UNIDAD DIDÁCTICA 2

UTILIZACIÓN DE LENGUAJE DE MARCAS EN ENTORNO WEB

MÓDULO PROFESIONAL: LENGUAJE DE MARCAS Y SISTEMAS DE GESTIÓN DE LA INFORMACIÓN

CESUR
Tu Centro Oficial de FP

Índice

RESUMEN INTRODUCTORIO

En la presente unidad se estudiará la utilización del lenguaje de marcas en el entorno web, conociendo los estándares web y aspectos importantes del lenguaje HTML.

INTRODUCCIÓN

La importancia del mundo web en una sociedad altamente tecnológica como la actual, hace necesario ser consciente y tener conocimientos sobre las distintas tecnologías y lenguajes de programación que permiten, sobre todo, a través de internet, ofrecer una gran cantidad de servicios web.

Si nos centramos en los lenguajes de marcas en entornos web, existe uno de ellos que juega un papel fundamental en el mundo web, y ese es HTML, con unos orígenes que se remontan a los años 80, y alcanza nuestros días con una sucesión de versiones hasta el ahora HTML 5.

HTML ha servido de base para otros tipos de tecnologías que han ido aportando a las páginas o sitios web dinamismo, conexión o comunicación, incluso tratamiento de datos mediante lenguajes como JavaScript, AJAX, etc., y bases de datos de tipo MySql, por ejemplo.

Esta evolución en los lenguajes, como es el caso de los de tipo script de lado cliente, han requerido que los desarrolladores deban conocer a la perfección su estructura, sintaxis, elementos, etc., con el fin de optimizar los desarrollos y evitar cometer errores a la hora de implementar nuevos desarrollos, así como de incorporar en el código hojas de estilo (CSS) o funcionalidades como la sindicación de contenidos.

La aparición de herramientas que ayudan a la escritura de código fuente supuso un plus de cara a la optimización de recursos personales en la creación de aplicaciones o sitios web, facilitando el esfuerzo por parte de las personas encargadas de su desarrollo.

CASO INTRODUCTORIO

Estás inmerso en el desarrollo de la web para la zapatería que te han encargado recientemente. Comienzas creando una página con las correspondientes etiquetas, elementos y atributos, para posteriormente, usarla como plantilla para el resto de la web, copiándola y modificándola para cada página concreta. Sin embargo, te has dado cuenta de que, si varías algún aspecto del diseño, deberás modificarlo en cada una de las páginas. Por ello, decides agrupar todos los estilos en una hoja de estilos común a todas las páginas de la web. De esta forma, podrás retocar cualquier aspecto de las páginas de una sola vez y sin necesidad de tocar el código HTML.

Al finalizar el estudio de la unidad, conocerás la estructura de un documento web, manejarás las diferentes etiquetas y atributos, sabrás definir estilos mediante una hoja de estilos y conocerás las diferentes formas de usar CSS en una página web.

1. ESTÁNDARES WEB. VERSIONES. CLASIFICACIÓN

En el desarrollo de la página web en la que estás trabajando, será importante aplicar los distintos estándares web que existen para su utilización que establece la W3C, teniendo en cuenta buscar las versiones más actualizadas y hacer una clasificación de los mismos de acuerdo a las necesidades planteadas.

Los estándares web corresponden a las diferentes tecnologías que podemos utilizar para la creación de sitios web, son creados por organismos o instituciones para que sirvan de referencia en el desarrollo web. Como ejemplos tenemos W3C, WHATWG, ECMA o Khronos.

Estándares web.
Fuente: https://www.inesem.es/revistadigital/informatica-y-tics/estandares-web/

Quizá, la más conocida es W3C (World Wide Web Consortium), es un consorcio internacional de organizaciones que están relacionadas con las tecnologías de la información, y facilita la estandarización mediante la publicación de una serie de normas a seguir.

Destacan por su carácter abierto y gratuito, siendo accesible para todo tipo de usuarios, además de iniciarse una labor de integración con cualquier tipo de navegador permitiendo su compatibilidad y simplificando el trabajo de los desarrolladores.

Las ventajas que ofrece el uso de estándares en el desarrollo web son muy numerosas, entre ellas, podemos destacar las siguientes:

1. Mayor sencillez en el código fuente.
2. Menor tiempo efectivo en el desarrollo y mantenimiento web.
3. Gratuidad y acceso a todos los usuarios.
4. Desarrollar webs más accesibles.
5. Gran compatibilidad con aplicaciones, herramientas, navegadores, etc.
6. Optimización de motores de búsqueda, ayudando al mantenimiento de un mejor SEO.

La correcta utilización de los distintos estándares web se comprobará mediante la validación correspondiente de las páginas web, donde atendiendo a lo marcado por W3C, se debe prestar especial atención a cuestiones como:

- Accesibilidad web.
  En el sentido de mirar por los usuarios para un más fácil acceso y comprensión del contenido web mediante la utilización de atributos ALT (descripciones), facilidad de navegación o corrección en lo menús. Además de controlar la velocidad con la que se cargan de modo correcto los contenidos, como mejora de experiencia de navegación de cara al usuario final.
- Corrección de código.
  Referido a los diferentes protocolos web y código que deben ser válidos en la escritura del código fuente de cualquiera de los estándares web: HTML, CSS, XML o JavaScript.
- Exactitud y presentación de contenidos.
  Se deben tener en cuenta dos aspectos a nivel contenido, uno referido a su calidad y, otro, referido al aspecto que presenta como su color, tipografía, imágenes, etc.

Versiones y clasificación.

De acuerdo con los estándares web W3C, encontramos entre los más populares los siguientes:

- HTML (Hyper Text Markup Language), con el que se definía la estructura de los documentos. Es un lenguaje de marcado que ofrece la posibilidad de marcar contenido a nivel semántico, lo que le otorga un significado y además estructura. Es un lenguaje que sirve de base para el desarrollo de otros lenguajes.
  Partía de la versión 2.0, evolucionando y añadiendo funcionalidades hasta llegar a la versión actual conocida como HTML5.

- XML (eXtensible Markup Language), muy utilizado en mayoría de tecnologías, basado en SGML permite trabajar con documentos de gran tamaño y soporta bases de datos, comunicación entre aplicaciones y la integración de información de manera efectiva.
  Su primera versión es XML 1.0 que ha ido recibiendo pequeñas actualizaciones hasta su quinta edición que continúa siendo utilizada por la mayoría de las industrias para compartir información estructurada.

- CSS (Cascading Style Sheets), conocidas como hojas de estilo en cascada que permiten asignar estilos a los documentos, en concreto al HTML, como dar colores al texto, fondos, etc.
  En su evolución de versiones, en la actualidad nos encontramos con CSS3, combinada con HTML5, es la más utilizada por la mayoría de los desarrolladores web.

- JavaScript, utilizado para hacer las web más dinámicas y funcionales, como pueden ser cambios de estilo de modo dinámico o actualizaciones proporcionadas desde un servidor, actuando siempre dentro del párrafo donde se referencia.
  La especificación donde se muestran los detalles del funcionamiento y comportamiento de JavaScript en los diferentes navegadores es la ECMAScript, donde se define el desarrollo de los motores de búsqueda de JavaScript para cualquier tipo de código, programa o aplicación. Con actualizaciones prácticamente anuales, en el año 2023 ha aparecido su 14ª edición.

ENLACE DE INTERÉS

Accede a la página oficial de W3C, donde consultar noticias, eventos, novedades, estándares, etc.

2. ESTRUCTURA DE UN DOCUMENTO HTML

Tras la recopilación de estándares web que debe cumplir la página que estás creando, deberás ponerte con el desarrollo del código con la creación del documento HTML de la página web que recogerá el código fuente, donde se respetará la estructura el documento HTML para evitar errores de sintaxis y de escritura de etiquetas o sentencias.

Las marcas son señales que se colocan entre el texto, añadiéndole significado. Habitualmente, estas marcas o etiquetas consisten en un nombre encerrado entre signos de menor y mayor. Además, suelen aparecer por parejas, de manera que la primera abre un contexto y la segunda lo cierra, afectando a todo lo que queda dentro de dicho contexto.

El nombre que se le haya dado a la etiqueta será el que le dé un significado u otro. Por ejemplo, ** ** será indicativo de párrafo y **** significará negrita. Además, se permite el anidamiento de etiquetas, es decir, pueden aparecer unas dentro del contexto limitado por otras.

A estas mismas etiquetas, se les puede añadir una barra (/) tras el símbolo de menor, convirtiéndolas así en etiquetas de cierre. En los ejemplos anteriores, serían ** ** y ****. A partir de estas etiquetas de cierre, las correspondientes etiquetas de apertura pierden vigencia, dejando así de afectar al texto:

Esto aparecerá en negrita. Esto ya no.

Un archivo HTML estará compuesto por un conjunto de etiquetas, que se utilizan para definir la forma o estilo que se quiere aplicar a cada parte del documento. Será el navegador el que se encargue de interpretar todas estas etiquetas y mostrar el contenido con el formato definido por ellas.

Es importante tener en cuenta que HTML no es un lenguaje de programación, ya que su función no es la de crear un programa, ni cuenta con las estructuras básicas de los lenguajes de programación. Por tanto, no se debe usar la palabra programar para referirse a la creación de páginas web. No obstante, es posible combinar HTML con otros lenguajes como javascript, que sí es un lenguaje de programación.

2.1 Partes de un documento html

Todo documento HTML ha de estar delimitado por las etiquetas **<html>** y **</html>.** Dentro de este documento, se pueden asimismo distinguir dos partes principales:

- La cabecera, delimitada por **<head>** y **</head>**, donde se colocarán etiquetas de índole informativo como, por ejemplo, el título de nuestra página.
- El cuerpo, flanqueado por las etiquetas **<body>** y **</body>**, que será donde se coloque todo el contenido de la página.

El resultado es un documento con la siguiente estructura:

```
<html>
<head>
    <title>Cesur</title>
    <! --Etiquetas y contenidos de la cabecera.
    Datos que no aparecen en la página pero que son importantes
    para catalogarla: Titulo, palabras clave, ...-->
</head>
<body>
  <! -- Etiquetas y contenidos del cuerpo.
  Parte del documento que será mostrada por el navegador: Texto e
  imágenes-->
</body>
</html>
```

Este ejemplo, puede servir también para ver cómo incluir comentarios en el archivo HTML. Tan sólo se trata de escribir el texto que interese entre las etiquetas <!-- y -->. Su función es la de poder añadir texto que no será interpretado por los navegadores, y por tanto no aparecerá por pantalla.

HTML ha ido evolucionando con el tiempo, dando lugar a diferentes versiones, cada una con soporte para unas u otras etiquetas, si bien, la estructura que se acaba de presentar es común para todas. No obstante, resulta importante para el navegador conocer qué versión de HTML usa la página que está leyendo, ya que ello podrá determinar si la sintaxis del documento es válida o no. Para eso, se debe usar el elemento DOCTYPE.

Las declaraciones DOCTYPE no son más que enlaces a documentos de definición (DTD), que se estudiarán más adelante, cuyo cometido es definir los elementos válidos del lenguaje y la relación entre ellos. Deben aparecer al principio del documento, antes de su estructura, con la sintaxis siguiente: <!DOCTYPE …>

Para la versión 4 de HTML, que ha sido la que ha tenido mayor vigencia en el tiempo y la más extendida hasta la aparición de la versión 5, se pueden encontrar las siguientes declaraciones DOCTYPE:

- Estricta: se trata de la declaración por defecto, que no permite usar elementos del lenguaje desfasados.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN" "http://www.w3.org/TR/REC-html40/strict.dtd">

- Transitional: para facilitar la transición entre versiones anteriores y esta, permite usar elementos obsoletos.

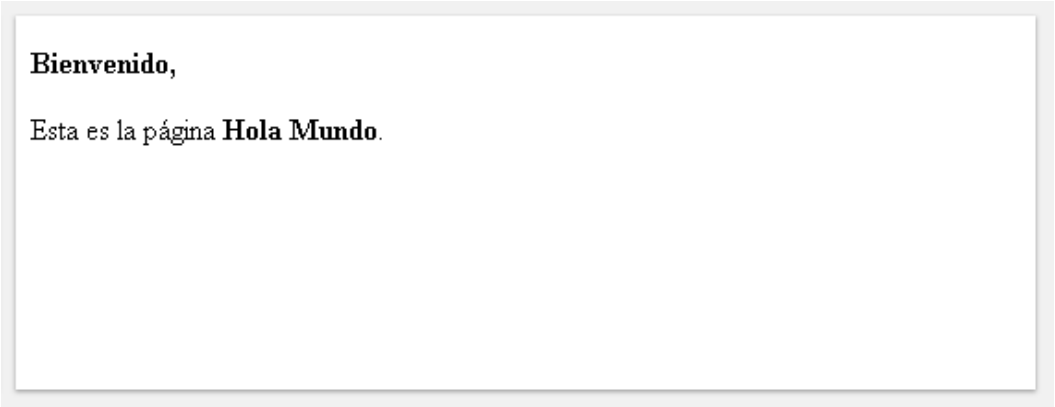
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">

- Frameset: una variante de transitional pero para usar páginas con marcos.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN" "http://www.w3.org/TR/REC-html40/frameset.dtd">

