caracteres. `\w{3}` admite la introducción de 3 caracteres.

- **+** indica que debe haber uno o más caracteres de los especificados anteriormente. De este modo el valor no estará limitado. **[a-z]+** indica que el usuario podrá insertar tantas letras minúsculas como desee.
- ***** indica que puede haber varios o ningún carácter de los especificados anteriormente. De este modo podemos obtener valores opcionales. **[A-Z]{2}[0-9]*** indica que el valor comenzará con dos mayúsculas que podrán ir seguidas de cero o n cifras.
- **^** indica que el valor deberá comenzar obligatoriamente por los elementos especificados justo a continuación. **^[0-9]{2}** indica que el valor deberá comenzar con dos cifras para que sea válido.
- **\$** indica que el valor deberá terminar obligatoriamente por los elementos específicos que la preceden. **[a-z]\$** indica que el valor deberá terminar con una letra en minúscula para que sea válido.

Esta es la sintaxis que permite autorizar un valor entre 20 y 29: **^(2[0-9])\$**. El valor deberá comenzar por 2 (**^(2**) y terminar con una cifra comprendida entre 0 y 9 (**[0-9])\$**).

8.5.6. Las ayudas para los usuarios de un formulario.

1. La ayuda para completar.

El atributo placeholder permite mostrar un valor de ejemplo en un campo de texto para ayudar al usuario. Este verá así un ejemplo del tipo de valor que tiene que insertar. En cuanto el usuario haga clic en el campo, el valor de ejemplo desaparecerá.

Esta es la sintaxis que podríamos usar, si volvemos a tomar el ejemplo anterior:

```
<input type="text" name="test" pattern="[0-9]{3}-[A-Z]{3}"  
placeholder="111-AAA" />
```

2. Activar un campo.

Se puede hacer que el punto de inserción parpadee en un campo específico utilizando el atributo booleano **autofocus**.

Veamos un ejemplo en el que se ha usado este atributo con una sintaxis abreviada:

```
<label for="nombre"> Su nombre y apellidos: </label>  
<input type="text" name="nombre" id="nombre"  
autofocus="autofocus" />
```

Cuando se abra la página, el punto de inserción parpadea en el interior del campo seleccionado.

Se debe usar un único `autofocus` en cada página.

Esta función se debe utilizar con moderación.

3. El completado automático

En los formularios, los navegadores usan de manera predeterminada el atributo `autocomplete` con el valor `on`, lo que permite acceder a la lista de los últimos valores introducidos en un campo. De este modo es posible seleccionar rápidamente un valor que ya hayamos insertado con anterioridad.

```
<form id="registro" method="post" action="registro.php"  
autocomplete="on">
```

Aunque se haya activado la funcionalidad de autocompletado para la totalidad de los campos de un formulario, no se tiene por qué usar, si no se desea, en un campo determinado:

```
<input type="url" name="url" id="url" autocomplete="off" />
```

9. OTROS ELEMENTOS DE HTML5

Como hemos visto en los puntos anteriores, HTML5 introduce algunos elementos nuevos y modifica o amplía otros. Esta lista que se muestra aquí amplía las nuevas etiquetas HTML5 vistas en puntos anteriores. No debemos olvidarnos de que la especificación de HTML5 se encuentra en continua evolución, por lo que es posible que aparezcan otros elementos nuevos.

9.1. El elemento `<hgroup>`

En HTML4 si se quería incluir un título y un subtítulo en una página web, se podía utilizar esta sintaxis:

```
<h1>Mi sitio web</h1>  
<p>Un sitio web sobre la creación de sitios web</p>
```

En este ejemplo, nada indica que exista un subtítulo.

También se podría usar la siguiente sintaxis:

```
<h1> Mi sitio web </h1>  
<h2> Un sitio web sobre la creación de sitios web</h2>
```

En ese caso, el resto de títulos de la página deberían usar las etiquetas de título que van de `<h1>` a `<h6>`.

En HTML5 el elemento `<hgroup>` permite agrupar los elementos del encabezado de tipo `<h1>` a `<h6>`. Resulta práctico para incluir en una página un título y un subtítulo, perfectamente definidos desde un punto de vista semántico.

Esta sería la sintaxis que debe usarse:

```
<hgroup>  
  <h1> Mi sitio web </h1>  
  <h2> Un sitio web sobre la creación de sitios web </h2>  
</hgroup>
```

Los elementos `<hx>` dentro de `<hgroup>` no tienen que empezar necesariamente con `<h1>`, sino que pueden tener el nivel de título que se prefiera. En otras palabras, no es obligatorio seguir una jerarquía estricta a la hora de usar las etiquetas `<hx>`. Se trata simplemente de una lista de elementos `<hx>` que se pueden combinar unos con otros. El objetivo de `<hgroup>` es agrupar los elementos de título que se quieren presentar. El elemento `<summary>` permite presentar un resumen de la información que se facilitará con el elemento `<details>`.

> no es el de crear un índice.

9.2. Los elementos `<details>` y `<summary>`

El elemento `<summary>` permite presentar un resumen de la información que se facilitará con el elemento `<details>`.

Ejemplo:

```
<details>  
  <summary> Haga clic para consultar todos los detalles de su  
pedido. </summary>  
  <p> Estos son los detalles de su pedido... </p>  
  <p> Y bla bla bla ... </p>  
</details>
```

Cada navegador podrá decidir cómo desea mostrar esos detalles.

9.3. El elemento `<progress>`

El nuevo elemento `<progress>` permite crear barras de progresión para las tareas. El movimiento de la barra de progresión se puede hacer con la ayuda de los atributos o bien con JavaScript.

Este sería un ejemplo de barra de progresión con dos atributos: `value`, para indicar el valor actual, y `max`, para indicar el valor máximo de la tarea. ermite presentar un resumen de la información que se facilitará con el elemento `<details>`.

```
<p>Progresión de la tarea:  
  <progress value="40" max="100"> 40% </progress>  
</p>
```

Nuevamente, el navegador podrá decidir cómo desea mostrarlo.

10. ESTRUCTURA Y LAYOUT

Los puntos anteriores muestran las decenas de etiquetas HTML disponibles para marcar y estructurar cada elemento individual de las páginas web: tablas, listas, enlaces, párrafos, imágenes, etc. Aunque combinando esas etiquetas es posible crear cualquier página web, no es posible hacer que las páginas muestren estructuras complejas.

La mayoría de páginas HTML disponen de estructuras complejas formadas por varias columnas de contenidos y otro tipo de divisiones. Utilizando exclusivamente HTML no es posible crear estas estructuras complejas, ya que es imprescindible emplear las hojas de estilos CSS.

No obstante, los estilos de CSS necesitan la ayuda de HTML para crear los diseños más avanzados. En concreto, el código HTML se encarga de agrupar los elementos de la página en diferentes divisiones en función de su finalidad: la zona de la cabecera de la página, la zona de contenidos, una zona lateral para el menú y otras secciones menores, la zona del pie de página, etc.

10.1. Los elementos de la estructura en HTML4

En HTML4, el elemento principal para estructurar las páginas web es la etiqueta `<div>`. Con el elemento `<div>` es posible crear zonas de visualización de forma rectangular.

La siguiente imagen muestra algunas de las zonas definidas en la página principal del sitio www.alistapart.com:

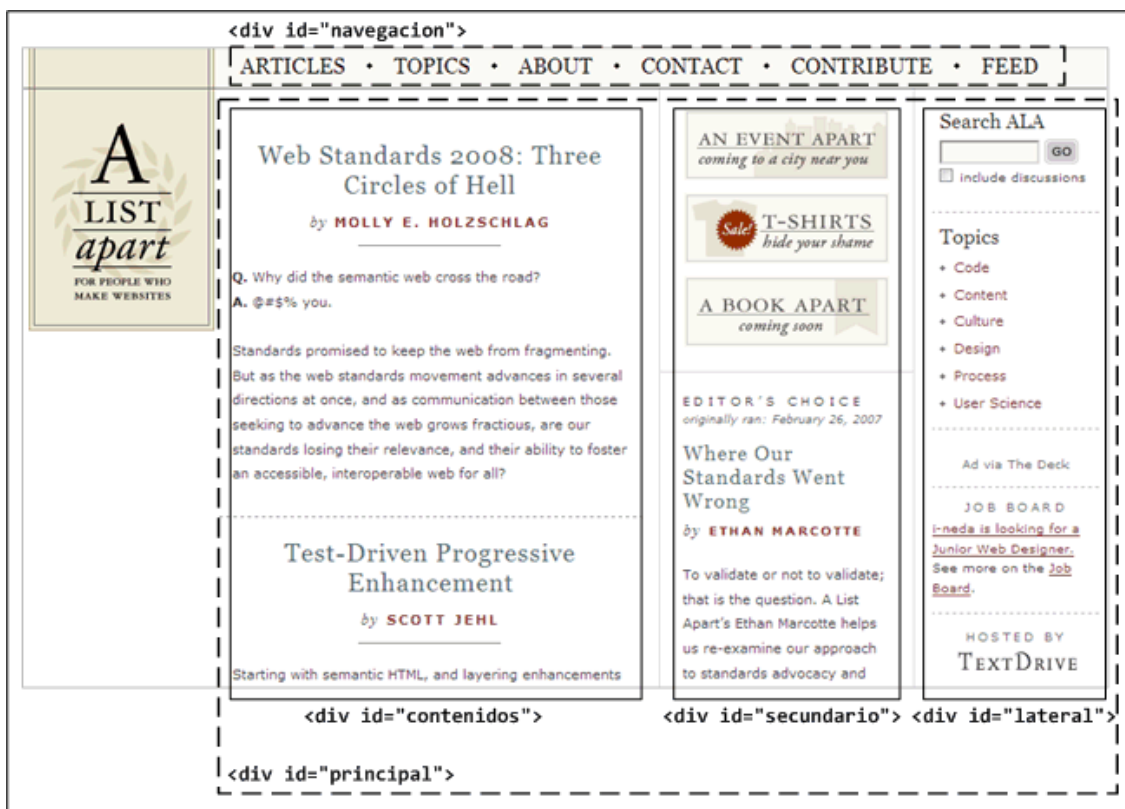


Figura 2.85. Ejemplo de página compleja estructurada con etiquetas div

Para agrupar los elementos que forman cada zona o división de la página se utiliza la etiqueta `<div>`:

<div>	Divisiones
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Agrupar elementos de bloque

El nombre de la etiqueta div tiene su origen en la palabra *división*, ya que esta etiqueta define zonas o divisiones dentro de una página HTML. En cualquier caso, casi todos los diseñadores web utilizan la palabra "capa" para referirse a una "división". Aunque se trata de un error grave (las capas se crean mediante una propiedad de CSS llamada *z-index*) es preferible seguir llamando "capas" a las zonas definidas con la etiqueta `<div>` para poder entenderse con el resto de diseñadores.

Las páginas web complejas que están bien diseñadas utilizan decenas de etiquetas `<div>`. Con mucha diferencia, los atributos más utilizados con esta etiqueta son `id`

(para identificar la capa de forma única) y `class` (para aplicar a la capa estilos CSS).