A continuación, se muestra cómo quedaría una página, con estos primeros conceptos que se han presentado:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<html>
<head>
    <title>Hola Mundo</title>
</head>
<body>
    <p>
        <b>Bienvenido, </b>
    </p>
    <p>
        Esta es la página <b>Hola Mundo</b>.
    </p>
</body>
</html>
```



2.2 Evolución de HTML

Aunque HTML nace en 1991, el primer estándar oficial aparece en 1995 con el nombre de HTML 2.0. No obstante, ni este ni el posterior estándar aparecido en 1997 (HTML 3.2) tuvieron tanta importancia como HTML 4.0, aparecido en 1998 y, su actualización, HTML 4.01 (1999).

En el año 2000, W3C (World Wide Web Consortium), el organismo encargado de elaborar estos estándares, publica XHTML 1.0, que en realidad se trata de una adaptación de la versión 4 de HTML a la especificación XML, de forma que cumpla con unas especificaciones más estrictas y permita la extensión del lenguaje.

Desde ese momento, tanto HTML 4 como XHTML han sido los estándares más ampliamente utilizados para la creación de páginas web hasta la aparición de la versión 5.

Por supuesto, el elemento DOCTYPE cambia para XHTML, escribiéndose de la siguiente manera, según cada uno de los 3 tipos presentados anteriormente:

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

Además, es importante tener en cuenta que en XHTML es necesario especificar también un espacio de nombres para que sea válido, por lo que la etiqueta <html>, se debe escribir de la siguiente forma:
<html xmlns="http://www.w3.org/1999/xhtml">

Posteriormente, la W3C trató de seguir por el camino iniciado con XHTML, abandonando HTML, hasta el punto de desarrollar una segunda versión de XHTML que no era compatible con HTML ni tan siquiera con la primera versión de XHTML. La falta de aceptación por parte de la industria y de los usuarios, interesados en mantener la compatibilidad con las versiones anteriores, hizo que se retomara el desarrollo de HTML y se abandonara el camino seguido por XHTML 2.0. Aunque en 2008 se redactó el primer borrador de HTML5, no fue hasta 2014 cuando se publicó su primera versión definitiva, lo cual dificultó su implantación. La última versión del estándar es la 5.2, aparecida en 2017.

HTML5 abandona los distintos tipos de DOCTYPE, quedándose únicamente en:
<!DOCTYPE html>

HTML 2.0	Año 1995. Es el primer estándar oficial de HTML, muy limitado, no admitía la utilización de tablas. Su sencillez de estructura se limitaba al uso de los elementos en el body, dejando en opción utilizar las etiquetas HTML y head.
HTML 3.2	Año 1997. Incorpora las recomendaciones de la W3C, además de mejoras como el uso de applets de Java y textos que podían acompañar a imágenes.
HTML 4.01	Año 1999. A partir de donde se centró el desarrollo en XHTML, dando lugar a un desarrollo paralelo por parte de grandes empresas para retomar HTML, comenzando con la versión HMTL 5, motivo que llevó a W3C a retomar el desarrollo de HTML e integrar en él XHTML.
HTML 5, 5.1 y 5.2	Año 2014. W3C libera HTML 5, que incorpora entre otras novedades, nuevas etiquetas, compatible con SVG (canvas y gráficos vectoriales), utilización de bases de datos SQL, ejecución separada en hilos de JavaScript y la interfaz del navegador o la integración directa de audio y video. Apareciendo nuevas versiones.
XHTML 1.0	Similar a HTML4, es una reformulación como aplicación de XML, teniendo compatibilidad limitada con versiones de HTML anteriores.
XHTML 1.1	Pierde la compatibilidad con versiones anteriores de HTML, a pesar de ser una versión modular de XHTML que permite su subdivisión y extensión, de cara a nuevas plataformas.

Resumen de versiones de HTML.

Durante los primeros años en la evolución de HTML, surge uno de sus principales problemas. La web comenzó a crecer muy rápidamente, y con ello, las necesidades de los usuarios, que fueron evolucionando a mayor velocidad que los estándares, por lo que los desarrolladores de los navegadores fueron creando sus propias extensiones del lenguaje para conseguir implementar las características que los usuarios demandaban. De esta forma, llegó un momento en que se encontraban diversas formas no estándar de conseguir un mismo resultado.

Otro problema, ha sido la distinta interpretación que cada navegador ha hecho de diversos aspectos de la especificación, como las medidas relativas o la asignación de diferentes valores por defecto. De igual forma, algunos navegadores son más flexibles en la aplicación de los estándares y admiten código no correcto que en otros navegadores es ignorado.

El resultado de toda esta problemática es que muchas páginas no se ven igual en un navegador u otro, lo que obliga a los desarrolladores a implementar diferentes soluciones en función del navegador que lea la página. HTML5 soluciona en gran medida estos problemas, si bien, la coexistencia con versiones anteriores y con navegadores no totalmente compatibles, hace que en muchos casos estos aspectos aún deban ser tenidos en cuenta.

ENLACE DE INTERÉS

En este enlace podrá ver las distintas versiones de HTML y diferencias entre HTML4 y HTML5

Diferencias entre HTML y XHTML

Partiendo de la base de que XHTML está basado en XML, tendremos la garantía de las ventajas de un documento bien formado, lo que dota a XHTML de una gran robustez, a la vez de facilitar el procesamiento automático de los documentos creados en XHTML.

Entre las diferencias entre HTML y XHTML, podemos destacar lo que incorpora XHTML:

- Permite la separación de los datos y de la estructura, diferenciando entre lo que constituye el modelo de datos de lo que es realmente su presentación o vista.
- Permite la utilización de nombres en los que se incluyan espacios.
- Ofrece la posibilidad de añadir etiquetas a las heredadas de HTML.
- El cierre de los elementos es obligatorio, también los que estén vacíos.
- Si existen elementos anidados, deben estarlo en un orden correcto.
- Las etiquetas deben estar escritas en minúsculas.
- Distingue entre mayúsculas y minúsculas.

- Los atributos de los elementos deben ir entrecomillados.
- Es obligatorio que cada atributo cuente con un valor asignado.

XHTML en los sistemas de gestión de la información

Los sistemas de gestión de la información constan de una serie de elementos, tanto físicos como lógicos, datos y personas que explotan esa información, donde su objetivo final es la transformación, a través de una serie de procesos, de los datos en información para la obtención de un conocimiento relacionado con ellos.

En la ejecución de esos procesos se utilizan lenguajes de programación, como es el caso de XHTML, con el que se desarrollan programas que permiten el acceso a datos para la obtención de resultados, lo cuales pueden ser almacenados en archivos o imprimirse como informes.
La utilización de lenguajes tipo XHTML, han mejorado el escenario de trabajo al permitir la conexión con bases de datos, incorporando nuevas funcionalidades que permiten, por ejemplo, realizar consultas y actualizaciones de datos por distintos usuarios, relacionar archivos, mejorando la eficiencia y evitando redundancias.

XHTML, basado en XML, permite que, al utilizar bases de datos, se puedan integrar distintos sistemas de información, por un lado, los basados en archivos y, por otro, los basados en el uso de bases de datos de tipo relacional, incluso, en ocasiones, las bases de datos orientadas a objetos.

Sí deberá ser tenido en cuenta, a la hora de realizar un desarrollo, si el modelo a utilizar será XHTML, de manera que las bases de datos sean del tipo NoSQL, en vez del modelo SQL de las bases de datos relacionales, donde el lenguaje más empleado es SQL.

Características a tener en cuenta si vamos a utilizar XTML serían:

- La información es almacenada en uno o más documentos.
- Los elementos de datos en los documentos tendrán el mismo orden del propio documento.
- Los documentos contienen tanto los datos como la codificación.
- Se deben definir los esquemas de la información.
- Los datos del documento tienen una relación jerárquica.
- La recuperación de información se realiza mediante lenguajes específicos.
- Cuenta con APIs (interfaces de programación) del tipo Document Object Model (DOM).

RECUERDA

En los sistemas de gestión de información existen vínculos entre distintas áreas de la gestión humana y la innovación tecnológica.

3. IDENTIFICACIÓN DE ETIQUETAS Y ATRIBUTOS HTML

Dentro del proceso de desarrollo web que estás llevando a cabo, y a la hora de continuar confeccionando el documento HTML, es fundamental que la escritura del código fuente sea correcta a nivel sintáctico, sobre todo, cómo deben escribirse las distintas etiquetas y los atributos que las pueden acompañar o no.

Un archivo HTML no es más que un conjunto de elementos, cada uno con un nombre (etiqueta), un contenido y unas características (atributos). Al ser un lenguaje de marcas, son las etiquetas (tags) las que se usan para delimitar el contenido de estos elementos.

Las etiquetas pueden ser de 3 tipos:

- De apertura: <xxx>
- De cierre: </xxx>
- Autocerradas. Son como una combinación de las dos anteriores, para elementos que no tienen contenido: <xxx />

Aunque los nombres de las etiquetas se pueden escribir tanto en mayúsculas como en minúsculas, como convención se usa la escritura en minúsculas.

Por otro lado, cada uno de los elementos puede tener una serie de propiedades que se denominan atributos, y se declaran en la etiqueta de apertura de dicho elemento, asignándoles un valor entre comillas (simples o dobles), de la forma: atributo="valor". Todos los atributos deben escribirse separados tanto de la etiqueta como de otros atributos.

Los atributos admitidos y el valor que se le puede dar a cada uno de ellos vendrá determinado por la especificación de la versión de HTML que se esté usando, siendo posibles valores de tipo textual o numérico. Éstos últimos, pueden aparecer en diferentes bases (decimal, hexadecimal…) y con diferentes unidades (sin unidades, en píxeles, en porcentajes…).

Por tanto, la sintaxis de un elemento es la siguiente:

```
<nombre_elemento atributo1="valor1" atributo2="valor2" …>
    contenido_elemento
</nombre_elemento>
```

Por ejemplo, como se vio anteriormente, el elemento principal en un documento html es el delimitado por <html> y </html>. Su contenido es el propio contenido del documento, compuesto a su vez por los elementos <head> y <body>.

El principal atributo de <html> es lang, que identifica el idioma en el que está nuestra página. Por tanto, para una página en español, se escribiría:

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

Cada etiqueta puede tener unos posibles atributos, según viene dado en la especificación de la versión de HTML correspondiente. Eso sí, no todos los atributos admitidos en versiones anteriores se admiten en las posteriores, como es el caso de los atributos relativos a la presentación, como se verá más adelante.

Se debe tener en cuenta que cada etiqueta tiene sus propios requerimientos:

- Algunas necesitan aparecer cerradas, como <a>.
- Otras pueden tener etiqueta de cierre o no (como , que no siempre requiere).
- Otras no llevan nunca etiqueta de cierre (como <img>, la cual jamás lleva </img>).

No obstante, es importante saber que la validez de estas etiquetas cambia según la versión de HTML o XHTML que se esté usando, lo que puede inducir a error en algunos casos.

Para no equivocarse, conviene saber:

- Elementos vacíos (aquellos que no tienen etiqueta de cierre, como <br>): en HTML podían cerrarse o no. En XHTML deben cerrarse siempre (<br />). Sin embargo, en HTML5 no se usan elementos autocerrados, por lo que actualmente se escribe <br>. De todos modos, <br /> no produce error (en ocasiones se usa para mantener compatibilidad con XHTML), ya que HTML5 ignora la barra de autocierre en las siguientes etiquetas: area, base, br, col, embed, hr, img, input, link, meta, param, source, track, wbr.

- Elementos no vacíos: mientras que en XHTML deben cerrarse siempre, en HTML5 hay muchos que se permite que se escriban sin cerrar. Por ejemplo, <p>. Esto permite ahorrar código, si bien, habitualmente no resulta recomendable, ya que el código así escrito pierde legibilidad.

- Atributos vacíos: en XHTML no estaban permitidos, mientras que en HTML5 sí (su valor por defecto es una cadena vacía).

- Atributos con valores sin comillas: en XHTML no era válido, necesitando especificar comillas (dobles o simples), mientras que en HTML5 son válidos. No obstante, es recomendable usar siempre comillas para mantener un estilo uniforme y evitar errores cuando el valor del atributo contenga ciertos caracteres como el espacio.

3.1 Caracteres especiales

Hay caracteres que no se pueden escribir tal cual, en código HTML, unos por coincidir con las etiquetas (como <, > y &), otros por no aparecer en la codificación básica ASCII (las letras con tilde, las eñes), etc...

En general se pueden obtener todos los caracteres mediante un código numérico o textual, con la etiqueta **&xxx;** sustituyendo xxx por el número o el identificador textual del carácter correspondiente.

Por ejemplo, para mostrar por pantalla la etiqueta <body>, se debe escribir: **<body>** o, de forma equivalente: **<body>**

Estos y otros códigos son:

Carácter	En números	En letras	Carácter	En números	En letras
!	&#33;	--	"	&#34;	--
#	&#35;	--	$	&#36;	--
%	&#37;	--	&	&#38;	--
'	&#39;	--	(	&#40;	--
)	&#41;	--	*	&#42;	--
+	&#43;	--	,	&#44;	--
-	&#45;	--	.	&#46;	--
/	&#47;	--	:	&#58;	--
;	&#59;	--	<	&#60;	--
=	&#61;	--	>	&#62;	--
?	&#63;	--	@	&#64;	--
[	&#91;	--	\	&#92;	--
]	&#93;	--	^	&#94;	--
_	&#95;	--	`	&#96;	--
{	&#123;	--	|	&#124;	--
}	&#125;	--	~	&#126;	--
			¡	&#161;	&iexcl;
¢	&#162;	&cent;	£	&#163;	&pound;
¤	&#164;	&curren;	¥	&#165;	&yen;
¦	&#166;	&brvbar;	§	&#167;	&sect;
¨	&#168;	&uml;	©	&#169;	&copy;
ª	&#170;	&ordf;	«	&#171;	&laquo;
¬	&#172;	&not;	-	&#173;	&shy;
®	&#174;	&reg;	¯	&#175;	&macr;
°	&#176;	&deg;	±	&#177;	&plusmn;
²	&#178;	&sup2;	³	&#179;	&sup3;
´	&#180;	&acute;	µ	&#181;	&micro;
¶	&#182;	&para;	·	&#183;	&middot;
¸	&#184;	&cedil;	¹	&#185;	&sup1;
º	&#186;	&ordm;	»	&#187;	&raquo;
¼	&#188;	&frac14;	½	&#189;	&frac12;
¾	&#190;	&frac34;	¿	&#191;	&iquest;
À	&#192;	&Agrave;	Á	&#193;	&Aacute;
Â	&#194;	&Acirc;	Ã	&#195;	&Atilde;
Ä	&#196;	&Auml;	Å	&#197;	&Aring;

Æ	Æ	Æ	Ç	Ç	Ç
È	È	È	É	É	É
Ê	Ê	Ê	Ë	Ë	Ë
Ì	Ì	Ì	Í	Í	Í
Î	Î	Î	Ï	Ï	Ï
Đ	Ð	Ð	Ñ	Ñ	Ñ
Ò	Ò	Ò	Ó	Ó	Ó
Ô	Ô	Ô	Õ	Õ	Õ
Ö	Ö	Ö	×	×	×
Ø	Ø	Ø	Ù	Ù	Ù
Ú	Ú	Ú	Û	Û	Û
Ü	Ü	Ü	Ý	Ý	Ý
Þ	Þ	Þ	ß	ß	ß
à	à	à	á	á	á
â	â	â	ã	ã	ã
ä	ä	ä	å	å	å
æ	æ	æ	ç	ç	ç
è	è	è	é	é	é
ê	ê	ê	ë	ë	ë
ì	ì	ì	í	í	í
î	î	î	ï	ï	ï
ð	ð	ð	ñ	ñ	ñ
ò	ò	ò	ó	ó	ó
ô	ô	ô	õ	õ	õ
ö	ö	ö	÷	÷	÷
ø	ø	ø	ù	ù	ù
ú	ú	ú	û	û	û
ü	ü	ü	ý	ý	ý
þ	þ	þ	ÿ	ÿ	ÿ

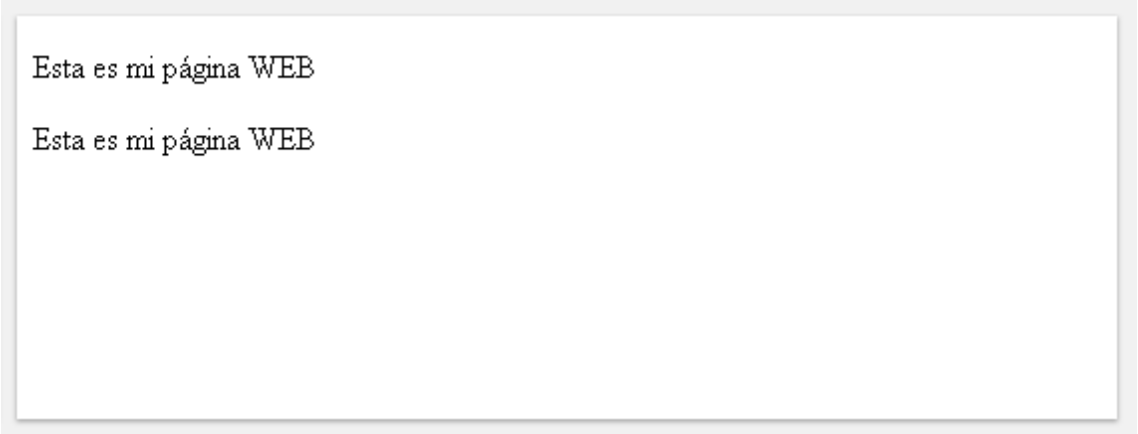
3.2 Espacios, tabuladores y saltos de línea

Todos los espacios en blanco, tabuladores y saltos de línea son considerados por el intérprete HTML como un espacio simple, ya que de esta forma se puede escribir el código ordenadamente y con sangrías.

Por tanto, si hace falta poner más de un espacio en blanco en medio de un texto, se debe usar el carácter especial equivalente al espacio en blanco visto en la sección anterior: ** **

Por ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>espacios</title>
</head>
<body>
   <p> Esta                es mi página WEB</p>
    <p>Esta es mi página WEB</p>
</body>
</html>
```



Por lo que, para que se vean las dos frases de forma diferente, como en el propio archivo html, habrá que escribir:

```
<!DOCTYPE html>
<html>
<head>
    <title>espacios</title>
</head>
<body>
   <p> Stainsby;        es mi
página WEB</p>
    <p>Esta es mi página WEB</p>
</body>
</html>
```

Ésta es mi página WEB

Esta es mi página WEB

Para conseguir saltos de línea, se debe usar la etiqueta **
** (no lleva etiqueta de cierre). No obstante, se debe tener en cuenta que otros elementos pueden generar por sí mismos saltos de línea, como los párrafos y encabezados.

Por el contrario, para conseguir tabulaciones, no se emplean etiquetas ni códigos especiales, sino que se definen márgenes, como se estudiará más adelante.