Por último, si observas el código HTML de algunas páginas web complejas, verás que la mayoría utilizan los mismos nombres para identificar sus divisiones. Los nombres más comunes, y sus equivalentes en inglés, se muestran a continuación:

- `contenedor` (`wrapper`) suele encerrar la mayor parte de los contenidos de la página y se emplea para definir las características básicas de la página: su anchura, sus bordes, imágenes laterales, si se centra o no respecto de la ventana del navegador, etc.
- `cabecera` (`header`) que incluye todos los elementos invariantes de la parte superior de la página (logotipo, imagen o banner, cuadro de búsqueda superior, etc.)
- `contenido` (`content`) engloba el contenido principal del sitio (la zona de noticias, la zona de artículos, la zona de productos, etc. dependiendo del tipo de sitio web)
- `menu` (`menu`) se emplea para agrupar todos los elementos del menú lateral de navegación de la página
- `pie` (`footer`) que incluye todos los elementos invariantes de la parte inferior de la página (aviso de copyright, política de privacidad, términos de uso, etc.)
- `lateral` (`sidebar`) se emplea para agrupar los elementos de las columnas laterales y secundarias de la página.

De esta forma, el esqueleto de una página HTML compleja suele ser similar al siguiente:

```
...  
<div id="contenedor">  
  <div id="cabecera">  
    ...  
  </div>  
  
  <div id="contenido">  
    <div id="menu">  
      ..  
    </div>  
    ...  
  </div>  
  
  <div id="pie">  
    ...  
  </div>  
</div>  
...
```

El equivalente para las páginas en inglés sería el siguiente:


```
...  
<div id="wrapper">  
  <div id="header">  
    ...  
  </div>  
  
  <div id="content">  
    <div id="menu">  
      ..  
    </div>  
    ...  
  </div>  
  
  <div id="footer">  
    ...  
  </div>  
</div>
```

En el siguiente ejemplo podemos ver la estructura de una página de tipo blog con cajas `<div>`:

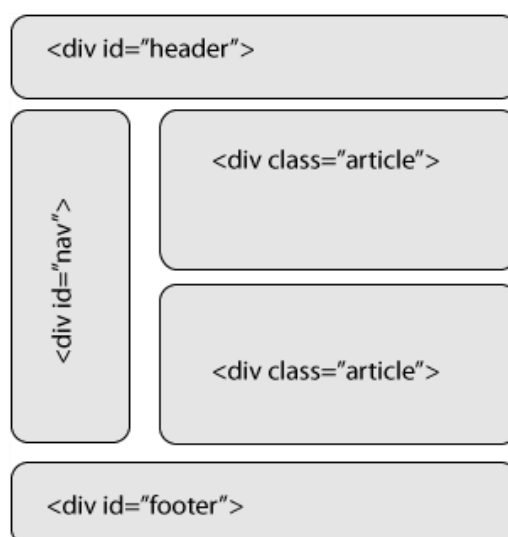


Figura 2.86. Ejemplo de estructura de una página con etiquetas `div`

Todas esas cajas `<div>` presentan otro problema: el de la semántica de los contenedores. Cada caja `<div>` se distingue de las demás gracias a su código de identificación único, que el diseñador web le habrá asignado. Por ese motivo, las cajas `<div>` no son semánticas: el contenido esperado no está definido por ningún parámetro. Una caja `<div>` identificada con `id="izquierda"` no nos aporta ningún dato sobre su contenido. Podría tratarse de una barra de navegación, de información legal o de cualquier otro tipo de contenido.

Ahora bien, la evolución del HTML se dirige hacia un mayor uso de la semántica.

10.2. Los elementos de la estructura en HTML5

1. Los nuevos elementos de la estructura

Con HTML5 llegan nuevos elementos de estructura semántica. Estos nuevos elementos han sido definidos y se les ha asignado un nombre tras un largo análisis de los nombres más usados en las cajas `<div>`.

2. El elemento `<header>`

El nuevo elemento `<header>` permite insertar una zona de visualización para las cabeceras. Puede definirse esta cabecera para toda la página o para una zona determinada: artículo, menú lateral...

Debemos considerar que este elemento se puede usar desde dos puntos de vista:

- a nivel de la página: es la típica cabecera de la página, que a menudo aparece en lo alto de la pantalla, con un logotipo, un eslogan, un menú de navegación principal...
- a nivel del contenido: permite disponer de una zona de introducción del contenido que se incluya a continuación.

Esta es la definición del W3C para el elemento `<header>`: *"The header element represents a group of introductory or navigational aids. A header element is intended to usually contain the section's heading (an h1–h6 element or an hgroup element), but this is not required. The header element can also be used to wrap a section's table of contents, a search form, or any relevant logos."*

Como puedes ver, el elemento `<header>` no tiene que aparecer obligatoriamente en la estructura de la página.

3. El elemento `<footer>`

El nuevo elemento `<footer>` permite insertar una zona de visualización para los pie de página. Volvemos a encontrarnos con la misma semántica de los encabezados. Es posible definir un pie de página para la página completa, o bien para cualquier otra zona de visualización o para un artículo.

Esta es la definición del W3C: *"The footer element represents a footer for its nearest ancestor sectioning content or sectioning root element. A footer typically contains information about its section such as who wrote it, links to related documents, copyright data, and the like."*

Su uso es similar al de los `<header>`, pero aplicado a los pie de página. Por lo tanto, no debemos seguir al "pie de la letra" la traducción literal de "pie de página". Se trata más bien de un "pie de zona de visualización", donde la zona de visualización puede ser una página, una sección, un artículo...

4. El elemento `<nav>`

El elemento `<nav>`, como su nombre indica, está pensado para la visualización de una zona de navegación con vínculos hipertexto. Cuidado, no quiere decir que solo se pueda tener una zona de navegación en cada página, o que haya que crear tantos elementos `<nav>` como zonas de navegación haya en tus páginas, bastará con que los identifiques correctamente. El elemento `<nav>` está pensado en especial para la navegación principal del sitio web, para la inserción de vínculos entre las páginas de dicho sitio web.

La definición del W3C es bastante clara: *"The nav element represents a section of a page that links to other pages or to parts within the page: a section with navigation links. Note: Not all groups of links on a page need to be in a nav element – the element is primarily intended for sections that consist of major navigation blocks. In particular, it is common for footers to have a short list of links to various pages of a site, such as the terms of service, the home page, and a copyright page. The footer element alone is sufficient for such cases; while a nav element can be used in such cases, it is usually unnecessary."*

5. El elemento `<section>`

El elemento `<section>` permite agrupar elementos que tengan la misma temática. De este modo podemos agrupar en un mismo elemento un contenido, con su título y su pie de página.

Esta es la definición del W3C: *"The section element represents a generic section of a document or application. A section, in this context, is a thematic grouping of content, typically with a heading."*

6. El elemento `<article>`

El elemento `<article>` permite insertar un contenido autónomo, que puede volver a usarse en otro lugar del sitio web, sin que por ello se anule su significado.

Esta es la definición del W3C: *"The article element represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication. This could be a forum post,*

a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content."

Claro está, un artículo podría tener una cabecera (`<header>`), un pie de página (`<footer>`) y varios títulos (`<hx>`).

7. El elemento `<aside>`

El elemento `<aside>` permite mostrar un contenido relacionado con el contenido al cual esté vinculado. Puede tratarse de barras laterales (*sidebars*), de zonas de widgets, de complementos sobre artículos, etc.

8. Las cajas `<div>`

Y, por último, hay que aclarar que aunque usemos HTML5 no tenemos por qué erradicar el uso de las cajas `<div>`. ¡Todavía podemos usarlas y siguen teniendo su utilidad!

10.3. El atributo semántico "role"

El XHTML (<http://www.w3.org/1999/xhtml/vocab/#XHTMLRoleVocabulary>) y el XHTML2 *Working Group* preconizaban el uso del atributo `role` (<http://www.w3.org/TR/xhtml2/mod-roleAttribute.html>) para definir de forma más semántica los elementos estructurales de una página web.

En HTML5 podemos usar el atributo `role` para incluir esa información adicional gracias al módulo **WAI-ARIA** (<http://www.w3.org/TR/html5/content-models.html#wai-aria>). Este módulo se ocupa de la gestión del contenido de las páginas web para las personas discapacitadas. Determinados elementos HTML5 tienen un role implícito, como, por ejemplo, el elemento `<nav>`, cuyo `role` implícito es `navigation`.

Estos son los principales valores del atributo `role`:

- `main`: define el contenido principal de un documento.
- `secondary`: define una parte secundaria del documento.
- `navigation`: define la barra de navegación del documento.
- `banner`: aparece por lo general en lo alto de la página y suele contener el logotipo y el eslogan de la empresa.
- `contentinfo`: indica que dicho elemento aporta información sobre el contenido de la página (autores, copyrights, menciones legales...).
- `definition`: presenta la definición de un elemento.

- **note**: corresponde, por lo general, a una nota entre paréntesis o al final de la página.
- **seealso**: indica que el elemento contiene información relacionada con el contenido principal de la página.
- **search**: contiene el formulario de búsqueda de una página web.

Veamos un ejemplo simple de cómo usarlo:

```
<div id="buscar" role="search">  
...  
</div>
```

Otro ejemplo:

```
<header id="banner" role="banner">  
...  
</header>
```

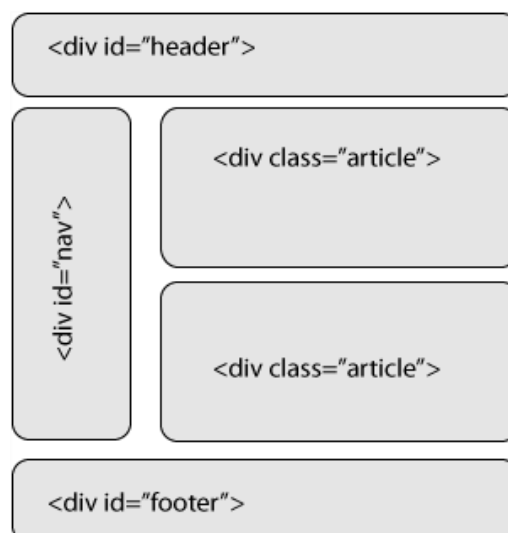
Para obtener más información sobre *Accessible Rich Internet Applications* (**WAI-ARIA**), consulta la *Candidate Recommendation* del 18 de enero de 2011:

<http://www.w3.org/TR/2011/CR-wai-aria-20110118/>

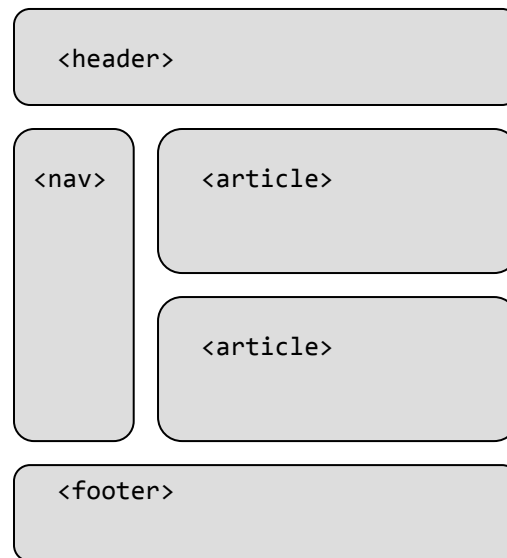
10.4. Ejemplos de estructura en HTML5.

1. Una estructura simple

Tomamos como ejemplo la siguiente estructura de página:



Podemos modificarla para adaptarla a HTML5



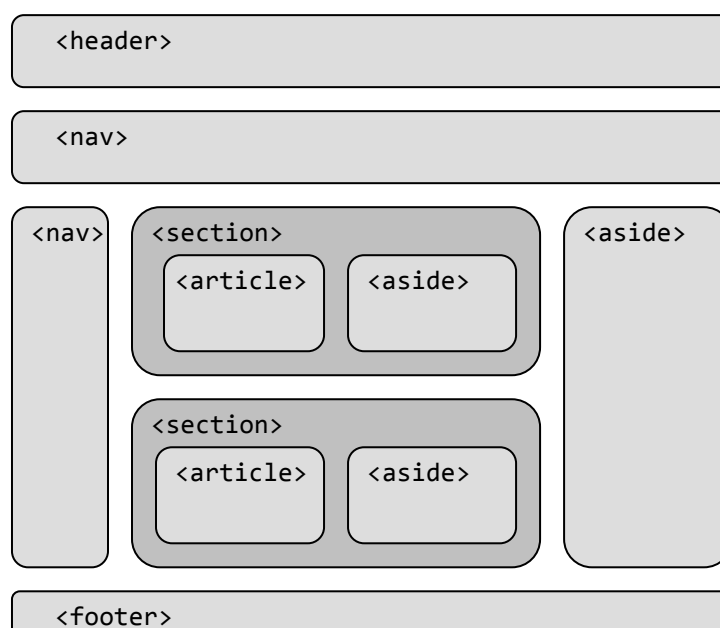
En el elemento `<header>` encontraremos los elementos del encabezado en la página, como el logotipo y el eslogan, por ejemplo.

En el elemento `<nav>`, situado a la izquierda, encontraremos los vínculos que nos permitirán navegar por ese sitio web.

Todos los artículos del blog estarán colocados, por supuesto, dentro de los elementos `<article>`.

Por último, el pie de página, `<footer>`, podrá contener las menciones legales o los vínculos de contacto, por ejemplo.

2. Una estructura más elaborada



Encontramos de nuevo el elemento `<header>`, que ya todos conocemos.

Debajo tendremos el primer elemento `<nav>`, para el menú de navegación general del sitio web, que nos permitirá navegar por las distintas páginas.

A la izquierda tenemos una segunda caja `<nav>`, para la navegación secundaria, que contiene los vínculos relacionados directamente con el contenido de la página en cuestión.

A la derecha encontramos el elemento `<aside>`, que muestra información relacionada con el contenido de la página, como los vínculos promocionales o los contenidos relacionados, por ejemplo.

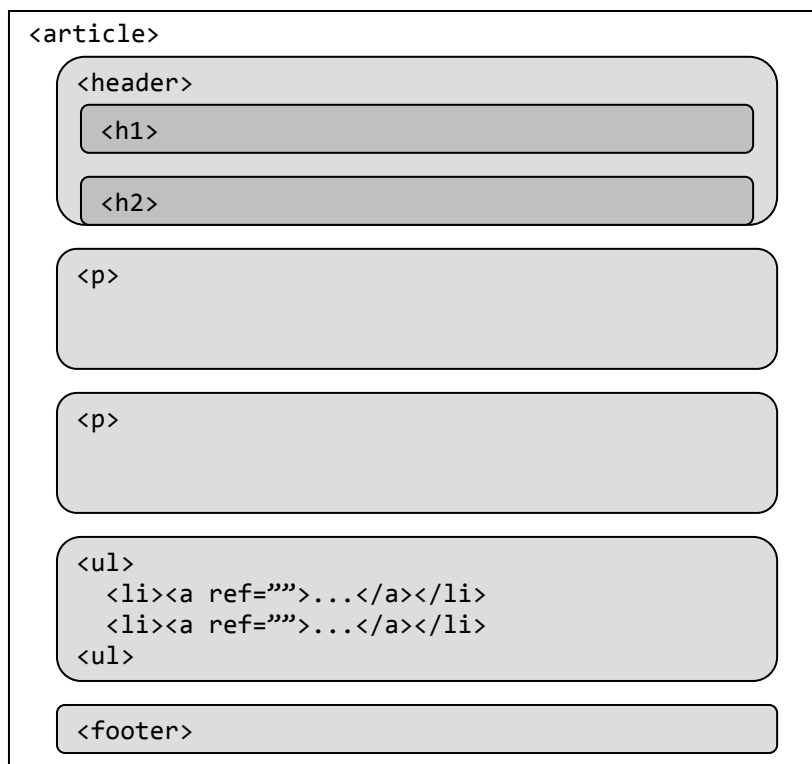
El contenido de la página aparece dentro de dos elementos `<section>`, de modo que podamos distinguir con facilidad esos dos contenidos diferentes. Cada `<section>` contiene un elemento `<article>` para el contenido textual y un elemento `<aside>` para incluir los elementos de información adicional relacionados con el artículo (iconografía, enlaces,...).

Por último, la página presenta un pie de página `<footer>` que contiene la información legal, las condiciones de venta, un vínculo de contacto, un plano de acceso,...

Claro está, cada elemento de la estructura deberá contar con un código de identificación único o con una clase común para varios elementos.

3. La estructura de un artículo

Veamos ahora la estructura de lo que podría ser un artículo de un sitio web en HTML5.



Tenemos un elemento `<article>` como contenedor general.

Nuestros artículos presentan cabeceras, introducciones, por lo que hemos usado el elemento `<header>`. Este elemento `<header>` contiene el título `<h1>` del artículo y su subtítulo `<h2>`.

El contenido textual del artículo se sitúa entre los elementos `<p>`. El artículo incluye vínculos, que completan el contenido facilitado, ordenados en una lista ``.

Por último, el artículo presenta un pie de página `<footer>` o, mejor dicho, un “pie de artículo”, con la fecha de publicación, la sección a la que pertenece dicho artículo y el nombre del autor, por ejemplo.

10.5. Plantillas para sitios web en HTML5.

Desde que se publicó HTML5, multitud de diseñadores web han creado plantillas de diseño de sitios web (en inglés, *templates*) que utilizan HTML5.

10.5.1. La plantilla ArchiteXture

1. La fuente

Esta es la URL de la plantilla: <http://freehtml5templates.com/architecture-html5-and-css3-template/>. A partir de esta dirección podremos probarla en línea y descargar los archivos fuente.

2. El diseño del sitio web

Observa el diseño de esta plantilla, que como ves es bastante sobrio.



Podemos distinguir fácilmente las cuatro “zonas” de visualización del sitio we:

- el menú de navegación en la parte superior.
- El banner de presentación del sitio web, que contiene el nombre, un eslogan y un logotipo.
- La zona central, que muestra información general en el lado izquierdo y los artículos en la parte derecha,
- El pie de página, que contiene los vínculos, en la parte inferior.

3. La estructura general

La estructura general del sitio web refleja fielmente la presentación visual.

Tenemos una caja `<div id="wrapper">` que sirve de contenedor general y que permite centrar el sitio web en la ventana del navegador.

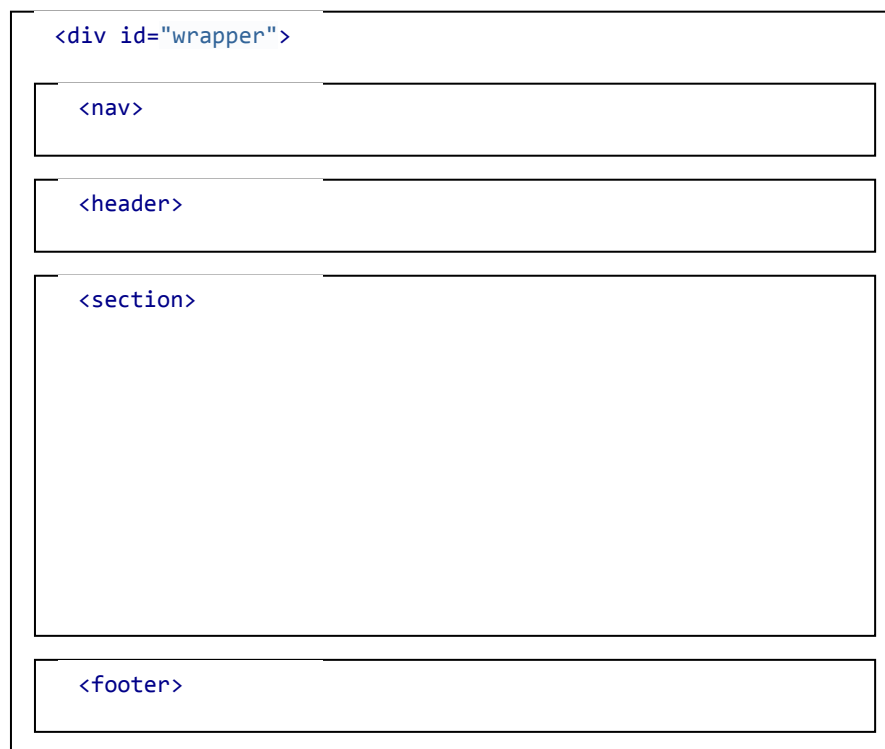
Encontramos a continuación el elemento `<nav>`, que contiene el menú de navegación de la parte de arriba. El diseñador ha empleado correctamente el elemento `<nav>`, ya que lo ha usado para insertar el menú de navegación principal del sitio web.

La información general del sitio web se ha insertado en un elemento `<header>`. Resulta bastante acertado, ya que se trata de información general que se mostrará en todas las páginas del sitio web. Efectivamente, se trata de una cabecera de página de sitio web.

La zona central del sitio web es un elemento `<section>`. Una sección es una zona de visualización que reúne elementos con una temática similar. En el ejemplo se trata, simplemente, de mostrar esos elementos en la zona central de la página.

El pie de página es, lógicamente, un elemento `<footer>`.

Veamos la estructura de las cajas.



4. El código de la estructura

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8" />
```

```
<title>ArchiteXture</title>
<link rel="stylesheet" href="styles.css" type="text/css"
media="screen" />
<link rel="stylesheet" type="text/css" href="print.css"
media="print" />
<!--[if IE]><script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></scrip
t><![endif]-->
</head>
<body>
<div id="wrapper">
  <nav>
    ...
  </nav>

  <header>
    ...
  </header>

  <section id="main">
    <section id="content">
      <article>
        ...
      </article>

      <article>
        ...
      </article>

    </section>

    <aside id="sidebar">
      ...
    </aside>

  </section>

  <footer>
    ...
  </footer>
</div>
</body>
</html>
```

5. El menú de navegación

El menú de navegación (`<nav>`), contiene una caja `<div>` que incluye la clásica lista (``) de vínculos.

Este es el código:

```
<nav><!-- top nav -->
  <div class="menu">
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">About</a></li>
      <li><a href="#">Products</a></li>
      <li><a href="#">Services</a></li>
      <li><a href="#">Locations</a></li>
      <li><a href="#">Contact Us</a></li>
    </ul>
  </div>
</nav><!-- end of top nav -->
```

La visualización del menú de navegación:

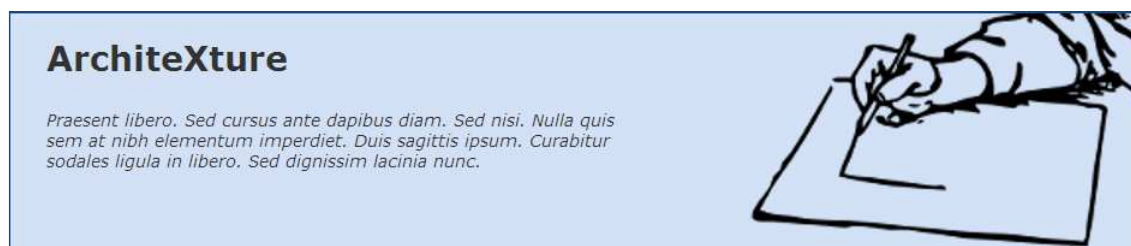


6. El banner de presentación

El banner de presentación (`<header>`), contiene una caja `<div>` con el logotipo, un título de nivel 1 (`<h1>`) y un párrafo.