VÍDEO DE INTERÉS

Visualiza qué son los atributos en HTML

4. HERRAMIENTAS DE DISEÑO WEB

El diseño web es un proceso bastante creativo, pero a su vez, requieres de una especial dedicación a la elaboración del código, que, si bien, antiguamente tenías que escribir prácticamente de forma manual, hoy en día existen herramientas que permiten simplificar el proceso, por lo que te dispones a utilizarlas para rentabilizar tu tiempo de trabajo.

En el desarrollo de páginas web, partimos de la creación de las mismas mediante el proceso de maquetación, para posteriormente convertirlo en un sitio que permita la navegación web, desde una página estática a una dinámica combinando diferente código como PHP, JavaScript, etc., y sus conexiones a bases de datos, por ejemplo.

Disponemos de una serie de aplicaciones o herramientas que nos van a facilitar la tarea del diseño, desde simples editores de texto, hasta herramientas que automatizan procesos y amplían funcionalidades mediante las opciones que ofrecen, incluso la instalación de extensiones o plugins.

A la hora de la elección de la herramienta de diseño web, sí debemos tener presenta una serie de recomendaciones que se basan en su facilidad de uso, las características que ofrece de acuerdo a nuestras necesidades, compatibilidad con otras aplicaciones o plataformas o si son de carácter gratuito o de pago.

Herramientas diseño web.
Fuente: https://federicosanchez.es/herramientas-diseno-web/

Antes de hablar de herramientas propiamente dichas, hemos comentado el tema de editores de texto, que por su sencillez se utilizan bastante en la elaboración de código y que encontramos aplicaciones un poco más evolucionadas como es el caso de Notepad++, editor de texto que permite la escritura de código fuente de manera gratuita, muy superior en prestaciones al Bloc de notas y se ejecuta en un entorno Windows. Existiendo otras alternativas como IntelliJ IDEA o Evernote Teams.

ENLACE DE INTERÉS

En este enlace tienes disponible la descarga de Notepad++:

Centrándonos ya en herramientas que facilitan el trabajo de los desarrolladores a la hora de crear páginas web, con acciones que le permiten entre otras, la elección de colores e imágenes, eliminación de redundancias, repeticiones o la automatización de procesos. Podemos destacar las siguientes herramientas:

WordPress

WordPress.
Fuente: https://www.net948.com/2022/05/17/que-es-wordpress/

Inicialmente diseñado para la creación de blogs, pronto se convirtió en un sistema de gestión de contenido (CMS) gratuito y de código abierto. Destaca por su simplicidad y facilidad de uso, muy utilizado para el desarrollo de sitios web.

A pesar de su carácter gratuito, se deberá contar un alojamiento web, existiendo la opción de ampliar sus funcionalidades mediante la instalación de plugins, tanto gratuitos como de pago.

Entre sus ventajas destacamos:

- Permite la creación de contenidos de manera sencilla mediante entradas, páginas, publicaciones, así como la protección del contenido mediante contraseña.
- Es multilenguaje.
- Gran oferta de temas tanto gratuitos como de pago, dentro del propio sitio en la url https://es.wordpress.org/themes/, incluso de proveedores externos como ThemeForest o Themeisle.
- Igualmente ofrece más de 50.000 plugins con funcionalidades que van desde la creación de formularios, hasta la creación de una tienda online.
- Administración de medios y de usuarios.
- Optimización del sitio web a nivel SEO.

Inconvenientes

- Mantenimiento, seguridad, actualizaciones y copias de seguridad por parte del usuario, en caso contrario, se debe contratar un alojamiento web administrado.
- Vulnerable a ataques de seguridad por su gran número de usuarios, e instalación de plugins de terceros. Requiere actualización continua del software.

Disponible desde su página oficial https://es.wordpress.org/.

Wix

Wix.
Fuente: https://twitter.com/MundoWix

Herramienta de diseño web con versión gratuita y de pago, de fácil manejo que ofrece tres niveles diferentes en función del tipo de usuario, encontramos:

- Editor Wix, que incorpora la opción de arrastrar y soltar, para la creación de contenido.
- Wix ADI (Inteligencia de Diseño Artificial), creador automático de sitios web completos, tras las respuestas obtenidas al cuestionario sobre preferencias.
- Velo by Wix, plataforma para desarrolladores web independientes con incorporación de código abierto y la opción de desarrollar aplicaciones independientes.

Entre sus ventajas destacamos:

- Gran variedad de plantillas y galerías de medios.
- Posibilidad de integrar aplicaciones oficiales y de terceros.
- Herramientas comerciales y de marca, SEO, etc.
- Alojamiento gratuito de hasta 500 MB.
- Soluciones integradas (según plan) de comercio electrónico.

Inconvenientes

- Carencias en el diseño adaptativo a nivel principiante, sí disponible para desarrolladores.
- Almacenamiento limitado.
- No permite acceso a las CSS ni modificar el sitio una vez publicado.

La descarga está disponible desde el sitio web https://es.wix.com/.

Adobe Dreamweaver

Adobe Dreamweaver.
Fuente: https://re-captha-version-3-25.top/ms/2808_desc_1_A/?c=46c14c44-87de-4cda-9b98-37ee0ad23458&a=l136701#

Herramienta muy utilizada en sus distintas versiones anteriores y que ahora pasó a propiedad de Adobe. Permite varios lenguajes de programación como pueden ser HTML, JavaScript o CSS, diseñando con ella sitios web responsive.

Su filosofía de trabajo no permite el arrastrar y soltar como ocurre con otras herramientas de diseño web, pero ofrece una gran potencialidad por su amplia variedad de funciones que incorpora y un muy efectivo editor de código.

Son versiones de pago mediante los correspondientes planes, ofreciendo una versión de prueba de 7 días, siendo operativo tanto para Wndows como MacOS.

Ventajas

- Cuenta con diseños de plantillas por defecto.
- Dispone de un panel de archivos para administrar, mover y sincronizar tanto archivos como carpetas.
- Edición de código con opción de vista en paralelo.
- Ofrece sugerencias de código que facilita su inserción.
- Mapas de imágenes que crean puntos de acceso interactivos.
- Multitarea en monitores Windows.
- Desarrollo de aplicaciones móviles con la integración de jQuery Mobile.

Inconvenientes

- Altos precios de los planes.
- No muy apropiada para principiantes por su complejidad si se carecen conocimientos de programación.

Otros tipos de herramientas para diseño web

Además de las herramientas propias de diseño web, encontramos una serie de herramientas muy útiles para los distintos ámbitos relacionados como son:

- Diseño gráfico
  - Adobe Photoshop
  - Adobe Illustrator
  - Affinity Designer
  - CorelDRAW Graphics Suite.

- Colaborativas
  - Adobe XD
  - InVision Studio
  - Sketch
  - Figma

- Marketing
  - Canva
  - Adobe Express
  - Visme
  - Ceros

ENLACE DE INTERÉS

En esta página puedes ver un completo tutorial sobre WordPress

EJEMPLO PRÁCTICO

Eduardo es el jefe del departamento de informática de su empresa, donde se realizan diferentes tareas entre las que se encuentra el desarrollo web del sitio corporativo, que se encuentra en pleno proceso de actualización.

Hasta el momento se estaban utilizando editores de texto y aplicaciones que requerían altos conocimientos de programación para el desarrollo de nuevas páginas web, pero en la actualidad existen una serie de herramientas en el mercado que facilitan en gran medida la tarea y no requiere de perfiles tan técnicos.

¿Qué dos herramientas podría destacar y qué ventajas aportarían?

En la actualidad, dos de las herramientas a destacar serían las siguientes:

WordPress

Entre sus ventajas destacamos:

- Permite la creación de contenidos de manera sencilla mediante entradas, páginas, publicaciones, así como la protección del contenido mediante contraseña.
- Es multilenguaje.
- Gran oferta de temas tanto gratuitos como de pago, dentro del propio sitio en la url https://es.wordpress.org/themes/, incluso de proveedores externos como ThemeForest o Themeisle.
- Igualmente ofrece más de 50.000 plugins con funcionalidades que van desde la creación de formularios, hasta la creación de una tienda online.
- Administración de medios y de usuarios.
- Optimización del sitio web a nivel SEO.

Wix

Entre sus ventajas destacamos:

- Gran variedad de plantillas y galerías de medios.
- Posibilidad de integrar aplicaciones oficiales y de terceros.
- Herramientas comerciales y de marca, SEO, etc.
- Alojamiento gratuito de hasta 500 MB.
- Soluciones integradas (según plan) de comercio electrónico.

5. HOJAS DE ESTILO (CSS)

Existen distintas formas de poder dar formato al contenido de una página web, pero la aparición de las hojas de estilo en cascada o CSS, aportaron una serie de opciones y funcionalidades que mejoran la calidad de los sitios a nivel de diseño y mantenimiento, por lo que vas a tener que profundizar un poco sobre qué son y cómo se utilizan.

Para solucionar todos los inconvenientes presentados en la introducción, surgen las llamadas hojas de estilo en cascada o CSS (Cascading Style Sheets). Se trata de un lenguaje que permite definir una serie de reglas que pueden ser aplicadas a cualquiera de los elementos HTML. De esta forma, el diseño se puede realizar exclusivamente con estas reglas, permitiendo así aislar contenido y presentación.

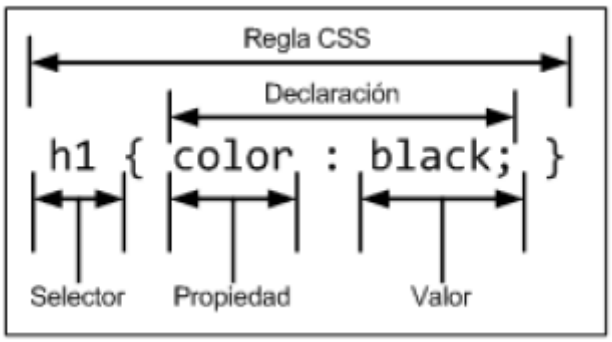
ENLACE DE INTERÉS

Accede a un completo tutorial sobre CSS en este enlace:

Las reglas CSS, que han permitido sustituir los antiguos atributos de presentación de las etiquetas HTML por reglas independientes de las propias etiquetas.

Además, las reglas CSS permiten ser aplicadas a elementos HTML concretos, por lo que, además de estar formadas por declaraciones (propiedades y sus valores asociados), como las vistas en el tema anterior, incluyen los llamados selectores, que no son más que filtros que hacen que una regla se aplique a un elemento u otro.

Cada uno de los estilos que se definen en una hoja de estilos es lo que se denomina regla CSS:



Regla CSS.

Cada regla está compuesta por una primera parte de selectores, un símbolo de llave de apertura ({), un conjunto de declaraciones que siguen el formato visto para los estilos usados en la unidad anterior y, finalmente, un símbolo de llave de cierre (}).

Desglosando cada uno de estos elementos, quedaría:

- **Selector**: indica el elemento o elementos HTML a los que se aplica la regla CSS.
- **Declaración**: especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS con sus valores correspondientes.
- **Propiedad**: permite modificar una determinada característica del elemento.
- **Valor**: indica el valor que se le quiere dar a esa característica del elemento.

Por ejemplo:

p {color:blue}

El selector sería p, que hace referencia a dichos elementos HTML, es decir a los párrafos. En su interior, la regla contiene una única declaración, para la propiedad color, asignándole el valor blue.

Así pues, se trata de una sencilla regla que aplica a los párrafos la propiedad color con el valor azul. Es decir, hace que se muestre el texto de los párrafos en azul.

Al igual que ocurría con HTML, el lenguaje CSS también permite escribir comentarios.

En html:

- *<!-- Este es un comentario en HTML -->*
- *<!-- Este es un comentario HTML de varias lineas -->*

En CSS:

- */* Este es un comentario en CSS */*
- */* Este es un comentario CSS de varias lineas */*

Es el uso de los selectores el que otorga a las hojas de estilo la capacidad real de separar contenido de presentación ya que, aunque los estilos vistos en la unidad anterior no forman parte de las propias etiquetas HTML, al estar definidos en su interior, hacen que sea necesario manipular el propio código HTML para modificar el diseño.

ENLACE DE INTERÉS

Siempre es conveniente tener a mano la referencia de CSS para poder consultar todas las posibles reglas que se pueden definir. En esta página se incluye una referencia muy completa y detallada del mismo:

Este conjunto de reglas CSS se podrá aplicar a:

- Un sitio web al completo, de manera que se puede modificar el diseño de todas las páginas que componen el sitio de una sola vez.
- Un solo documento HTML o página, definiendo el estilo en la cabecera.
- Una porción del documento, aplicando estilos solo a un trozo de la página.
- Una etiqueta concreta, pudiendo incluso definir varios estilos diferentes para una misma etiqueta. Esto permite definir, por ejemplo, un estilo común para los párrafos y luego asignar un color distinto a cada uno de ellos manteniendo el resto de las propiedades ya definidas.

Además, las reglas CSS permiten mucha mayor potencia que la que permitían los pocos atributos existentes para las etiquetas HTML. El control de los distintos aspectos gráficos es mucho más exacto, además de contar con muchas de las características más comunes para maquetación de documentos, como las que están presentes en los procesadores de texto. Esto permite:

- Definir la distancia entre líneas del documento.
- Aplicar sangrado a las primeras líneas del párrafo.
- Colocar elementos en la página con mayor precisión.
- Definir la visibilidad de los elementos, su opacidad, etc.

Además, si con HTML tan sólo era posible definir atributos en las páginas usando pixeles y porcentajes, con CSS es posible utilizar una gran variedad de unidades como:

- Píxeles (px) y porcentaje (%), al igual que antes.
- Pulgadas (in)
- Puntos (pt)
- Centímetros (cm)
- Otras unidades relativas, como se verá posteriormente.

Existen tres formas de usar CSS: aplicar estilos en línea, mediante una hoja de estilos interna y mediante una hoja de estilos externa.

1. Estilos en línea

La forma más sencilla de aplicar reglas de estilo es definirlas directamente sobre las etiquetas HTML, como se vio en la unidad anterior.