```
<header><!-- header -->
  <div id="plandesign"></div>
  <h1><a href="#">ArchiteXture</a></h1>
  <p>Praesent libero...</p>
</header><!-- end of header -->
```

La visualización del banner de presentación:



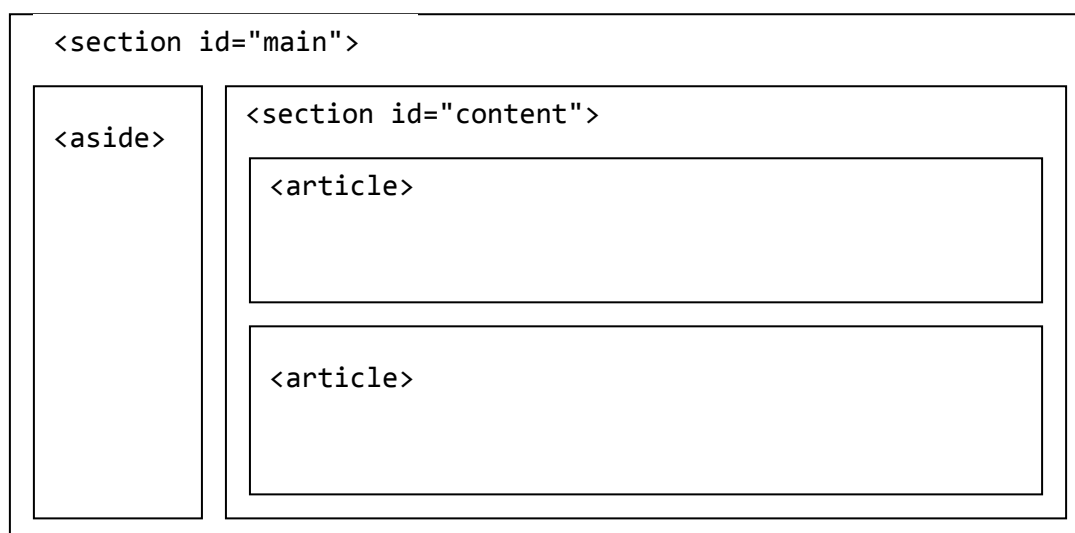
7. La zona central

La zona central del sitio web se ha insertado en un elemento `<section id="main">`. Este contiene a su vez otra caja `<section id="content">` para mostrar el contenido propiamente dicho del sitio web, los artículos de la parte derecha, y un elemento `<aside id="sidebar">` para mostrar información general en la parte izquierda.

El elemento `<section id="content">` contiene, lógicamente, los elementos `<article>` para los distintos artículos del sitio web. Cada artículo contiene títulos `<h2>` y párrafos `<p>`.

El elemento `<aside>` contiene diversos elementos: títulos `<h3>`, listas `` e imágenes ``.

Esta es la estructura de las cajas:



Veamos el código de la parte central:

```
<section id="main"><!-- #main content and sidebar area -->
  <section id="content"><!-- #content -->
    <article>
      <h2><a href="#">First Article Title</a></h2>
      <p>Lorem ipsum ....</p>
    </article>

    <article>
      <h2><a href="#">Second Article Title</a></h2>
      <p>Lorem ipsum ...</p>
      <p>Nulla quis ...</p>
      <p>Mauris massa...</p>
    </article>
```

```
</section><!-- end of #content -->
<aside id="sidebar"><!-- sidebar -->
  <h3>Things To Do</h3>
  <ul>
    <li><a href="#">Play Games and Network</a></li>
    <li><a href="#">Chat With Friends</a></li>
    <li><a href="#">Swap Exciting Stories</a></li>
    <li><a href="#">Sell Tons of Your Stuff</a></li>
    <li><a href="#">Buy Stuff You Don't Need</a></li>
    <li><a href="#">Trade Stuff With Others</a></li>
  </ul>
  <h3>Sponsors</h3>
  
  <br/>
  
  <br/><br/>
  <h3>Connect With Us</h3>
  <ul>
    <li><a href="#">Twitter</a></li>
    <li><a href="#">Facebook</a></li>
  </ul>
</aside><!-- end of sidebar -->
</section><!-- end of #main content and sidebar-->
```

Así se representa a zona central, `<section id="main">`:

Things To Do

- Play Games and Network
- Chat With Friends
- [Swap Exciting Stories](#)
- Sell Tons of Your Stuff
- Buy Stuff You Don't Need
- Trade Stuff With Others

Sponsors

ADVERTISE HERE	ADVERTISE HERE
ADVERTISE HERE	ADVERTISE HERE

Connect With Us

- [Twitter](#)
- [Facebook](#)

First Article Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla.

Second Article Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc.

Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

8. El pie de página

El pie de página utiliza el elemento `<footer>`. Este contiene dos elementos `<section>` para obtener el diseño deseado. Dentro del segundo `<section>`

encontramos cuatro elementos `<aside>`, que corresponden a las cuatro zonas de visualización.

Este es el código:

```
<footer>
  <section id="footer-area">
    <section id="footer-outer-block">
      <aside class="footer-segment">
        <h4>Friends</h4>
        <ul>
          <li><a href="#">one linkylink</a></li>
          <li><a href="#">two linkylinks</a></li>
          <li><a href="#">three linkylinks</a></li>
        </ul>
      </aside><!-- end of #first footer segment -->

      <aside class="footer-segment">
        <h4>Awesome Stuff</h4>
        <ul>
          <li><a href="#">one linkylink</a></li>
          <li><a href="#">two linkylinks</a></li>
          <li><a href="#">three linkylinks</a></li>
        </ul>
      </aside><!-- end of #second footer segment -->

      <aside class="footer-segment">
        <h4>Coolness</h4>
        <ul>
          <li><a href="#">one linkylink</a></li>
          <li><a href="#">two linkylinks</a></li>
          <li><a href="#">three linkylinks</a></li>
        </ul>
      </aside><!-- end of #third footer segment -->

      <aside class="footer-segment">
        <h4>Blahdyblah</h4>
        <p>Integer nec odio. Praesent libero. Sed cursus ante dapibus diam.</p>
      </aside><!-- end of #fourth footer segment -->

    </section><!-- end of footer-outer-block -->
  </section><!-- end of footer-area -->
</footer>
```

Y el diseño obtenido:

Friends	Awesome Stuff	Coolness	Blahdyblah
<ul style="list-style-type: none">• one linkylink• two linkylinks• three linkylinks	<ul style="list-style-type: none">• one linkylink• two linkylinks• three linkylinks	<ul style="list-style-type: none">• one linkylink• two linkylinks• three linkylinks	Integer nec odio. Praesent libero. Sed cursus ante dapibus diam.

10.5.2. La plantilla Da Front Page

1. La fuente

Esta es la URL de dicha plantilla: <http://freehtml5templates.com/dafrontpage-html5-and-css3-template/>. A partir de esta dirección podremos probarla en línea y descargar los archivos fuente.

2. El diseño del sitio web

Así se presenta esta plantilla, al estilo de una revista.



3. La estructura general

La estructura general del sitio web aparece dentro de una caja `<div id="wrapper">`

El título del sitio web se encuentra dentro de un elemento de título de nivel 1: `<h1>`.

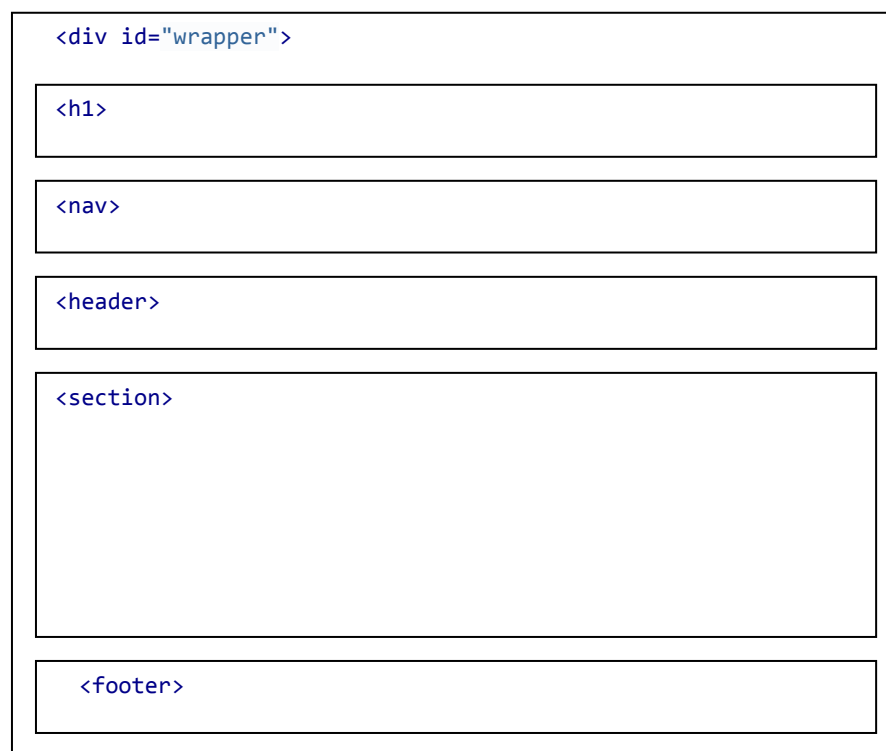
El menú de navegación se incluye, como cabía esperar, dentro de un elemento `<nav>`.

El artículo resaltado en la parte superior se encuentra dentro del elemento `<header>` de la página.

La estructura de tres columnas se encuentra dentro de un elemento `<section>`, ya que las tres columnas presentan un contenido similar.

Por último, el pie de página aparece, claro está, dentro de un elemento `<footer>`.

Veamos la estructura de las cajas:



Veamos el código de la estructura general:

```
<!doctype html>  
<html lang="en">  
<head>  
  <meta charset="utf-8" />
```

```
<title>DaFrontPage</title>
<link rel="stylesheet" href="styles.css" type="text/css"
media="screen" />
<link rel="stylesheet" type="text/css" href="print.css"
media="print" />
<!--[if IE]><script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></scrip
t><![endif]-->
</head>
<body>
<div id="wrapper">
  <h1><a href="#">Da Front Page</a></h1>
  <nav>
    ...
  </nav>
  <header>
    ...
  </header>
  <section id="main">
    ...
  </section>

  <footer>
    ...
  </footer>
</div>
</body>
</html>
```

4. El título del sitio web

El título del sitio web ha sido insertado dentro de un elemento `<h1>`.