Consiste en usar el atributo **style** incluyendo en su interior tantas declaraciones como propiedades se deseen ajustar en ese elemento. Dichas declaraciones se expresan de la forma propiedad:valor. Para separar unas de otras se usa el punto y coma (el último se puede omitir):

style="propiedad1:valor1;propiedad2:valor2;…"

EJEMPLO PRÁCTICO

A continuación, se muestra un párrafo en color azul, usando estilos en línea:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
</head>
<body>
    <p style="color:blue">
        Esto es un párrafo con las letras en color azul.
    </p>
</body>
</html>
```

Esto es un párrafo con las letras en color azul.

Como se ha comentado anteriormente, **no es recomendable definir los estilos de esta forma**, para separar verdaderamente contenido y presentación.

2. Hoja de estilos interna

Si todos los estilos individuales usados en el método anterior se agrupan al principio del documento, se obtiene lo que se denomina hoja de estilos interna. Al definir tales estilos en la cabecera del documento, éstos serán aplicados a toda la página. El funcionamiento será similar al anterior, pero debiendo usar selectores para poder indicar cuáles son los elementos de la página sobre los que actúa cada regla. Tiene la ventaja de poder especificar una sola vez una misma regla que deba ser aplicada a varios elementos. Así mismo, si se desea cambiar la apariencia de la página, permite hacerlo sin tener que modificar el código HTML, sino sólo modificando dicha cabecera. Esto representa un paso más en la separación entre contenido y presentación, consiguiendo aislar ambas partes completamente, aunque se trate del mismo archivo.

El funcionamiento es sencillo, consiste en añadir a la cabecera una sección de estilos mediante el elemento style. La etiqueta correspondiente es **<style>**, debiendo cerrarse (</style>). Dentro de este elemento se incluyen tantas reglas CSS como se desee, siguiendo el formato explicado anteriormente, es decir, indicando por cada una un selector y un conjunto de declaraciones de estilos entre llaves y separadas por punto y coma (el último se puede omitir, aunque resulta conveniente escribirlo para evitar errores si se añade un nuevo estilo).

Por ejemplo, para realizar el caso anterior del párrafo con letras azules, se extraería la regla que anteriormente se incluyó dentro de la etiqueta <p> y se le especificaría delante el selector p:

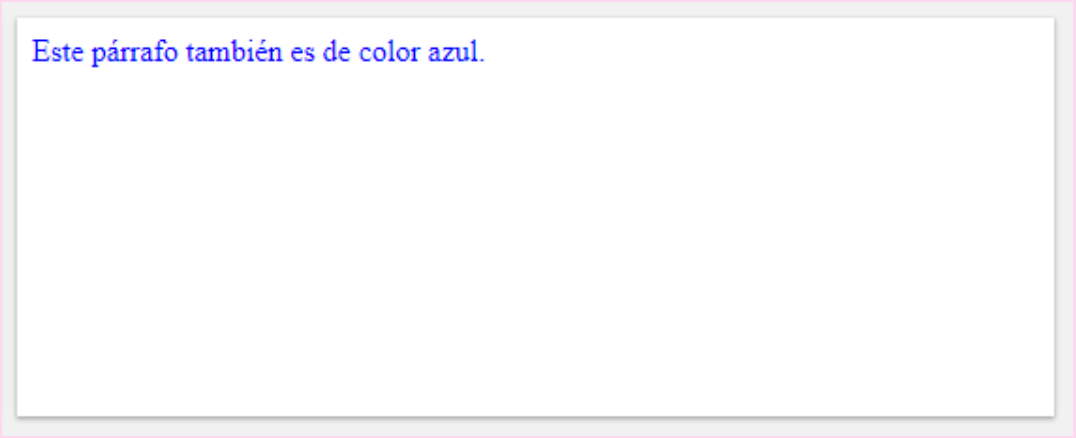
```
p {color:blue;}
```

Esta regla CSS se incluye ahora en la sección <style> de la cabecera HTML.

EJEMPLO PRÁCTICO

En este ejemplo, se muestra el mismo párrafo azul del ejemplo anterior, pero usando una hoja de estilos interna.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        p{color: blue;}
    </style>
</head>
<body>
    <p>
        Este párrafo también es de color azul.
    </p>
</body>
</html>
```

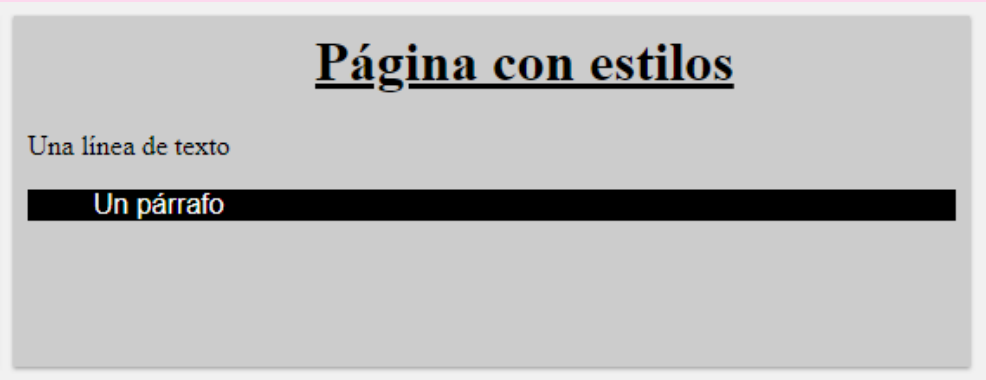


A continuación, se muestra un ejemplo más avanzado, en el que se aplican estilos a un título (h1), un párrafo (p) y a la página al completo (body):

EJEMPLO PRÁCTICO

Ejemplo más completo realizado con una hoja de estilos interna.

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

        body {
             color: black;
             background-color: #cccccc;
             text-indent: 1cm;
           }

         h1 {
             text-decoration: underline;
             text-align: center;
           }

          p {
             font-family: arial,verdana;
             color: white;
             background-color: black;

           }
    </style>
</head>



<body>
           <h1>  Página con estilos </h1>
                 Una línea de texto
           <p> Un párrafo  </p>
</body>
</html>
```



Como se puede apreciar en el código, sobre la etiqueta <body>, se aplican los tres estilos siguientes:

- color: black, que establece el color negro como color del texto. En principio, cualquier texto de la página aparecerá de este color.
- background-color: #cccccc, establece como color de fondo un tono de gris.
- text-indent:1cm, define un sangrado de 1 centímetro, es decir, todo el texto comenzará con un centímetro de separación con el margen izquierdo.

Sobre la etiqueta <h1>, se aplican los dos siguientes estilos:

- text-decoration: underline, que marca el texto como subrayado.
- text-align: center, que centra el texto.

Por tanto, los títulos aparecen centrados y subrayados. Y, finalmente, sobre la etiqueta <p>:

- font-family: arial,verdana, define este tipo de fuente para el párrafo.
- color: white, hace que el color de la fuente del párrafo sea blanco.
- background-color: black, pone como fondo del párrafo un color gris.

Una de las ventajas de agrupar reglas en lugar de usarlas en el interior de la propia etiqueta, es que se pueden aplicar características comunes a un conjunto de etiquetas sin necesidad de repetir los mismos estilos para cada una de ellas. Se puede apreciar fácilmente si se recuerdan los ejemplos de tablas vistos en la unidad anterior. Para asignar un borde a cada celda, era necesario especificarlo en cada una de ellas:

EJEMPLO PRÁCTICO

En la unidad anterior, se definían estilos para cada uno de los bordes de las celdas de forma individual:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
</head>
<body>
    <table style="border:1px solid black">
        <tr>
            <td style="border:1px solid black">00</td>
            <td style="border:1px solid black">01</td>
        </tr>
        <tr>
            <td style="border:1px solid black">10</td>
            <td style="border:1px solid black">11</td>
        </tr>
    </table>
</body>
</html>
```

Ahora, sin embargo, bastará con especificar una sola regla usando el selector td:

EJEMPLO PRÁCTICO

En este ejemplo, se muestra la misma tabla que en el ejemplo anterior, pero definiendo un estilo para los bordes de todas las celdas de forma conjunta:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        td{border:1px solid black;}
    </style>
</head>
<body>
    <table style="border:1px solid black">
        <tr>
            <td>00</td>
            <td>01</td>
        </tr>
        <tr>
            <td>10</td>
            <td>11</td>
        </tr>
    </table>
</body>
</html>
```

Como se puede ver, también es posible combinar las diferentes formas de aplicar las reglas CSS, en este caso, en línea e interna. No obstante, como se ha comentado anteriormente, no es recomendable usar estilos en línea.

NOTA DE INTERÉS

Se debe tener en cuenta que los estilos definidos de esta forma se aplicarán a todos los elementos de la página, por lo que, si en el ejemplo hubiera otra tabla, sus celdas aparecerían también con bordes. Más adelante, se verá cómo usar selectores más específicos para estos casos.

3. Hoja de estilo externa

Esta tercera solución consiste en definir esas mismas reglas que se han colocado antes en la cabecera del archivo, en un archivo independiente. De esta forma, se puede usar una misma hoja de estilos para todas las páginas de un sitio web.

Al compartir todas las páginas una misma declaración de estilos, si se cambia alguno, afectará a todas las páginas. Esto resulta muy útil ya que, habitualmente, un sitio web presenta un aspecto uniforme en todas sus páginas. Además, permite reducir el total de bytes transferidos cuando se está navegando por un sitio web, puesto que en muchas ocasiones los navegadores no volverán a descargar el archivo de estilos repetidamente cada vez que se visite una página.

La creación de un fichero de declaración de estilos es un fichero de texto normal, que puede tener cualquier extensión, aunque se le suele asignar la extensión **.css** para una mejor organización de los archivos de la web.
El contenido de dicho archivo, debe ser exclusivamente un conjunto de reglas CSS, no se puede incluir contenido HTML.

Por ejemplo, para el caso visto en el punto anterior, el archivo estilos.css quedaría así:

EJEMPLO PRÁCTICO

Contenido del archivo de estilos. Como se puede observar, no debe aparecer ninguna etiqueta HTML:

```
body {
    color: black;
    background-color: #cccccc;
    text-indent: 1cm;
}
h1 {
    text-decoration: underline;
    text-align: center;
}
p {
    font-family: arial,verdana;
    color: white;
    background-color: black;
}
```

Existe la posibilidad de, una vez que se ha guardado el archivo .css, que contiene las reglas de estilo, se debe referenciar en la página para que pueda ser leído al cargarla. Para ello, se usa la etiqueta sin cierre **<link>** con los siguientes atributos:

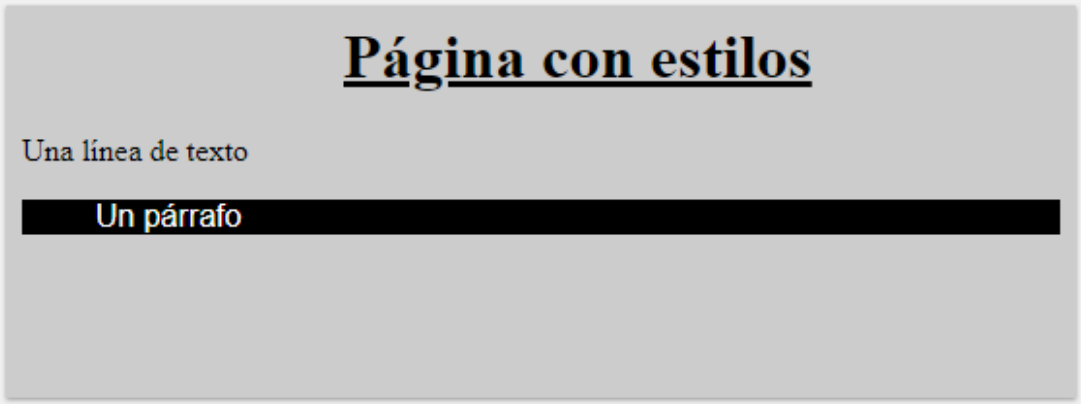
- **rel**="stylesheet", indicando que se trata de un el enlace a una hoja de estilos.
- **type**="text/css", que indica que el archivo es de texto y sigue la sintaxis CSS.
- **href="…"**, para incluir el nombre del archivo en que se han almacenado los estilos. En el ejemplo anterior, sería href="estilos.css".

El archivo HTML de dicho ejemplo quedaría tan sencillo como se ve a continuación:

EJEMPLO PRÁCTICO

Esta página usa la hoja de estilos presentada en el ejemplo anterior:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="estilos.css">
</head>
<body>
    <h1>Página con estilos</h1>
    Una línea de texto
    <p>Un párrafo</p>
</body>
</html>
```



Si el navegador no encontrara el fichero estilos.css, lo que mostraría sería:

5.1 Herencia

Una de las características fundamentales de las hojas de estilo en cascada es la herencia. Esto se traduce en que las propiedades de estilo definidas para una etiqueta son heredadas por el resto de las etiquetas que aparezcan más abajo en la jerarquía, siempre que éstas no definan dicha propiedad nuevamente.

El funcionamiento se puede ver en el mismo ejemplo anterior. Para la etiqueta <body> se define el color de fuente como negro, por lo que tanto título como párrafo deben aparecer en este color. Sin embargo, la regla para <p> redefine dicha propiedad a blanco, y la regla para <h1> no la vuelve a redefinir. Por ello, el título aparece en negro, heredando dicho valor de <body>. Y el texto del párrafo aparece en blanco, al sobrescribir el valor negro heredado de <body> por el blanco definido en su propia regla.

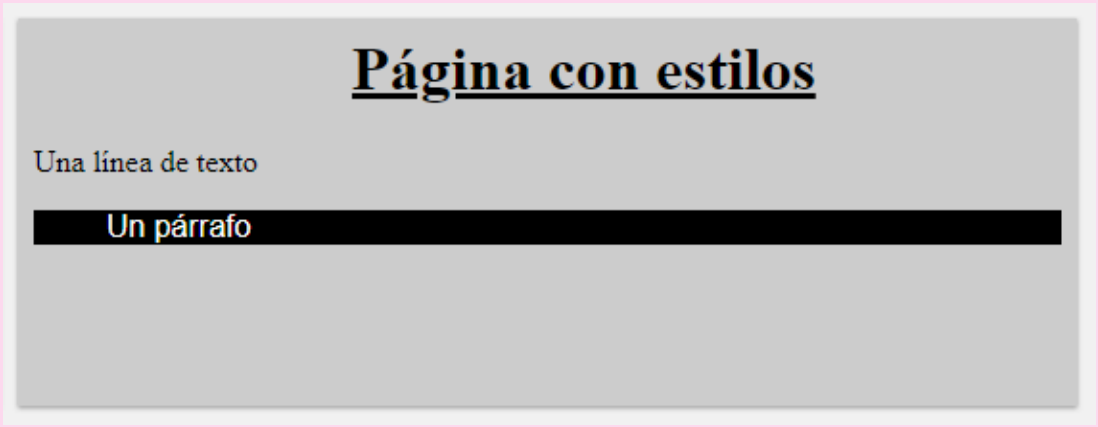
En este caso, no importa el orden en el que fueron definidas las reglas, ya que se aplican en el orden en el que aparezcan las etiquetas HTML. Por tanto, el siguiente código, con la regla para <body> al final, dará el mismo resultado:

EJEMPLO PRÁCTICO

En este ejemplo, el orden de las reglas es diferente al de ejemplos anteriores:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        h1 {
            text-decoration: underline;
            text-align: center;
        }
        p {
            font-family: arial,verdana;
            color: white;
            background-color: black;
        }
        body {
            color: black;
            background-color: #cccccc;
            text-indent: 1cm;
        }
    </style>
</head>
<body>
    <h1>Página con estilos</h1>
    Una línea de texto
```

```
    <p>Un párrafo</p>
</body>
</html>
```



Como se puede observar, otra de las propiedades heredadas, que es el color de fondo, también se sobrescribe en la regla para <p>, por lo que aparece de un color distinto al del fondo de la página, en este caso de color negro.

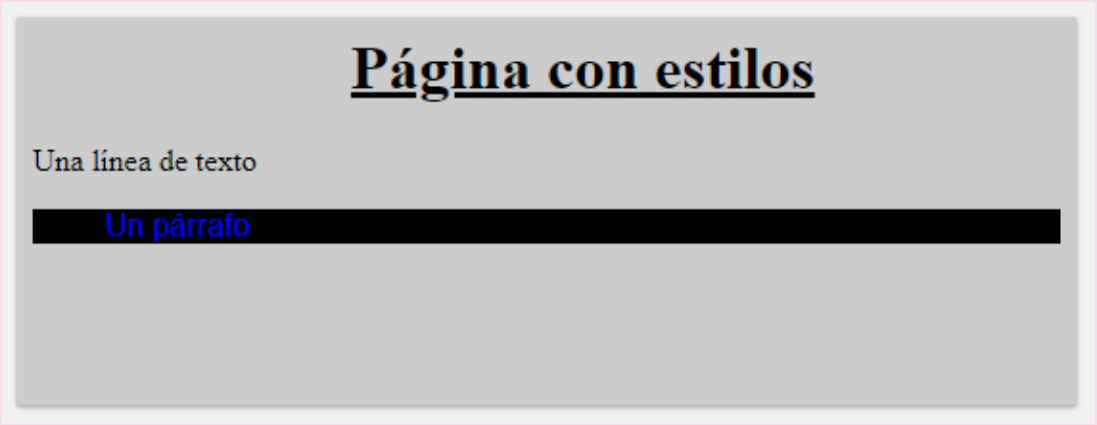
Es posible que se redefinan las propiedades para un mismo elemento. En ese caso, se aplican en el orden en el que se encuentren definidas, por lo que la regla que tendrá efecto será la definida en último lugar. Por ejemplo, si se vuelve a definir otro color para el párrafo, éste se mostrará en ese último color:

EJEMPLO PRÁCTICO

El siguiente ejemplo, añade una regla al ejemplo anterior en la que se redefine la propiedad color para el elemento p:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        h1 {
            text-decoration: underline;
            text-align: center;
        }
        p {
            font-family: arial,verdana;
            color: white;
            background-color: black;
        }
        body {
```

```
            color: black;
            background-color: #cccccc;
            text-indent: 1cm;
        }
        p { color: blue; }
    </style>
</head>
<body>
    <h1>Página con estilos</h1>
    Una línea de texto
    <p>Un párrafo</p>
</body>
</html>
```



5.2 Elementos en línea y bloques

Según se ha visto hasta el momento, sólo sería posible definir estilos diferentes a etiquetas diferentes. Pero ¿qué sucedería si por ejemplo se quisiera dar un color distinto a una sola palabra dentro de un párrafo? La solución es usar etiquetas cuya función es agrupar otros elementos. Esas etiquetas son y .

Antes de nada, es necesario saber qué es un elemento en línea y qué es un elemento en bloque. De forma general, en HTML, todos los elementos son de una de estas dos clases (en HTML5 la clasificación es más amplia y los elementos en bloque son conocidos como elementos dinámicos, frente a los en línea, que serían estáticos).

Los elementos en línea son los que se comportan como un texto dentro de un párrafo, es decir, ocupan un espacio determinado, y tras éste, aparecerán los siguientes elementos. Por el contrario, los elementos en bloque tienden a ocupar todo el espacio disponible dentro del elemento padre o contenedor, formando un bloque.

Tanto los párrafos como los títulos son elementos en bloque, mientras que las imágenes son elementos en línea. Para comprobarlo, en el siguiente ejemplo, se puede observar viendo el coloreado del fondo de párrafo y título, que queda delimitado un bloque y se produce un salto de línea, mientras que, en el caso de la imagen, ésta permanece en medio del texto en el que ha sido insertada, ocupando únicamente su espacio:

EJEMPLO PRÁCTICO

A continuación, se puede observar la diferencia entre los elementos en línea y en bloque:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        h1 {
            color: green;
            background-color: blue;
        }
        p {
            color: white;
            background-color: black;
        }
        body {
            background-color: #cccccc;
        }
    </style>
</head>
<body>
    <h1>Título</h1>
    Una línea de texto <img src="playa.jpg" width="100"
alt="playa">con una imagen en medio
    <p>Un párrafo</p>
</body>
</html>
```

PARA SABER MÁS

En esta página se explica detalladamente la diferencia entre elementos en línea y en bloque:

**Etiqueta **

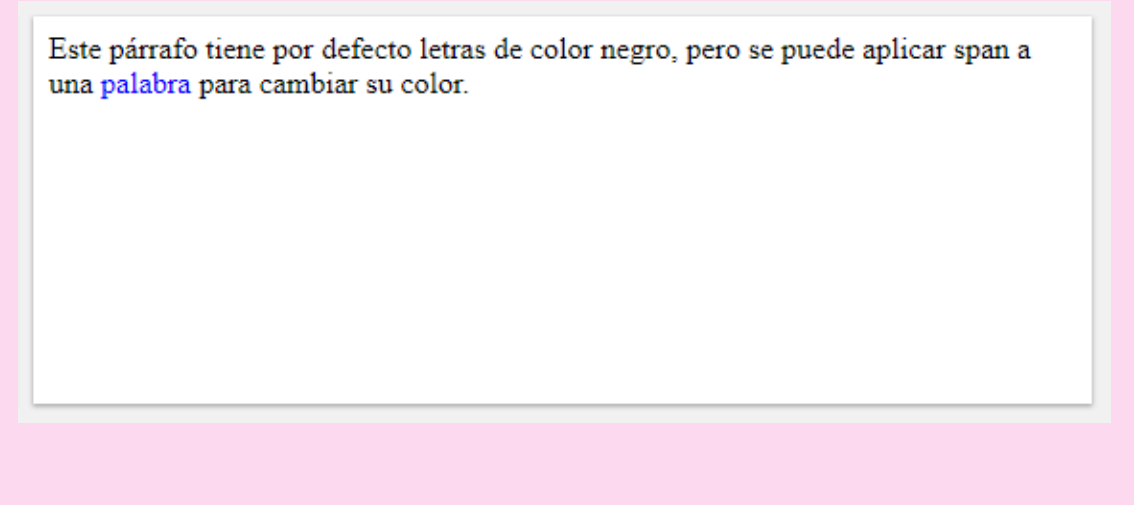
La etiqueta sirve para agrupar elementos en línea. Se trata de una etiqueta que no realiza ninguna función por sí misma. Por ello, puede ser usada para asignar estilos a un elemento sin tener otras consecuencias. Debe usarse con etiqueta de cierre.

Esto permite, por ejemplo, aplicar un estilo particular a una sola palabra dentro de un párrafo. En el siguiente código, se usa para cambiar el color:

EJEMPLO PRÁCTICO

En este ejemplo se utiliza la etiqueta span para dar un color diferente a una palabra dentro de un párrafo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
<body>
    <p>
        Este párrafo tiene por defecto letras de color negro, pero
se puede aplicar span a una <span style="color:blue;">
palabra</span> para cambiar su color.
    </p>
</body>
</html>
```



**Etiqueta **

La etiqueta (división) se utiliza para agrupar contenido o crear secciones. Se trata de un elemento en bloque que, al igual que sucede con la etiqueta , se utiliza para dar estilo sin afectar a las propiedades del elemento. No obstante, en este caso al ser un elemento en bloque, por defecto, generará un salto de línea. Debe usarse también con etiqueta de cierre.

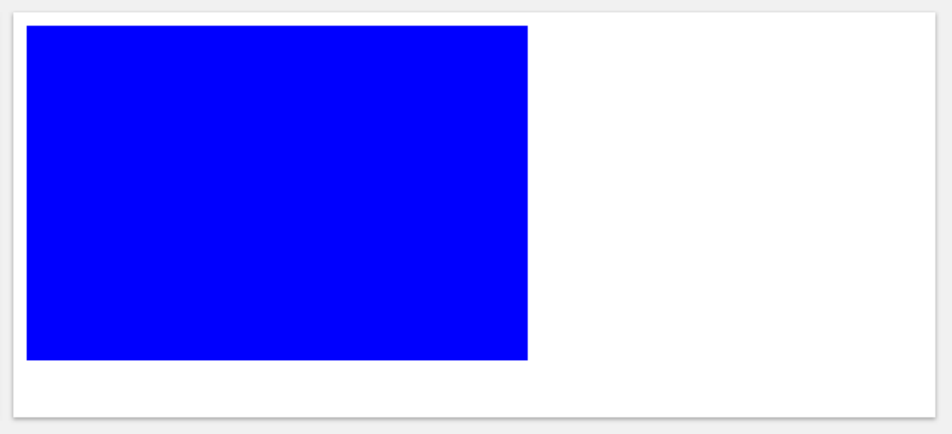
Los bloques definidos mediante etiquetas son los principales elementos de maquetación de una página web. Se pueden ver como cajas que se van colocando unas junto a otras.

Las etiquetas semánticas estructurales vistas en la unidad anterior son en cuanto a comportamiento, equivalentes a bloques , pero con un nombre más descriptivo.

EJEMPLO PRÁCTICO

En este ejemplo se utiliza la etiqueta div para definir varios bloques:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
<body>
    <div style="background-color:blue;width:300px;height:200px;">
    </div>
</body>
</html>
```



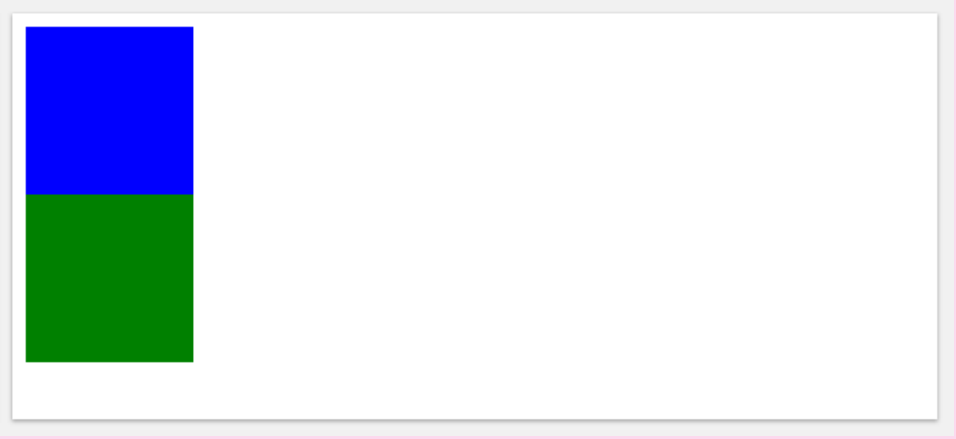
Los bloques o sus equivalentes semánticos, son los principales elementos usados para maquetar. Para ello, es fundamental conocer los conceptos de posicionamiento:

- Estático: se define mediante la propiedad position:static. Se usa para indicar que el posicionamiento sigue el flujo natural de la página, es decir, se usa para indicar que el elemento no tiene ningún posicionamiento específico. Es el valor por defecto de todos los elementos HTML.

EJEMPLO PRÁCTICO

A continuación, se muestran dos bloques posicionados de forma estática.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        div {width:100px;height:100px;}
    </style>
<body>
    <div style="background-color:blue;">
    </div>
    <div style="background-color:green;">
    </div>
</body>
</html>
```

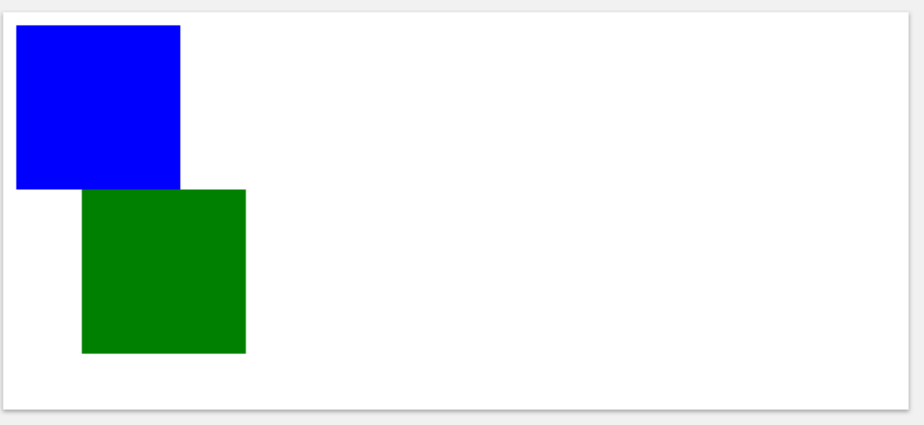


- Relativo: se define mediante position:relative. Los elementos con posicionamiento relativo se ajustan con respecto a su posición normal, es decir, la que tendrían si mantuvieran el valor static.

EJEMPLO PRÁCTICO

Este ejemplo muestra el mismo caso anterior, pero posicionando el segundo bloque de forma relativa.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        div {width:100px;height:100px;}
    </style>
<body>
    <div style="background-color:blue;">
    </div>
    <div style="background-color:green;position:relative;
                left:40px;">
    </div>
</body>
</html>
```

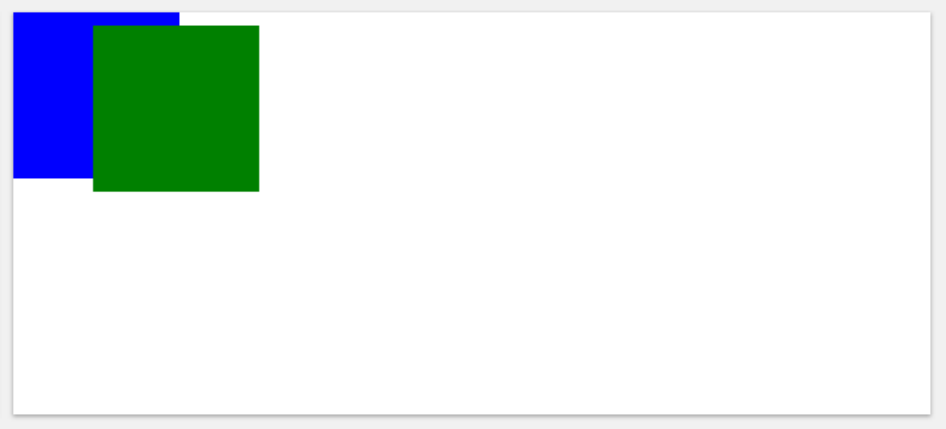


- Fijo: se usa para posicionar un elemento de forma fija. Su posición no variará, aunque se haga scroll. Para ello se utiliza position:fixed. No quita espacio al resto de los elementos. Su posición se expresa en relación con el área visible de la página.

EJEMPLO PRÁCTICO

Siguiendo con el mismo ejemplo, ahora se posiciona el primer bloque de forma fija en el origen, con lo que el segundo se coloca como si no hubiera ninguno colocado previamente, quedando superpuesto.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        div {width:100px;height:100px;}
    </style>
<body>
    <div style="background-color:blue;position:fixed;left:0;
                right:0;">
    </div>
    <div style="background-color:green;position:relative;
                left:40px;">
    </div>
</body>
</html>
```



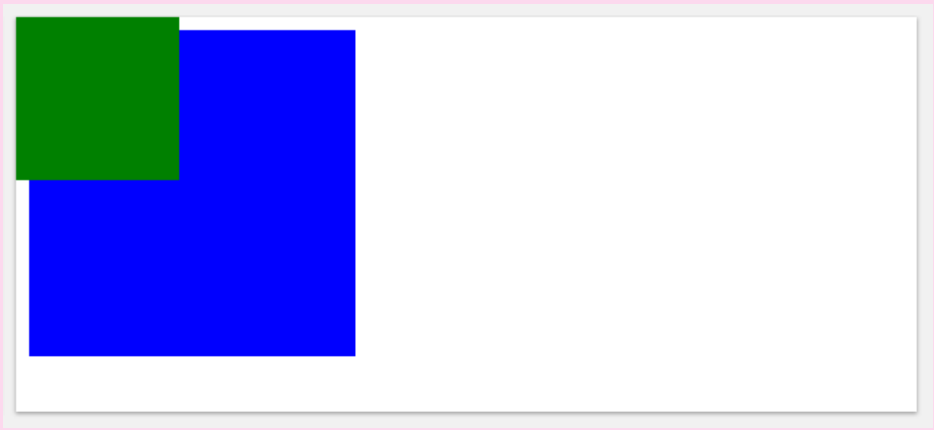
- Absoluto: se define con position:absolute. Sirve para posicionar un elemento de forma fija, pero especificando su posición en relación al anterior elemento posicionado en la jerarquía, en lugar de con respecto a la página como ocurría en fixed. Por ejemplo, si un div posicionado de forma absoluta está dentro de otro que usa algún tipo de posicionamiento (es decir, no static), la posición se fijará con respecto a este último. Por el contrario, si en la jerarquía no se encuentra ningún elemento anterior con algún posicionamiento definido, la posición quedará fijada con respecto a body.

Se debe tener en cuenta también que, a diferencia de fixed, estos elementos varían su posición con el scroll.