```
<h1><a href="#">Da Front Page</a></h1>
```

Da Front Page

5. El menú de navegación

El menú de navegación aparece lógicamente dentro de un elemento `<nav>`. En una caja `<div>`, todos los vínculos están ordenados con la clásica lista ``.

Este es el código que se ha usado:

```
<nav><!-- top nav -->
  <div class="menu">
    <ul>
      <li><a href="#">Top News</a></li>
      <li><a href="#">National</a></li>
      ...
    </ul>
  </div>
</nav><!-- end of top nav -->
```

La visualización del menú de navegación:

Top News National World Politics Business Technology Sports Entertainment Science Health Travel

6. El encabezado del sitio web

En el elemento `<header>` encontramos el artículo “en portada”. Es el diseñador del sitio web quien ha querido mostrar un artículo como cabecera del sitio web. Siguiendo la lógica semántica, este artículo se mostrará en todas las páginas del sitio web.



FANS MOURN POP SINGER'S TRAGIC DEATH

Duis sagittis ipsum. Praesent de mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Vestibulum lacinia arcu eget nulla.

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Duis sagittis ipsum. Praesent de mauris.

El elemento `<header>` contiene dos cajas `<div>`: una para mostrar la imagen y otra para el contenido textual.

Este es el código que se ha usado:

```
<header><!-- header -->
  <div class="headlineimage">
    
```

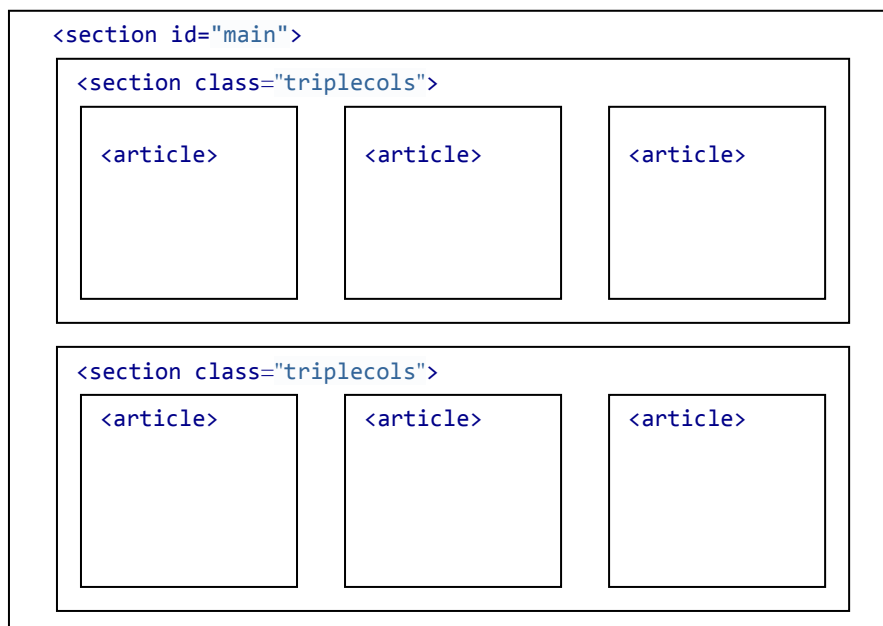
```
</div>
<div class="headline">
  <h2><a href="#">Fans Mourn Pop Singer's Tragic Death</a></h2>
  <p>Duis sagittis ipsum... </p>
  <p>Class aptent taciti ... </p>
</div>
</header><!-- end of header -->
```

7. La zona central

La zona central del sitio web, que contiene los artículos en columnas, utiliza el elemento `<section id="main">`. Este contiene dos elementos `<section class="triplecols">` para presentar las dos series de artículos en tres columnas.

Cada elemento `<section class="triple-cols">` contiene tres elementos `<article>`. Y cada elemento `<article>` contiene, como es lógico, un artículo, que incluye elementos `<a>`, para los vínculos, `` para las imágenes y `<p>` para los párrafos.

Veamos la estructura de los artículos en columnas:



Este es el código que se ha usado:

```
<section id="main"><!-- #main content area -->

  <!-- Triple columns with images, captions, and info blocks -->
  <section class="triplecols">
```

```
<article class="tripleblocks tripleleftblock">
  <a href="#">
    
    <span class="caption"><b>Protest Outside
Courtroom</b></span>
  </a>
  <p class="byline">By Jeffrey Wiggins</p>
  <p>Duis sagittis ipsum. ... </p>
</article>

<article class="tripleblocks triplemiddleblock">
...
</article>
<article class="tripleblocks triplemiddleblock">
...
</article>

</section>

<!-- Triple columns with images, captions, and info blocks -->
<section class="triplecols">
  <article class="tripleblocks tripleleftblock">
</article>

  <article class="tripleblocks triplemiddleblock">
...
</article>
  <article class="tripleblocks triplemiddleblock">
...
</article>

</section>

</section><!-- end of #main content -->
```

Así se presenta la zona central con los artículos:



PROTEST OUTSIDE COURTROOM

By JEFFREY WIGGINS

Duis sagittis ipsum. Praesent de mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi.



THOMPSON WINS SENATE RACE

By ANDERSON MEALEY

Duis sagittis ipsum. Praesent de mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero.



SCHOOLS GET HUGE BOOST IN DONATIONS

By MARCIA BURNS

Duis sagittis ipsum. Praesent de mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero.



ROADS CLOSED DUE TO STORM

By JIM STRONG

Duis sagittis ipsum. Praesent de mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi.



STRANGE ACCIDENT IN MIDTOWN

By LARRY EDMONDS

Duis sagittis ipsum. Praesent de mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero.



FIREFIGHTERS PREPARE FOR DRY SEASON

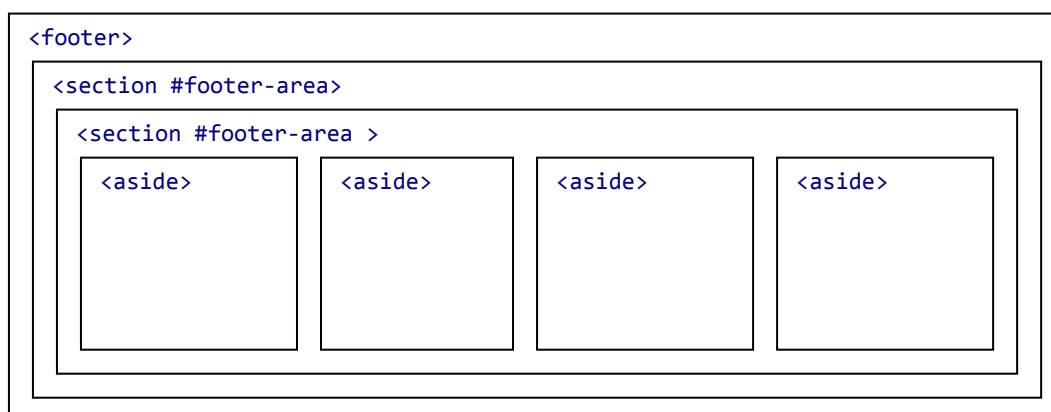
By FAITH CHANCE

Duis sagittis ipsum. Praesent de mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero.

8. El pie de página

El pie de página `<footer>` usa dos cajas `<div>` para su presentación. Cada una de las cuatro zonas de visualización utiliza un elemento `<aside>`.

Esta es estructura del pie de página:

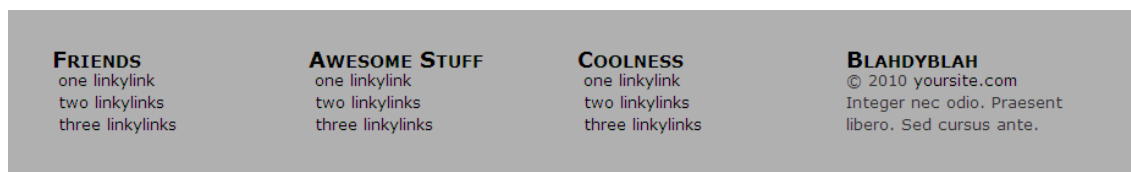


Y este es el código que se ha usado:

```
<footer>
  <section id="footer-area">
    <section id="footer-outer-block">
      <aside class="footer-segment first">
        <h3>Friends</h3>
        <ul>
          <li><a href="#">one linkylink</a></li>
          <li><a href="#">two linkylinks</a></li>
          <li><a href="#">three linkylinks</a></li>
        </ul>
      </aside><!-- end of #first footer segment -->
      <aside class="footer-segment second">
        ...
      </aside><!-- end of #second footer segment -->
      <aside class="footer-segment third">
        ...
      </aside><!-- end of #third footer segment -->
      <aside class="footer-segment last">
        ...
      </aside><!-- end of #fourth footer segment -->