EJEMPLO PRÁCTICO

A continuación, se muestra el efecto de definir un bloque con posición absoluta, cuando no está contenido en ningún bloque con posicionamiento, por lo que se posiciona con respecto a body:

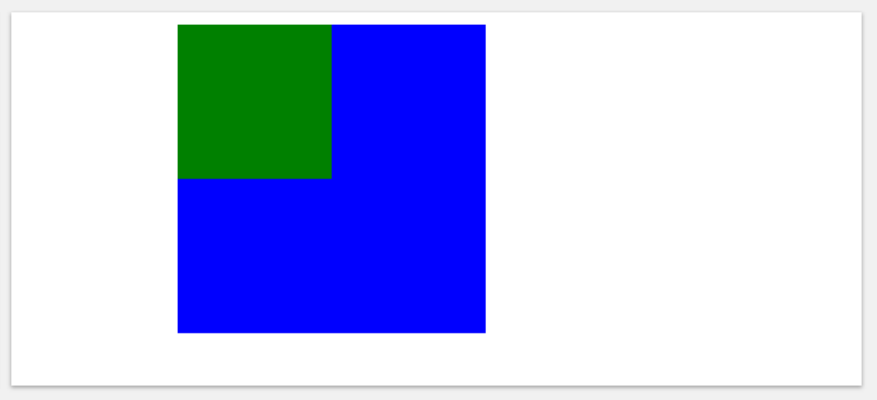
```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
<body>
    <div style=" background-color:blue;width:200px;height:200px;">
        <div style="background-color:green;width:100px;height:100px;
                    position:absolute;left:0;top:0;">
        </div>
    </div>
</body>
</html>
```



EJEMPLO PRÁCTICO

Si en el caso anterior, se indica algún tipo de posicionamiento al primer bloque, entonces el segundo quedará posicionado con respecto al primero:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
<body>
    <div style=" background-color:blue;width:200px;height:200px;
                 position:relative;left:100px;">
        <div style="background-color:green;width:100px;height:100px;
                    position:absolute;left:0;top:0;">
        </div>
    </div>
</body>
</html>
```



- Sticky: se trata de un posicionamiento poco común ya que aún no está admitido por todos los navegadores, que coloca el elemento de forma relativa, mientras que el scroll no haga que dicho elemento sobrepase la parte superior. Cuando eso suceda, quedará fijado en la parte superior de la ventana para evitar que desaparezca por efecto del scroll.

Elementos flotantes

Otra de las características más importantes de los bloques para realizar la maquetación, son los elementos flotantes.

Al igual que se vio para las imágenes en la unidad anterior, es posible usar la propiedad float para hacer que los bloques sean flotantes. De esta manera, se consigue alterar el flujo normal de los elementos en una página web.

NOTA DE INTERÉS

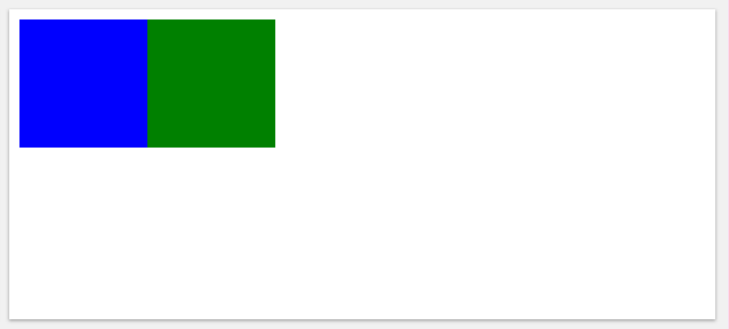
Los elementos posicionados de forma absoluta ignorarán la propiedad float.

Los elementos flotantes, se van apilando a uno de los lados, mientras haya espacio de ancho en la página. En cuanto no quede espacio suficiente, los siguientes elementos pasarán a colocarse debajo de los anteriores.

EJEMPLO PRÁCTICO

A continuación, se muestran dos bloques que flotan a la izquierda. En lugar de aparecer uno debajo de otro, aparecerán al lado.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        div {width:100px;height:100px;float:left;}
    </style>
<body>
    <div style="background-color:blue;">
    </div>
    <div style="background-color:green;">
    </div>
</body>
</html>
```

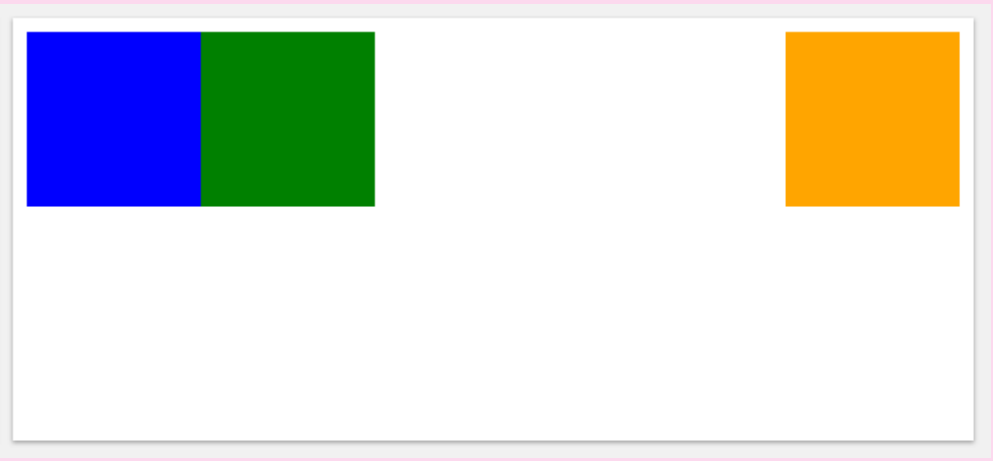


También es posible incluir elementos flotantes a ambos lados, como se muestra en el siguiente ejemplo:

EJEMPLO PRÁCTICO

En esta ocasión, se añade un bloque más al caso anterior, pero esta vez flotando a la derecha.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        div {width:100px;height:100px;float:left;}
    </style>
<body>
    <div style="background-color:blue;">
    </div>
    <div style="background-color:green;">
    </div>
    <div style="background-color:orange;float:right;">
    </div>
</body>
</html>
```



Es posible forzar que los elementos dejen de flotar sobre los anteriores. Para ello, se debe emplear la propiedad clear. Si se ajusta a both, tendrá en cuenta los elementos flotantes a ambos lados para aplicar esta restricción. Si se ajusta a left o a right, sólo considerará los elementos flotantes a izquierda o derecha, respectivamente.

EJEMPLO PRÁCTICO

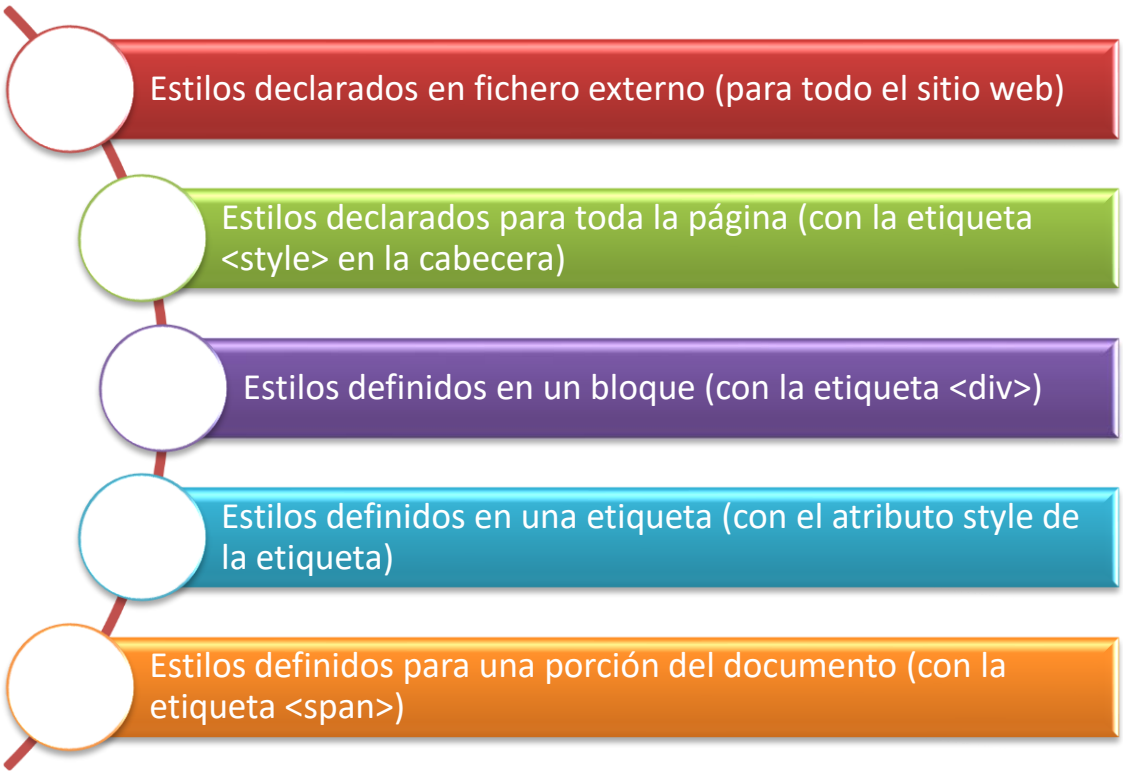
En este ejemplo, se añade la propiedad clear:both al último de los bloques del caso anterior. Como los bloques anteriores flotan a la izquierda, el resultado sería el mismo si se escribiera float:left.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        div {width:100px;height:100px;float:left;}
    </style>
<body>
    <div style="background-color:blue;">
    </div>
    <div style="background-color:green;">
    </div>
    <div style="background-color:orange;float:right;clear:both;">
    </div>
</body>
</html>
```

Reglas para la aplicación de estilos

Como se vio anteriormente, los estilos se heredan de una etiqueta a otra que esté dentro de su ámbito. Por ejemplo, si hay unos determinados estilos declarados para la etiqueta <body>, dichas reglas también afectarán a etiquetas que estén dentro de <body>, es decir, dentro de todo el cuerpo. Esto hace que, en muchas ocasiones, un mismo elemento se vea afectado por más de una declaración de estilos. En el ejemplo anterior, para un párrafo se aplicaría tanto lo definido para <body> como lo definido para <p>. En esos casos, siempre se tiene en cuenta la declaración más específica.

No obstante, como hay diferentes modos de declarar estilos (en un fichero externo, en línea, etc.), también se deberá tener esto en cuenta a la hora de decidir cuál de las reglas que actúan sobre el mismo elemento será la que resulte efectiva. A continuación, se puede ver el orden de aplicación de las reglas, de más generales a más específicas:



Como se ha comentado, CSS significa hoja de estilos en cascada. Pues bien, la cascada es el mecanismo por el que se decide qué regla es la que se aplica finalmente sobre un elemento. Para ello se tiene en cuenta tanto la especificidad que se acaba de mencionar, como la importancia de la regla y el orden en el que aparecen declaradas en el código.

En referencia a los selectores a utilizar, encontramos:

1. De tipo

Se trata de los selectores vistos hasta el momento, formados únicamente por un nombre de etiqueta. Todos los elementos cuya etiqueta coincida con el selector, se verán afectados por esa regla:

etiqueta {declaraciones_reglas}

2. Universal

El selector universal afecta a todos los elementos HTML. No se suele usar por sí solo sino como parte de otros selectores compuestos, como se verá más adelante. Se representa mediante un asterisco:

*** {declaraciones_reglas}**

3. Múltiple

Un selector múltiple es un selector compuesto en el que se encadenan varios selectores simples para indicar que sigan todos ellos un mismo conjunto de reglas. Se especifica separando los selectores simples con comas.

Se usa para no repetir las mismas reglas varias veces para distintos selectores. Por ejemplo, si se quiere especificar que todos los títulos desde h1 hasta h3 tengan unas mismas características determinadas, según se ha visto hasta ahora, se debe escribir:

```
h1 { font-family:arial; color:blue; }
h2 { font-family:arial; color:blue; }
h3 { font-family:arial; color:blue; }
```

Lo cual resulta muy redundante, por lo que lo mejor es usar un selector múltiple, con lo que queda reducido a:

```
h1, h2, h3 { font-family:arial; color:blue;}
```

Como es lógico, no es habitual encontrarse con casos en los que varios elementos estén regidos por exactamente las mismas reglas. Sin embargo, sí suele haber un conjunto de reglas comunes para muchos elementos. En ese caso, lo que se suele hacer es usar un selector múltiple para agrupar esas reglas comunes, seguido de selectores simples para añadir las reglas particulares a cada caso. Así, si las reglas fueran:

```
h1 { font-family:arial; color:blue; font-size:20px; }
h2 { font-family:arial; color:blue; font-size:18px; }
h3 { font-family:arial; color:blue; font-size:16px; }
```

Se pueden escribir:

```
h1, h2, h3 { font-family:arial; color:blue;}
h1 { font-size:20px; }
h2 { font-size:18px; }
h3 { font-size:16px; }
```

4. Descendiente

Se trata de un selector compuesto formado por más de un selector simple encadenado. De esta forma, se restringe la aplicación de la regla a los elementos que se encuentren tras la aplicación sucesiva de dichos selectores simples. Por tanto, sirven para seleccionar etiquetas que están dentro de otras determinadas etiquetas.

Se representa mediante selectores simples separados por espacios (sin comas). El orden de aplicación de los selectores es de izquierda a derecha.

Por ejemplo, si se quiere asignar un color diferente a las negritas de un párrafo, se deberá escribir:

```
p b {color:blue;}
```

Así, el resto de las negritas que aparezcan en la página no se verán afectadas. Se puede ver claramente en el siguiente ejemplo:

EJEMPLO PRÁCTICO

Uso del selector descendente para cambiar el color de las negritas de los párrafos. Como se puede observar, las del título permanecen en negro.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        p b {color:blue;}
    </style>
</head>
<body>
   <h1>Título con <b>negritas</b></h1>
   <h2>Subtítulo</h2>
   <p>Texto con <b>negritas</b></p>
   <p>Otro párrafo con <i>más <b>negritas</b></i></p>
</body>
</html>
```

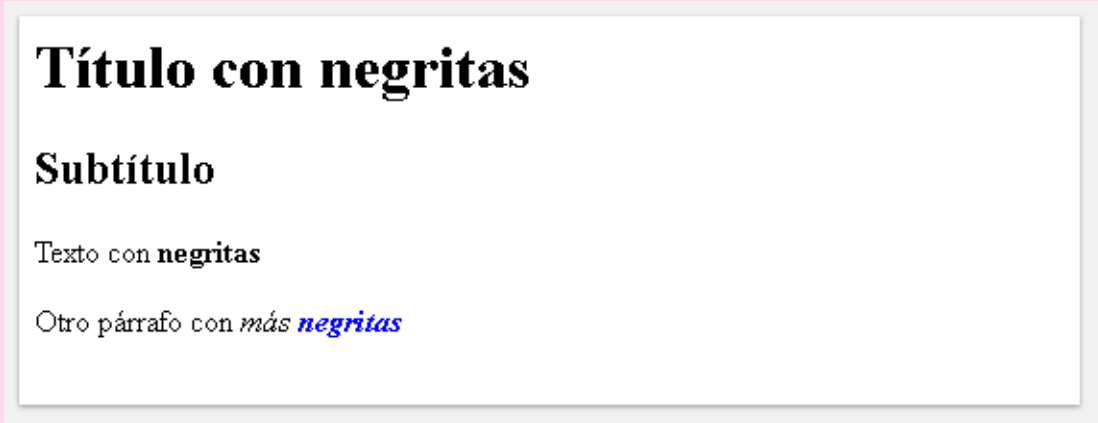
Es importante tener en cuenta que en este tipo de selectores no se indica que los elementos del tipo que aparece en segundo lugar deban ser descendientes directos del tipo que aparece en primer lugar. En el ejemplo, se puede ver cómo se ven afectados tanto la primera palabra en negrita dentro del párrafo, como la segunda, que se encuentra a un nivel inferior, bajo la etiqueta <i>.

El número de selectores simples que pueden aparecer en un selector descendente puede ser cualquiera, con lo que se conseguirá restringir más la aplicación de la regla. Siguiendo el ejemplo anterior, si se añade la etiqueta <i> a la regla se conseguirá que sólo la segunda palabra se vea afectada:

EJEMPLO PRÁCTICO

Este caso es similar al anterior, pero ahora el selector incluye también la etiqueta i, por lo que sólo se mostrará en negritas la palabra del segundo párrafo, que también está en cursiva.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        p i b {color:blue;}
    </style>
</head>
<body>
   <h1>Título con <b>negritas</b></h1>
   <h2>Subtítulo</h2>
   <p>Texto con <b>negritas</b></p>
   <p>Otro párrafo con <i>más <b>negritas</b></i></p>
</body>
</html>
```



En estos casos, también se pueden usar selectores universales para indicar de forma general que los elementos no sean descendientes directos de otros. Se trata de usar el asterisco entre las etiquetas que no se quiere que sean descendientes directas, en lugar de usar una etiqueta concreta como se hacía en el ejemplo anterior:

```
p * b {color:blue;}
```

El efecto en el ejemplo es el mismo, pero funcionaría de forma general para todas las negritas que no sean descendientes directas de un párrafo, por lo que si hubiera negritas dentro del ámbito de otra etiqueta diferente de <i>, también se verían afectadas:

EJEMPLO PRÁCTICO

Ejemplo de uso del selector universal para seleccionar descendientes no directos.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        p * b {color:blue;}
    </style>
</head>
<body>
   <h1>Título con <b>negritas</b></h1>
   <h2>Subtítulo</h2>
   <p>Texto con <b>negritas</b></p>
   <p>Otro párrafo con <i>más <b>negritas</b></i> y <s><b>otras
más</b></s>, que también se ven afectadas</p>
</body>
</html>
```

5. De clase

Hasta ahora se han definido selectores genéricos que no permiten diferenciar entre diferentes elementos definidos por la misma etiqueta. Sin embargo, no es habitual que, por ejemplo, todos los párrafos de una misma página sean iguales. Una forma de asignarles estilos diferentes es mediante el uso de estilos en la propia etiqueta. No obstante, como se vio anteriormente, no es una práctica recomendable.

Para solucionar esto, se utilizan clases para poder diferenciar unos elementos de otros independientemente de la etiqueta que los defina.

Las clases son atributos de cualquier elemento HTML, que se definen mediante la palabra class:

<etiqueta class="nombredelaclase">

Las clases no sólo sirven para diferenciar unos elementos de otros, sino también para crear definiciones de estilos que se pueden utilizar repetidas veces. Será suficiente con definir un conjunto de reglas que podrán ser aplicadas a diferentes elementos sin importar sus etiquetas.

La sintaxis del selector de clase consiste en un punto seguido del nombre de la clase:

.nombredelaclase {declaraciones_reglas}

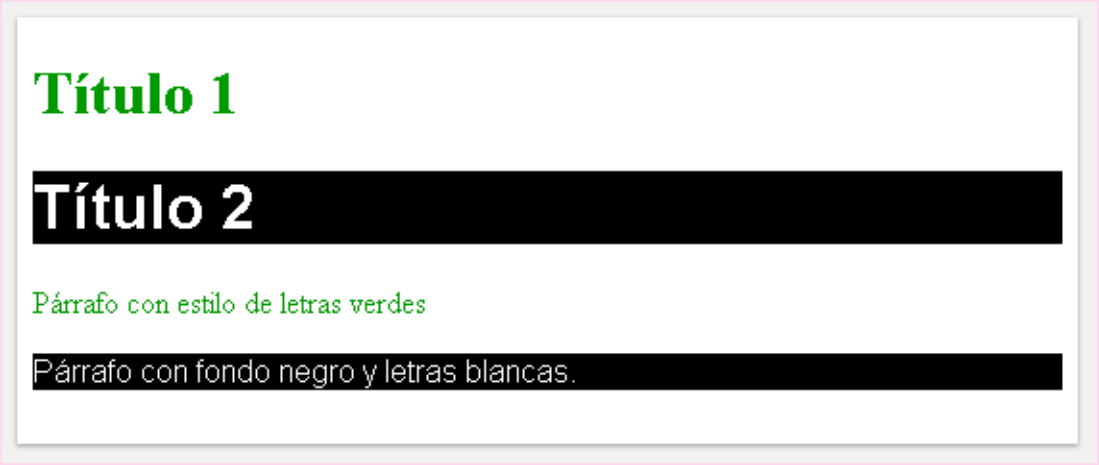
De esta forma, un párrafo definido mediante <p class="parrafo1"> podrá diferenciarse de otro definido por <p class="parrafo2">. A los dos párrafos les afectarán las reglas definidas por p, pero además al primero le afectarán las definidas por parrafo1 y al segundo las definidas por parrafo2.

EJEMPLO PRÁCTICO

Ejemplo de uso de selectores de clase en el que se definen dos: una para letras verdes y otra para letras blancas sobre fondo negro. Los estilos definidos se pueden aplican a diferentes elementos, como h1 y p:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        .fondonegroletrasblancas
        {
            background-color: black;
            color: white;
            font-size: 12;
            font-family: arial;
        }
        .letrasverdes
        {
            color: #009900;
        }
    </style>
</head>
<body>
    <h1 class="letrasverdes">
```

```
        Título 1</h1>
    <h1 class="fondonegroletrasblancas">
        Título 2</h1>
    <p class="letrasverdes">
        Párrafo con estilo de letras verdes</p>
    <p class="fondonegroletrasblancas">
        Párrafo con fondo negro y letras blancas.</p>
</body>
</html>
```



Como se puede observar en el ejemplo, una misma clase puede asignarse a diferentes tipos de elemento, como son un párrafo y un título. Además, es posible aplicar una regla sólo para unos elementos determinados. De esta forma, se podrían definir diferentes características para un párrafo de la clase letrasverdes y para un título de esa misma clase. La forma de hacerlo es añadiendo el nombre de la etiqueta justo antes del punto (sin espacio):

etiqueta.clase {declaraciones_reglas}

Su funcionamiento se puede comprobar en el siguiente ejemplo, en el que los elementos de la clase rojo se ven en rojo, pero de los de la clase azul, sólo se ve azul el párrafo, ya que es la única etiqueta para la que está definida la regla:

EJEMPLO PRÁCTICO

Ejemplo de uso de selector de clase para una etiqueta concreta. En este caso, no debe aparecer ningún especio entre ambos nombres:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        p.azul {color:blue;}
        .rojo {color:red;}
        </style>
</head>
<body>
    <h1 class="azul">ejemplo de selectores de clase</h1>
    <h2 class="rojo">con colores</h2>
    <p>párrafo 1</p>
    <p class="azul">párrafo2</p>
    <p class="rojo">párrafo3</p>
</body>
</html>
```

Es importante tener en cuenta que no se trata de un selector descendente, ya que tendría que existir un espacio entre el nombre de la etiqueta y el punto para que así fuera. Así, si en el ejemplo anterior se escribiera la regla p .azul, ésta actuaría sobre las clases azul descendientes de algún elemento de tipo p. Como el párrafo2 no es descendiente de otro elemento p, la regla no se aplicará, por lo que no se verá en azul:

EJEMPLO PRÁCTICO

Este caso es similar al anterior, pero con un espacio entre ambos nombres, por lo que, en vez de ser un selector de clase para un elemento determinado, se convierte en un selector descendente. Por eso, párrafo2 no se ve afectado por la regla azul.

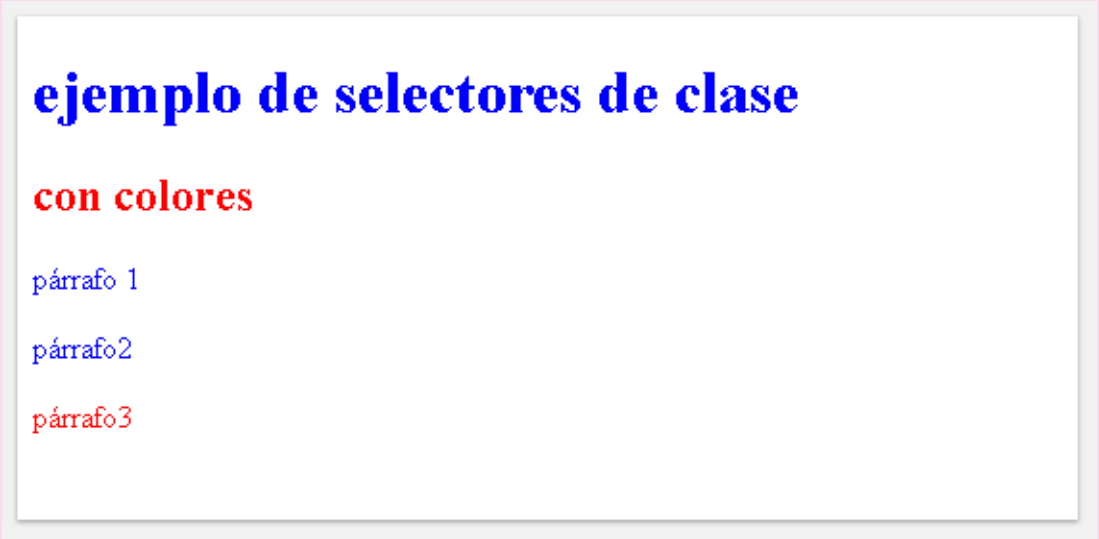
```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        p .azul {color:blue;}
        .rojo {color:red;}
        </style>
</head>
<body>
    <h1 class="azul">ejemplo de selectores de clase</h1>
    <h2 class="rojo">con colores</h2>
    <p>párrafo 1</p>
    <p class="azul">párrafo2</p>
    <p class="rojo">párrafo3</p>
</body>
</html>
```

También se debe tener cuidado con no confundir este selector con el selector múltiple, que se definiría con una coma entre el nombre de la etiqueta y el punto. En el ejemplo sería p,.azul y el resultado sería también diferente ya que esta regla lo que está haciendo es definir ese estilo para todos los párrafos y para todos los elementos cuya etiqueta sea azul:

EJEMPLO PRÁCTICO

Siguiendo el mismo ejemplo anterior, ahora se separan los nombres del selector con comas, por lo que, lo que se está definiendo es un selector múltiple. De esta forma, todos los párrafos y elementos con clase azul se verán afectados por dicha regla.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        p,.azul {color:blue;}
        .rojo {color:red;}
        </style>
</head>
<body>
    <h1 class="azul">ejemplo de selectores de clase</h1>
    <h2 class="rojo">con colores</h2>
    <p>párrafo 1</p>
    <p class="azul">párrafo2</p>
    <p class="rojo">párrafo3</p>
</body>
</html>
```



Una misma etiqueta puede tener asignadas varias clases. Para ello, se especifican varios nombres separados por espacios, como valor del atributo class:

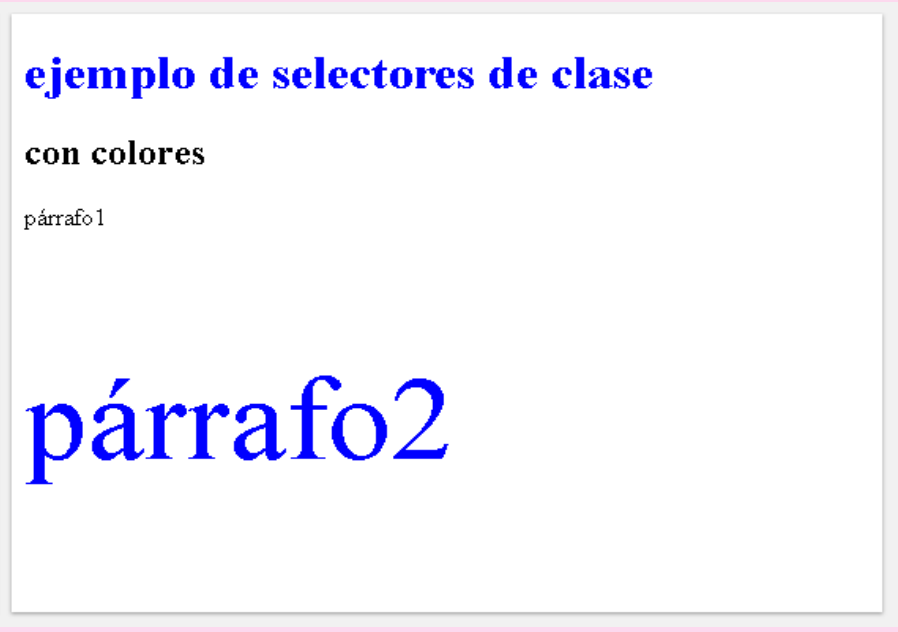
<etiqueta class="clase1 clase2 clase3…">

Ahora, los elementos se verán afectados por las reglas definidas para todas esas clases:

EJEMPLO PRÁCTICO

En este ejemplo, se definen varias clases que luego se aplican a un mismo elemento, en este caso, el segundo párrafo, que se verá afectado tanto por la regla azul como por la regla grande.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        .azul{color:blue}
        .grande{font-size:500%}
    </style>
</head>
<body>
    <h1 class="azul">ejemplo de selectores de clase</h1>
    <h2>con colores</h2>
    <p>párrafo1</p>
    <p class="azul grande">párrafo2</p>
</body>
</html>
```



6. De id

De la misma forma que es posible definir atributos de clase, todos los elementos HTML pueden tener definido un identificador o atributo id. La diferencia entre identificadores y clases es que los primeros deben ser únicos para todo el documento HTML. Por lo demás, la función de un identificador es similar a la de una clase, permitiendo aplicar estilos a un elemento con un identificador determinado.

<etiqueta id="nombredelidentificador">

Los selectores se definen también de forma parecida a los de clase, pero usando en este caso el símbolo almohadilla en lugar del punto:

#identificador {declaraciones_reglas}

El funcionamiento es el mismo que para las clases, pero se debe tener en cuenta que **cada identificador debe pertenecer a un único elemento** en una misma página. Por tanto, estos selectores se utilizan para dar estilo únicamente a un elemento, mientras que los selectores de clase están pensados para dar estilo a un conjunto de elementos.

EJEMPLO PRÁCTICO

Ejemplo que combina el uso de selectores de id y de clase.

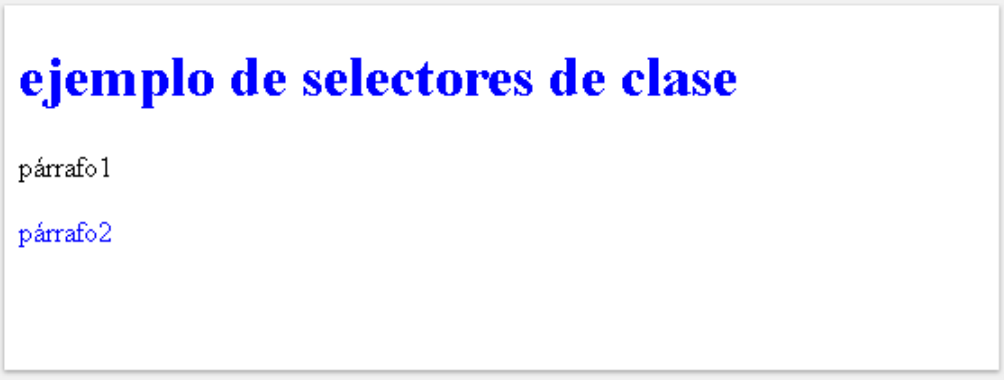
```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        #tituloprincipal{font-size:32px}
        .azul{color:blue}
    </style>
</head>
<body>
    <h1 id="tituloprincipal">ejemplo de selectores de clase</h1>
    <p>párrafo1</p>
    <p class="azul">párrafo2</p>
</body>
</html>
```

También es posible combinar el uso de identificadores y clases en un mismo elemento:

EJEMPLO PRÁCTICO

Ejemplo que combina selectores de id y de clase en un mismo elemento.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        #tituloprincipal{font-size:500%}
        .azul{color:blue}
    </style>
</head>
<body>
    <h1 id="tituloprincipal" class="azul">ejemplo de selectores de
clase</h1>
    <p>párrafo1</p>
    <p class="azul">párrafo2</p>
</body>
</html>
```



7. De atributo

Aunque no son tan comunes como los anteriores, también es posible usar atributos para discriminar, en lugar de clases e identificadores. La sintaxis es:

[atributo] {declaraciones_reglas}

Así, las reglas se aplicarán sólo a los elementos que tengan ese atributo. Normalmente, se especifica un valor para ese atributo, de la siguiente forma:

[atributo=valor] {declaraciones_reglas}

Por ejemplo, [target="_blank"] seleccionará sólo los enlaces que se abran en una nueva pestaña o ventana.

8. Otros selectores

Se pueden construir otros selectores en base a los pseudoelementos y las pseudoclases. Se usan añadiendo el nombre del pseudoelemento o la pseudoclase a un selector simple.

- Pseudoelementos

Sirven para referirse a una determinada parte del documento como, por ejemplo, la primera línea de un párrafo. Se expresan por medio de un doble signo de dos puntos seguido del nombre del pseudoelemento:

::pseudoelemento

Para construir el selector, se escribe detrás de la etiqueta a que se deba referir:

etiqueta::pseudoelemento {declaraciones_reglas}

Por ejemplo, para poner en azul la primera línea de un párrafo, sería:

p::first-line {color:blue}

Se debe tener en cuenta que la distinción entre pseudoelementos y pseudoclases, aparece en CSS3, por lo que es posible encontrarse con los mismos pseudoelementos siguiendo la notación de las pseudoclases, es decir, solamente con dos puntos.