    </section><!-- end of footer-outer-block -->
  </section><!-- end of footer-area -->
</footer>
```

Así se visualiza el pie de página:



10.5.3. La plantilla Learning Center

1. La fuente

Esta es la URL de dicha plantilla: <http://www.templatemonster.com/free-templates/free-website-template-learning-center.php>. A partir de esta dirección podremos probarla en línea y descargar los archivos fuente.

2. El diseño del sitio web

Veamos el diseño de esta plantilla, de tipo portal web para centros educativos.



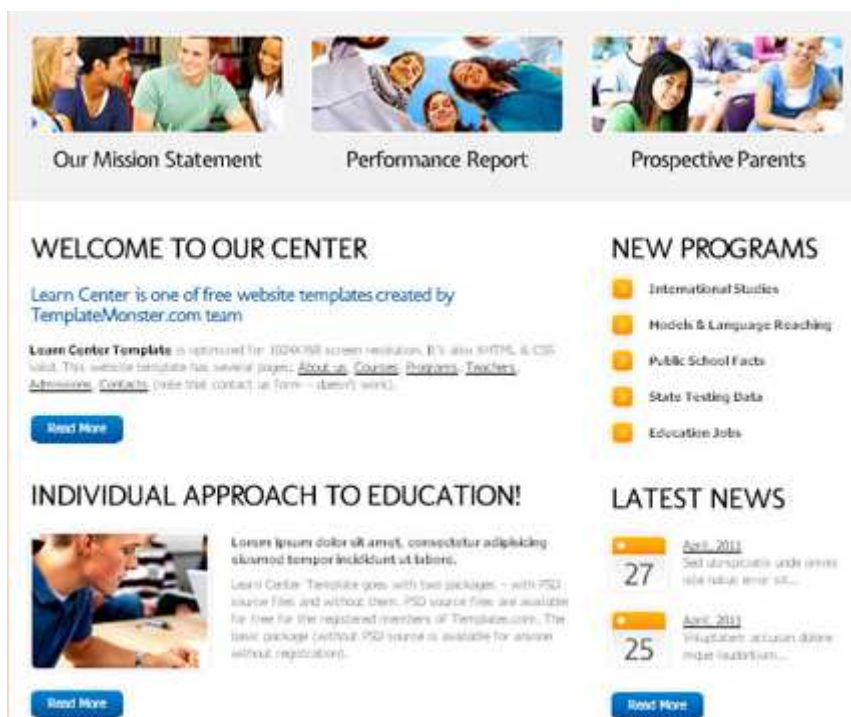
3. La estructura general

El sitio web se compone de dos cajas `<div>`, que utilizan dos clases CSS.

La primera caja `<div class="body1">` permite visualizar la parte superior del sitio web. Esta contiene a su vez una caja `<div class="main">` que se ha usado para aplicar el diseño. En esta caja encontramos un elemento `<header>` que corresponde a la parte superior del sitio web, la cabecera.



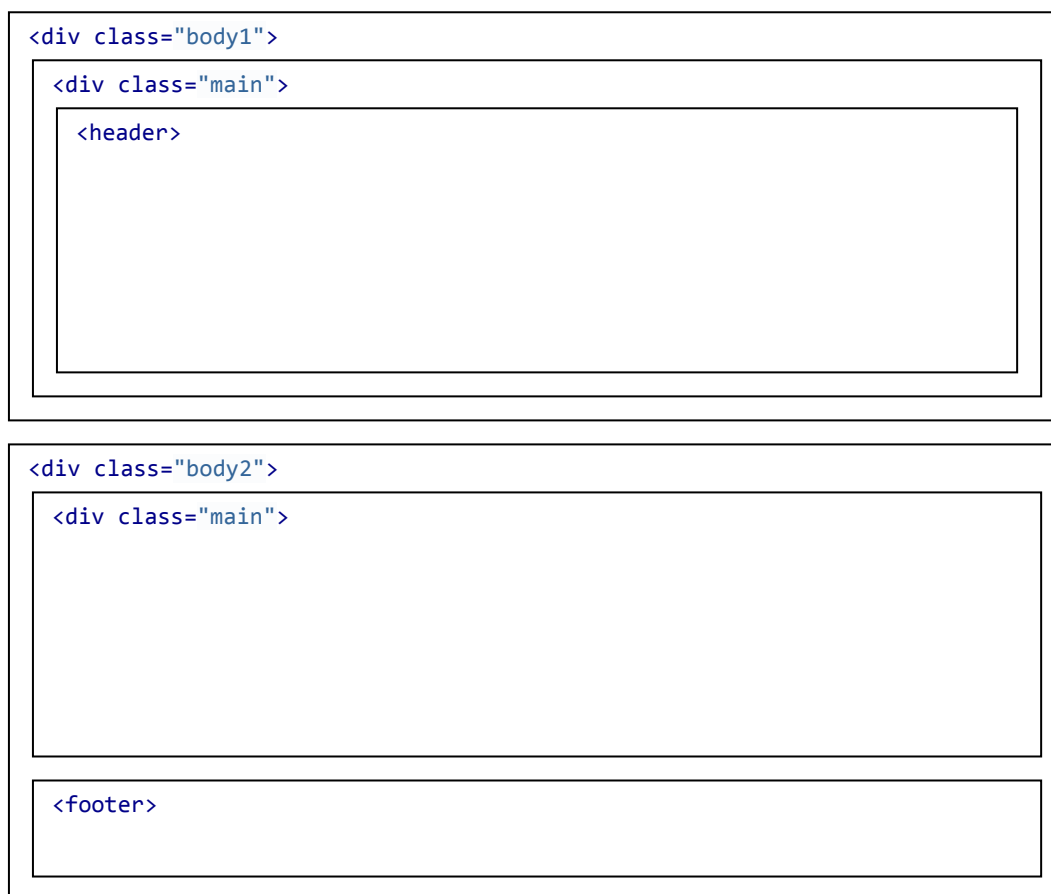
La segunda caja `<div class="body2">` permite visualizar la parte inferior del sitio web. Esta contiene a su vez una caja `<div class="main">` que se ha usado para aplicar el diseño. Esta caja contiene un elemento `<section id="content">` para la visualización de la zona central del sitio web.



A continuación tenemos una caja `<footer>` para la visualización del pie de página del sitio web.



Esta es la estructura general:



4. La cabecera

Como acabamos de ver, el elemento `<header>` se encuentra dentro de dos cajas `<div>`. Este elemento `<header>` contiene a su vez tres cajas `<div>` para las tres partes de la cabecera y una lista ``:

- el menú de navegación.
- El título “Learn Center”,
- El eslogan,
- Los tres grandes botones en una columna.



5. El menú de navegación

El menú de navegación utiliza, como cabía esperar, el elemento `<nav>`, que contiene una lista `` con todos los vínculos. Los tres vínculos de la derecha para las redes sociales se incluyen en otra lista ``.



Este es el código de la primera caja `<div class="wrapper">`:

```
<div class="wrapper">
  <nav>
    <ul id="menu">
      <li><a href="index.html">About</a></li>
      <li><a href="Courses.html">Courses</a></li>
      <li><a href="Programs.html">Programs</a></li>
      <li><a href="Teachers.html">Teachers</a></li>
      <li><a href="Admissions.html">Admissions</a></li>
      <li class="end"><a href="Contacts.html">Contacts</a></li>
    </ul>
  </nav>
  <ul id="icon">
```

```
<li><a href="#"></a></li>
<li><a href="#"></a></li>
<li><a href="#"></a></li>
</ul>
</div>
```

6. La continuación de la cabecera

A continuación, la cabecera presenta las típicas cajas `<div>`, un título `<h1>` y una lista ``:

```
<div class="wrapper">
  <h1><a href="index.html" id="logo">Learn Center</a></h1>
</div>
<div id="slogan">
  We Will Open The World<span>of knowledge for you!</span>
</div>
<ul class="banners">
  <li><a href="#"></a></li>
  <li><a href="#"></a></li>
  <li><a href="#"></a></li>
</ul>
```

7. Los artículos resaltados

Veamos con mayor detalle la estructura que permite visualizar los artículos resaltados en la parte superior de la zona central.

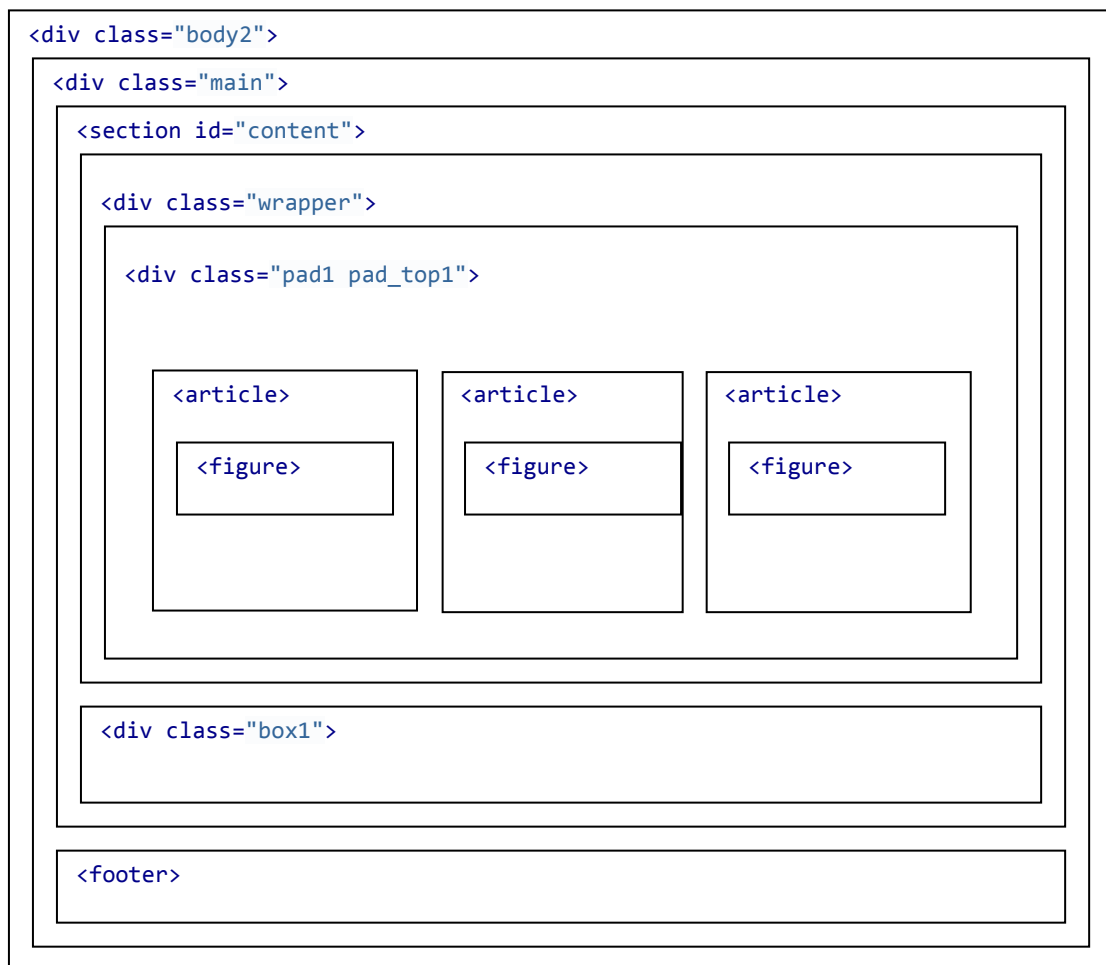


La zona central del sitio web es un elemento `<section id="content">` que se encuentra dentro de dos cajas `<div>`.

Los tres artículos resaltados en la parte superior de la zona central se visualizan lógicamente gracias a tres elementos `<article>`.

La imagen que ilustra cada artículo se ha insertado gracias al elemento `<figure>`.

Como puedes ver, esta plantilla utiliza correctamente los nuevos elementos de estructura del HTML5.



Este es el código que se ha usado para la presentación de los tres artículos resaltados:

```
<div class="body2">
  <div class="main">
    <section id="content">
      <div class="wrapper">
        <div class="pad 1 pad_top1">
          <article class="cols marg_right1">
            <figure><a href="#"></a></figure>
            <span class="font1">Our Mission Statement</span>
          </article>
          <article class="cols marg_right1">
            ...
          </article>
          <article class="cols">
            ...
          </article>
        </div>
      </div>
    </section>
  </div>
</div>
```

8. La continuación de la zona central

La continuación de la zona central sigue la misma lógica, con los elementos `<article>` y `<figure>`.

9. El pie de página

En el pie de página también se usan los elementos `<article>` para cada uno de los seis “bloques” de visualización. Probablemente sería más conveniente usar elementos `<section>`.

The image shows a footer template with four columns:

- Address:**
 - Country: USA
 - City: San Diego
 - Address: Beach st 54
 - Email: kent@freesite.com
- Join In:**
 - [Sign Up](#)
 - [Forum](#)
 - [Members](#)
 - [Login](#)
- Why Us:**
 - [Learn about us](#)
 - [Contact us](#)
 - [Services](#)
 - [Features](#)
- Newsletter:**
 - Input field
 - [Subscribe](#) button

At the bottom, there is a call to action: **Call Us Now: 1-800-567-8934** and a footer note: [Website Template](#) by [TemptationBlossom.com](#) / [2018](#) [Theme](#) provided by [TemptationBlossom.com](#)

11. METAINFORMACIÓN

Las páginas y documentos HTML incluyen más información de la que los usuarios ven en sus pantallas. Estos datos adicionales siempre están relacionados con la propia página, por lo que se denominan *metainformación* o *metadatos*. La metainformación siempre se incluye en la sección de la cabecera, es decir, dentro de la etiqueta `<head>`.

Aunque la metainformación más conocida y utilizada es el título de la propia página, se puede incluir mucha otra información útil para los navegadores y para los buscadores. En los próximos apartados se explica cómo incluir la metainformación.

11.1. Estructura de la cabecera

Como ya se explicó anteriormente, las páginas XHTML se dividen en dos partes denominadas cabecera y cuerpo. La sección de la cabecera está formada por todas las etiquetas encerradas por la etiqueta `<head>`:

<head>	Cabecera
Atributos comunes	i18n
Atributos específicos	<code>profile = "url"</code> - Especifica la URL del perfil o perfiles que utilizan los metadatos. No soportado en HTML5 <code>lang = "codigo_de_idioma"</code> - Especifica el idioma principal de los contenidos de la página
Tipo de elemento	-
Descripción	Define la cabecera del documento HTML

La cabecera típica de una página HTML completa presenta la siguiente estructura:

```
<head>
  <!-- Zona de etiquetas META -->
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
/>

  <!-- Zona de título -->
  <title>El título del documento</title>

  <!-- Zona de recursos enlazados (CSS, RSS, JavaScript) -->
  <link rel="stylesheet" href="#" type="text/css" media="screen" />
  <link rel="stylesheet" href="#" type="text/css" media="print" />
```

```
<link rel="alternate" type="application/rss+xml" title="RSS 2.0"
href="#" />

<script src="#" type="text/javascript"></script>
</head>
```

La etiqueta `<title>` establece el título de la página. Los navegadores muestran este título como título de la propia ventana del navegador. Los buscadores utilizan este título como título de sus resultados de búsqueda.

Por tanto, el valor de `<title>` no sólo es importante para los usuarios, sino que también es importante para que los usuarios encuentren las páginas a través de los buscadores. Un error común de muchos sitios web consiste en mostrar un mismo título genérico en todas sus páginas. Cada página debe mostrar un título corto, adecuado, único y que describa inequívocamente los contenidos de la página.

Las páginas HTML deben tener definido un título y sólo uno, por lo que todas las páginas web deben incluir obligatoriamente una etiqueta `<title>`, cuya definición formal se muestra a continuación:

<title>	Título del documento
Atributos comunes	i18n
Atributos específicos	<code>lang = "codigo_de_idioma"</code> - Especifica el idioma principal del título de la página
Tipo de elemento	-
Descripción	Define el título del documento HTML

Por último, la etiqueta `<head>` permite definir en el atributo `profile` la URL de un documento externo que contiene el perfil que siguen los metadatos de la cabecera. Los blogs creados con el programa WordPress incluyen por ejemplo el siguiente perfil en su cabecera:

```
<head profile="http://gmpg.org/xfn/11">
...
</head>
```

El documento <http://gmpg.org/xfn/11> es un perfil que define atributos adicionales para establecer la relación entre sitios web.

11.2. Metadatos

Una de las partes más importantes de la metainformación de la página son los metadatos, que permiten incluir cualquier información relevante sobre la propia página.

La especificación oficial de HTML no define la lista de metadatos que se pueden incluir, por lo que las páginas tienen libertad absoluta para definir los metadatos que consideren adecuados. La etiqueta empleada para la definición de los metadatos es `<meta>`.

<meta>	Metadatos
Atributos comunes	i18n
Atributos específicos	<p><code>name</code> = "texto" - El nombre de la propiedad que se define (no existe una lista oficial de propiedades)</p> <p><code>content</code> = "texto" - El valor de la propiedad definida (no existe una lista de valores permitidos)</p> <p><code>http-equiv</code> = "texto" - En ocasiones, reemplaza al atributo "name" y lo emplean los servidores para adaptar sus respuestas al documento</p> <p><code>scheme</code> = "texto" - Indica el esquema que se debe emplear para interpretar el valor de la propiedad. No soportado en HTML5</p> <p><code>charset</code> = "conjunto_caracteres" – Especifica la codificación de caracteres para el documento HTML.</p>
Tipo de elemento	-
Descripción	Permite definir el valor de los metadatos que forman la metainformación del documento

Los metadatos habituales utilizan solamente los atributos `name` y `content` para definir el nombre y el valor del metadato:

```
<meta name="autor" content="Juan Pérez" />
```

No obstante, algunas etiquetas `<meta>` muy utilizadas hacen uso del atributo `http-equiv`. Este atributo se utiliza para indicar que el valor establecido por este metadato puede ser utilizado por el servidor al entregar la página al navegador del usuario. El siguiente metadato indica al servidor que el contenido de la página es código HTML y su codificación de caracteres es UTF-8:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

El atributo `schema` no suele utilizarse, aunque permite proporcionar información de contexto para que el navegador interprete correctamente el valor del metadato. En el siguiente ejemplo, el atributo `schema` indica al navegador que el valor del metadato hace referencia al código ISBN:

```
<meta schema="ISBN" name="identificador" content="789-1392349610">
```

Aunque no existe una lista oficial con los metadatos que se pueden definir, algunos de ellos se utilizan en tantas páginas que se han convertido prácticamente en un estándar. A continuación se muestran los metadatos más utilizados:

Definir el autor de la página:

```
<meta name="author" content="Juan Pérez" />
```

Definir el programa con el que se ha creado el documento:

```
<meta name="generator" content="WordPress 2.8.4" />
```

Definir la codificación de caracteres del documento:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
```

En HTML5 se ha dado prioridad a la simplicidad:

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

Disponemos así de lo estrictamente necesario para una correcta gestión por parte de los navegadores web.

Definir el copyright del documento:

```
<meta name="copyright" content="diw.es" />
```

Definir el comportamiento de los buscadores:

```
<meta name="robots" content="index, follow" />
```

Definir las palabras clave que definen el contenido del documento para los motores de búsqueda:

```
<meta name="keywords" content="diseño, css, hojas de estilos, web, html" />
```

Definir una breve descripción del sitio:

```
<meta name="description" content="Artículos sobre diseño web,
usabilidad y accesibilidad" />
```

Refrescar la página cada 30 segundos:

```
<meta http-equiv="refresh" content="30" />
```

La etiqueta que define la codificación de los caracteres (`http-equiv="Content-Type"`) se emplea prácticamente en todas las páginas y las etiquetas que definen la descripción (`description`) y las palabras clave (`keywords`) también son muy utilizadas.

12. OTRAS ETIQUETAS IMPORTANTES

12.1. Comentarios

Al igual que la mayoría de lenguajes de marcado, HTML permite incluir comentarios dentro de su código para añadir información que no se debe mostrar por pantalla.

Normalmente, los diseñadores y programadores incluyen comentarios para marcar el comienzo y el final de las secciones de las páginas, para incluir avisos y notas para otros diseñadores o para incluir explicaciones sobre la forma en la que se ha creado el código HTML.

Aunque los comentarios no se muestran por pantalla y por tanto son *invisibles* para los usuarios, sí que se descargan con el código HTML de la página. Por este motivo, nunca debe incluirse información sensible o confidencial en los comentarios.

La sintaxis de los comentarios es la siguiente:

- Apertura del comentario: `<!--`
- Contenido del comentario: `(cualquier texto)`
- Cierre del comentario: `-->`

El siguiente ejemplo muestra el uso de los comentarios HTML para indicar el comienzo y final de cada sección. Recuerda que los comentarios no se muestran por pantalla y que no influyen en la forma en la que se ven las páginas:

```
<!-- Inicio del menú -->
<div id="menu">
<ul>
  <li>...</li>
  <li>...</li>
```

```
<li>...</li>
<li>...</li>
</ul>
<!-- Fin del menú -->

<!-- Inicio de la publicidad -->
<div id="publicidad"> ... </div>
<!-- Fin de la publicidad -->
```

Los comentarios de HTML pueden ocupar tantas líneas como sea necesario. Sin embargo, los comentarios no se pueden anidar, es decir, no se puede incluir un comentario dentro de otro comentario.

12.2. JavaScript

Como ya se explicó en los apartados anteriores, la etiqueta `<script>` se utiliza para enlazar archivos JavaScript externos y para incluir bloques de código JavaScript en las páginas. Sin embargo, algunos navegadores no disponen de soporte completo de JavaScript, otros navegadores permiten bloquearlo parcialmente e incluso algunos usuarios bloquean completamente el uso de JavaScript porque creen que así navegan de forma más segura.

Si JavaScript está bloqueado o deshabilitado y la página web requiere su uso para un correcto funcionamiento, es habitual incluir un mensaje de aviso al usuario indicándole que debería activar JavaScript para disfrutar completamente de la página.

El siguiente ejemplo muestra una misma página web que requiere JavaScript tanto cuando se accede con JavaScript activado y como cuando se accede con JavaScript completamente desactivado.

Imagen de www.netvibes.com con JavaScript activado



Figura 2.87. Ejemplo de página compleja con JavaScript activado

Imagen de www.netvibes.com con JavaScript deshabilitado

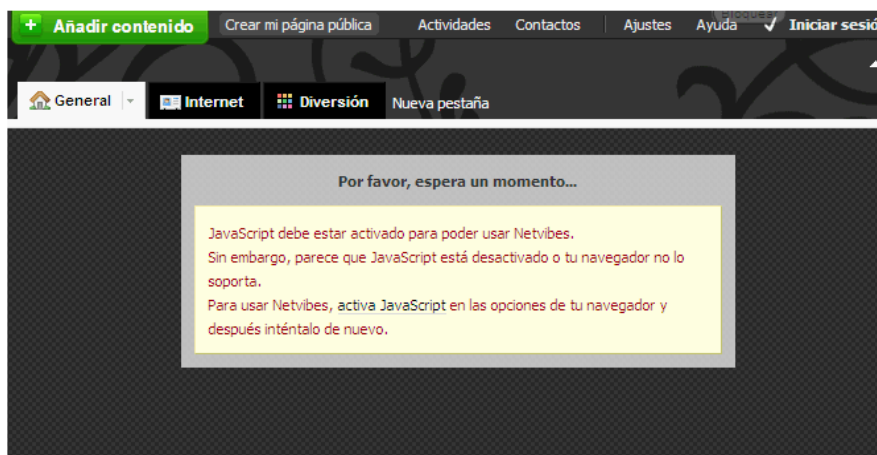


Figura 2.88. Ejemplo de página compleja con JavaScript desactivado

HTML define la etiqueta `<noscript>` para incluir un mensaje que los navegadores muestran cuando JavaScript se encuentra bloqueado o deshabilitado.

<code><noscript></code>	Sin soporte de scripts
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Define un mensaje alternativo que se muestra al usuario cuando su navegador no soporta la ejecución de scripts

De esta forma, incluir un mensaje de aviso que solamente sea visible en los navegadores que tienen bloqueado JavaScript es tan sencillo como incluir la etiqueta `<noscript>` dentro del `<body>`.

```
<head> ... </head>
<body>
<noscript>
  <p>Bienvenido a Mi Sitio</p>
  <p>La página que estás viendo requiere para su funcionamiento
el uso de JavaScript.
Si lo has deshabilitado intencionadamente, por favor vuelve a
activarlo.</p>
</noscript>
</body>
```

12.3. CSS

Algunos de los atributos más utilizados en la creación de páginas web son `id`, `class` y `style`. Los tres atributos están muy relacionados con CSS, sobre todo `class` y `style`.

El atributo `id` se emplea para asignar un identificador único a cada elemento de la página, lo que es útil tanto para aplicar estilos CSS a ese elemento como para programar aplicaciones con JavaScript.

Por otra parte, el atributo `class` se emplea para definir la clase CSS que se aplica a un elemento. La clase CSS es el nombre de un conjunto de estilos que se definen en la hoja de estilos y que se quieren aplicar a un elemento:

```
<p class="resumen">Lorem ipsum dolor sit amet, consectetur  
adipiscing elit.  
Maecenas at diam id enim viverra semper. Nulla id urna. Donec  
sodales.</p>
```

El párrafo del ejemplo anterior se muestra por pantalla con el aspecto definido por el conjunto de estilos llamado `resumen` y que se define en la hoja de estilos CSS enlazada por la página web.

El atributo `style` se emplea para definir estilos CSS directamente sobre los elementos HTML, tal y como se muestra en el siguiente ejemplo:

```
<p>Algunas palabras de esta frase se muestran de <span  
style="color:red">color rojo</span></p>
```

No se debe confundir el atributo `style` con la etiqueta `<style>` que se explicó anteriormente. La etiqueta `<style>` se utiliza para incluir bloques de código CSS:

```
<head>  
...  
<style type="text/css">  
  span {color:red;}  
</style>  
</head>
```

12.5. Otras etiquetas

La etiqueta `<address>` es una de las etiquetas más desconocidas de HTML, por lo que uso no está muy extendido. La etiqueta `<address>` se utiliza para proporcionar información de contacto.

<address>	Direcciones
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Define la información de contacto de un documento

El siguiente ejemplo sencillo muestra directamente el nombre, dirección y teléfono de contacto de una empresa:

```
<address>
  Nombre de la empresa
  Dirección completa
  Teléfono y Fax
</address>
```

La especificación oficial de HTML muestra un ejemplo complejo del uso de la etiqueta `<address>`:

```
<address>
  <a href="../People/Raggett/">Dave Raggett</a>,
  <a href="../People/Arnaud/">Arnaud Le Hors</a>,
  contact persons for the <a href="Activity">W3C HTML
  Activity</a><br/>
  $Date: 1999/12/24 23:37:50 $
</address>
```

Hasta hace unos años, la etiqueta `<hr>` era una de las más utilizadas, ya que permite mostrar una línea horizontal de separación. Sin embargo, hoy en día apenas se utiliza, ya que se considera un elemento puramente estético, del que no debería preocuparse HTML y para el que CSS ofrece alternativas mucho mejores.

<hr>	Línea horizontal
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Permite incluir una línea horizontal de separación

La siguiente imagen muestra el aspecto con el que los navegadores muestran por defecto las líneas horizontales creadas con `<hr>`:

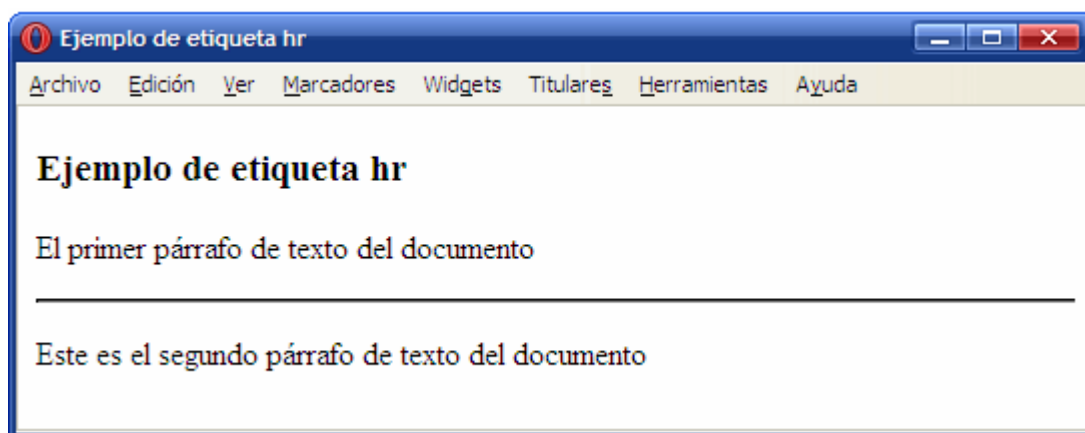


Figura 2.89. Ejemplo de uso de la etiqueta hr

El código HTML del ejemplo anterior se muestra a continuación:

```
<html>
<head><title>Ejemplo de etiqueta hr</title></head>

<body>
<h3>Ejemplo de etiqueta hr</h3>

<p>El primer párrafo de texto del documento</p>

<hr/>

<p>Este es el segundo párrafo de texto del documento</p>
</body>
</html>
```

13. COMPROBACIÓN DE ERRORES EN LAS PÁGINAS HTML

Incluso un diseñador web con las mejores intenciones puede entrar en conflicto con las estrictas reglas HTML. Aunque los navegadores deberían detectar estos fallos, casi ninguno lo logra. En su lugar, hacen todo lo posible por ignorarlos y mostrar documentos erróneos. A primera vista, parece un buen diseño, ya que después de todo suaviza cualquier desliz que haya podido tener. Pero hay un lado oscuro en la tolerancia a errores. En particular, este comportamiento hace que fallos graves pasen inadvertidos en las páginas Web. ¿Qué es un error grave? Un problema que es inofensivo cuando se visualiza la página en nuestro navegador favorito pero que tiene un aspecto vergonzoso cuando alguien ve el código en otro navegador; un error que pasa sin ser detectado hasta que se edita el código, mostrando el fallo la siguiente vez

que se enseña la página; o un problema que no tiene efecto sobre la visualización de la página pero impide que una herramienta automática (como un motor de búsqueda) la lea.

Afortunadamente, hay una forma fácil de detectar inconvenientes como estos. Es posible utilizar una herramienta de validación que lea la página Web y compruebe si cumple las estrictas reglas HTML. Si empleamos una herramienta de diseño Web profesional como Dreamweaver, podemos recurrir a su comprobador de errores integrado. Si realizamos las páginas a mano en un editor de texto, podemos usar una herramienta de validación en línea gratuita. Veamos algunos problemas que puede detectar un validador:

- Omisión de elementos obligatorios (por ejemplo, el `<title>`).
- Etiqueta de inicio sin etiqueta final.
- Etiquetas incorrectamente anidadas.
- Etiquetas a las que faltan atributos (por ejemplo, una `` sin el atributo `src`).
- Elementos o contenido mal colocados (por ejemplo, un texto puesto directamente en la sección `<body>`).

Puedes escoger entre muchas herramientas de validación en línea. A continuación, vamos a ver cómo usar el validador que ofrece la organización de estándares W3C:

1. Comprueba que tu documento tiene la definición de tipo de documento correcta. Ésta le indica al validador las reglas a usar para certificar el documento. Por ejemplo, si empleamos HTML5, el validador verifica si cumple las reglas de HTML5 y permite determinadas funciones no aceptadas en el antiguo estándar XHTML 1.0.

Nota: Como el estándar HTML5 sigue evolucionando, los validadores HTML5 no son tan completos como los XHTML.

2. En tu navegador Web, introduce <http://validator.w3.org/> (Ver fig. 2.90). El validador W3C ofrece tres opciones, en fichas independientes: validar por URI (páginas ya en línea), validar por archivo (para páginas almacenadas en un archivo de su

equipo) y validar por entrada directa (para un marcado introducido directamente en el cuadro que se ofrece).



Figura 2.90. Validador del W3C

3. Haz clic en la ficha adecuada e introduce el contenido HTML. Al validar por URI puedes examinar una página Web existente. Basta con introducir la URL de la página Web en el cuadro *Address* (Dirección). La validación por archivo permite transferir un fichero desde tu equipo. Primero, haz clic en el botón *Examinar* (*Seleccionar archivo* en Chrome) para abrir un cuadro de diálogo *Abrir* estándar. Selecciona la ubicación del archivo HTML y pulsa en *Abrir*.

La validación por entrada directa permite certificar cualquier marcado; solo hay que introducirlo en el cuadro correspondiente. La manera más sencilla de usar esa opción consiste en copiarlo desde tu editor de texto y pegarlo en el cuadro de la página de validación del W3C. Antes de continuar, puedes hacer clic en *More*

Options (Más opciones) para ver otras posibilidades pero no es recomendable. Conviene que el validador detecte automáticamente el tipo de documento; de ese modo, usará la DTD especificada en tu página Web. Asimismo, usa la detección automática para el conjunto de caracteres a menos que tu página HTML no esté en inglés y el validador tenga dificultades para establecer el conjunto de caracteres correcto.

4. Pulsa el botón *Check* (Comprobar). Al hacerlo, se envía la página HTML al servidor W3C y, tras una pausa, aparece el informe. Verás si el documento pasó la comprobación y, si falló, qué errores detectó el validador.

El validador puede mostrar diversas advertencias en documentos HTML totalmente válidos, incluidas las de determinación automática del conjunto de caracteres y del carácter experimental del servicio de validación HTML5.

14. CARACTERÍSTICAS HTML5 QUE SOPORTA CADA NAVEGADOR

Para saber qué características de HTML5 soporta el navegador que estamos utilizando o cualquier otro navegador utilizado sobre cualquier dispositivo (PC, tablet, smartphone, televisión,...) e incluso comparar lo que soportan varios navegadores podemos utilizar el siguiente sitio web: <http://html5test.com/>

La calificación del test HTML5 es una indicación de la compatibilidad de tu navegador con las próximas especificaciones del estándar HTML5 y afines. A pesar de que la especificación no está finalizada aún, todos los principales fabricantes de navegadores se están asegurando de que su navegador esté listo para el futuro. Sirve para averiguar qué partes de HTML5 son soportados por los navegadores actuales y comparar los resultados con otros navegadores.

The HTML5 TEST - HOW WELL DOES YOUR BROWSER SUPPORT HTML5?

your browser | other browsers | compare | news | about the test

your browser scores

338

AND 10 BONUS POINTS

out of a total of 500 points

You are using Firefox 15.0.1 on Windows XP Correct? ✓ X

Parsing rules +2 bonus points **10**

<!DOCTYPE html> triggers standards mode	Yes ✓
HTML5 tokenizer	Yes ✓
HTML5 tree building	Yes ✓
HTML5 defines rules for embedding SVG and MathML inside a regular HTML document. Support for SVG and MathML is not required though, so bonus points are awarded if your browser supports embedding these two technologies.	
SVG in text/html	Yes ✓
MathML in text/html	Yes ✓

Canvas **20**

canvas element	Yes ✓
----------------	-------

7,492 11k 3.2k

Tweet Like +1

The HTML5 test score is an indication of how well your browser supports the upcoming HTML5 standard and related specifications. Even though the specification isn't finalized yet, all major browser manufacturers are making sure their browser is ready for the future. Find out which parts of HTML5 are already supported by your browser today and compare the results with other browsers.

HTML5

SPONSORS

MOBILE GAME ARCH

jq.Mobi

SUPER FAST HTML5 FOR MOBILE

Figura 2.91. Imagen del sitio htm5test.com