- Pseudoclases

Las pseudoclases se usan para hacer referencia a diferentes estados de un elemento. De esta forma, es posible saber cuándo un enlace ha sido visitado, o si el ratón está situado sobre un elemento, etc. Se expresan mediante el signo de dos puntos seguido del nombre de la pseudoclase:

:pseudoclase

Al igual que con los pseudoelementos, los selectores construidos a partir de pseudoclases se construyen escribiendo ésta detrás de la etiqueta a la que se quiera referir:

etiqueta:pseudoclase {declaraciones_reglas}

El uso más común de las pseudoclases es sobre los enlaces, permitiendo cambiar su apariencia según el estado en el que se encuentren:

- **Enlaces normales:** los que aún no se han visitado.
    - a:link {…}

- **Enlaces visitados:** en los que se ha hecho clic previamente.
    - a:visited {…}

- **Enlaces activos:** cuando se está haciendo clic en ellos.
    - a:active {…}

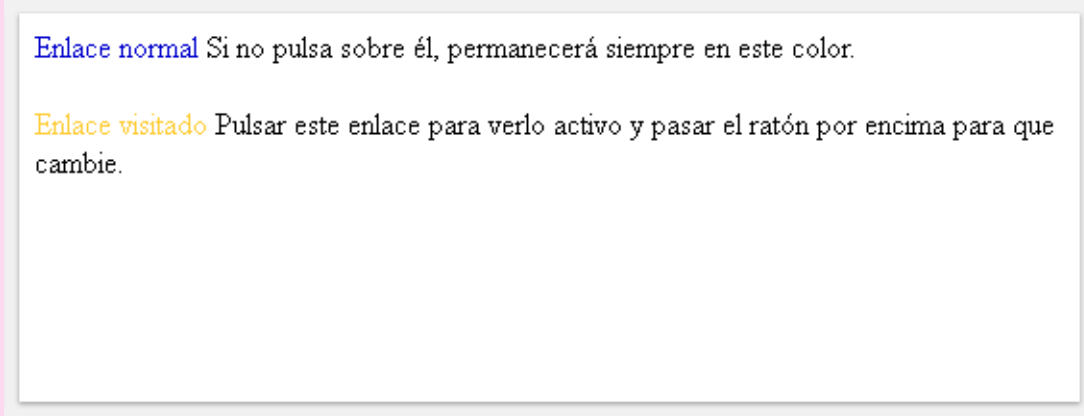
- **Enlaces hover:** cuando el ratón está encima de ellos.
    - a:hover {…}

Por ejemplo, para quitar el subrayado por defecto de los enlaces, se puede usar la propiedad text-decoration con el valor none, y asignársela a todos los estados excepto cuando el ratón pase por encima del enlace, para ayudar a identificarlo como tal. A continuación, se muestra el resultado añadiendo también un color diferente a cada uno de los estados para diferenciarlos claramente:

EJEMPLO PRÁCTICO

Uso de pseudoclases para cambiar el estilo de los enlaces según su estado.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        a:link
        {
            text-decoration: none;
            color: #0000cc;
        }
        a:visited
        {
            text20: ddecoration:none;color:#ffcc33;}
        a:active
        {
            text-decoration: none;
            color: #ff0000;
        }
        a:hover
        {
            text-decoration: underline;
            color: #999999;
            fontweight: bold;
        }
    </style>
</head>
<body>
    <a href="http://example.com">Enlace normal</a> Si no pulsa sobre
él, permanecerá siempre en este color.
    <br>
    <br>
    <a href="enlaces.html" target="_blank" >Enlace visitado</a>
Pulsar este enlace para verlo activo y pasar el ratón por encima
para que cambie.
</body>
</html>
```



5.3 El modelo de cajas

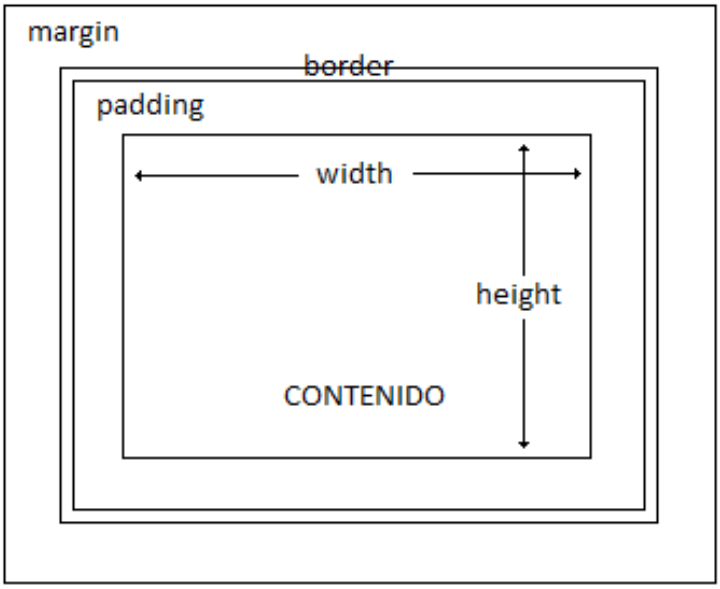
El modelo de cajas es el modelo que utiliza CSS para representar sus elementos, por lo que entender su funcionamiento es fundamental para realizar la maquetación de una web.

Todos los elementos que se pueden encontrar dentro de una página HTML, están contenidos en una caja rectangular con las siguientes partes:

- **Contenido**: donde aparece todo lo que contenga el elemento.
- **Relleno o margen interno**: el espacio entre el contenido y el borde.
- **Borde**: el borde que enmarca al elemento.
- **Margen**: el espacio que se mantendrá entre el borde y otra caja adyacente.

Las propiedades width y height son las que se utilizan para asignar un tamaño al área de contenido.

El borde, por defecto, tendrá un tamaño de 0, por lo que será invisible. Para ajustarlo, se debe usar la propiedad border vista en la unidad anterior, o una de las propiedades específicas para cada uno de los bordes (border-left, border-right, border-top, border-bottom).
Para el margen y el relleno, las propiedades son margin y padding, respectivamente, pudiendo especificarlas igualmente para cada uno de los lados, usando la propiedad terminada en -left, -right, -top o -bottom, de forma similar a border.



Margen y relleno

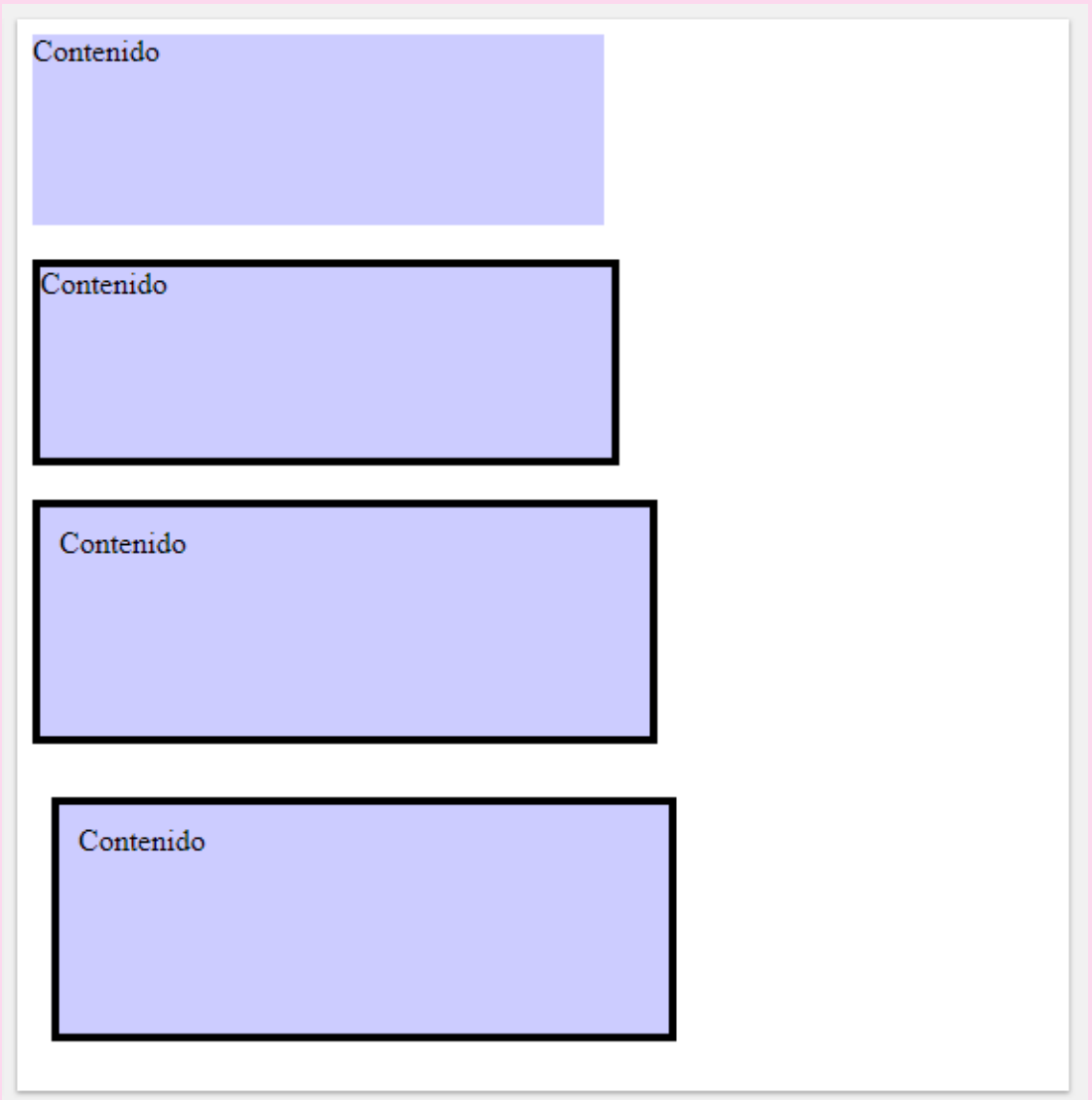
Es importante tener en cuenta que el modelo de caja, por defecto, suma todas las partes de las que se compone para calcular el tamaño real que ocupa en pantalla. Es decir, en el maquetado, el tamaño que ocupará cada caja en realidad es igual al tamaño del contenido más el del padding, más el del borde, más el del margen. Se puede observar fácilmente en el siguiente ejemplo, que añade sucesivamente a un div, cada una de estas propiedades:

EJEMPLO PRÁCTICO

Este ejemplo, muestra cómo se va incrementando progresivamente el tamaño de la caja al añadir cada una de sus posibles componentes.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
<style>
#caja1 {
  width:300px;
  height:100px;
  background-color:#CCF;
}
#caja2 {
  width:300px;
  height:100px;
  background-color:#CCF;
  border:4px solid black;
}
#caja3 {
  width:300px;
  height:100px;
  background-color:#CCF;
  border:4px solid black;
  padding:10px;
}
#caja4 {
  width:300px;
  height:100px;
  background-color:#CCF;
  border:4px solid black;
  padding:10px;
  margin:10px;
}
</style>
<body>
<div id="caja1">Caja</div>
<br>
<div id="caja2">Caja con borde</div>
<br>
<div id="caja3">Caja con borde y padding</div>
<br>
<div id="caja4">Caja con borde, padding y margin</div>
```

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



5.4 Unidades de medida

Gran parte de las reglas CSS hacen uso de medidas para expresar posiciones, tamaños y distancias con otros elementos.

Existen numerosas unidades en las que expresar estas medidas, pero pueden clasificarse en dos grandes grupos: absolutas y relativas.

- Absolutas

Las medidas expresadas en estas unidades no se verán afectadas por las medidas de otros elementos, sino que permanecerán siempre igual:

- **cm**: centímetros.
- **mm**: milímetros.
- **in**: pulgadas (= 2.54cm).
- **px**: píxeles. Aunque se considere como unidad absoluta, depende de la resolución del dispositivo utilizado ya que, en ocasiones, un píxel puede estar formado por varios píxeles reales del dispositivo.
- **pt**: puntos (1/72 pulgadas).
- **pc**: picas (12 puntos).

Los píxeles suelen utilizarse para indicar medidas en pantallas mientras que el resto están más ligados a impresión de documentos.

Por lo general, no resulta aconsejable usar estas medidas a no ser que sea para un dispositivo concreto.

- Relativas

Las unidades relativas hacen que las medidas cambien en relación con las de otros elementos. Su uso es aconsejable para conseguir un diseño adaptable. Son:

- **%**: porcentaje relativo al tamaño del elemento padre.
- **em**: medida relativa al tamaño de la fuente del elemento. Por ejemplo, 2em indicaría un tamaño doble que el de la fuente.
- **rem**: relativo al tamaño de la fuente del elemento raíz.
- **ex**: es similar a em, pero tomando como referencia el tamaño de la letra x.
- **ch**: relativa al ancho del dígito cero.
- **vw**: relativo al 1% del ancho del área visible de la página (viewport).
- **vh**: relativo al 1% de la altura del área visible de la página (viewport).
- **vmin**: relativo al 1% de la menor de las dimensiones del viewport.
- **vmax**: relativo al 1% de la mayor de las dimensiones del viewport.

Por ejemplo, una medida con porcentajes se puede usar para hacer que una imagen se adapte al tamaño del elemento en el que está contenida. De esta forma, puede mostrarse al completo, aunque su resolución sea mayor que la de la pantalla:

EJEMPLO PRÁCTICO

En este ejemplo, se incluye una imagen dentro de un párrafo, haciendo que ocupe un ancho del 100% del tamaño del párrafo.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        p{background-color:#ccc}
    </style>
</head>
<body>
    <p>Párrafo que contiene una imagen
        <br>
        <img src="playa.jpg" width="100%" alt="playa">
    </p>
</body>
</html>
```

Y si se especifica un tamaño del 50%, quedaría:

EJEMPLO PRÁCTICO

Caso similar al anterior, pero con un tamaño para la imagen del 50%.

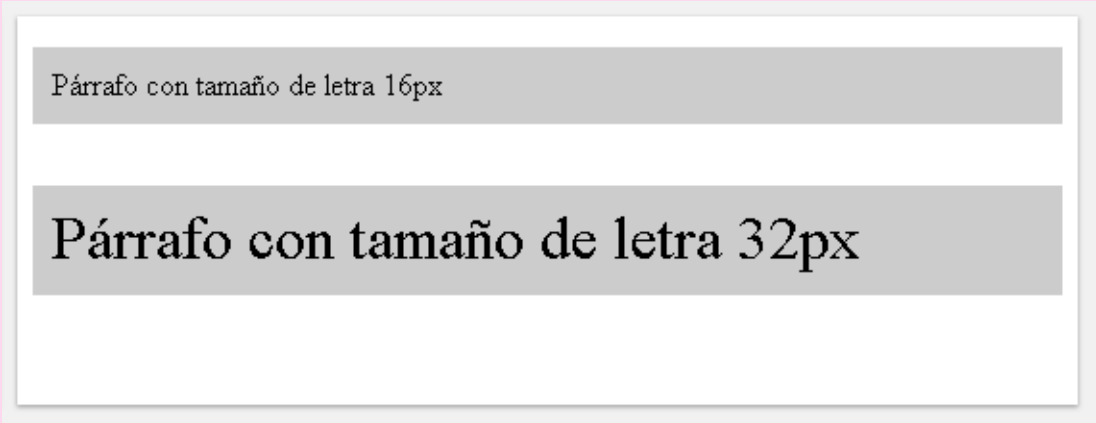
```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        p{background-color:#ccc}
    </style>
</head>
<body>
    <p>Párrafo que contiene una imagen
        <br>
        <img src="playa.jpg" width="50%" alt="playa">
    </p>
</body>
</html>
```

Muchas veces, resulta conveniente usar medidas relativas al tamaño de la letra. Por ejemplo, si se establece un margen para el texto de un párrafo, puede ser interesante que, si se aumenta el tamaño de letra, el margen aumente proporcionalmente. En estos casos se usan las unidades relativas al tamaño de la fuente presentadas en el punto anterior, siendo em y rem las más usuales. En el siguiente ejemplo, se aumenta el tamaño de letra del segundo párrafo, considerando un margen dado por unidades absolutas:

EJEMPLO PRÁCTICO

Uso de unidades absolutas para definir distintos tamaños de fuente.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        p{background-color:#ccc;padding:10px;}
        .parrafo1{font-size:16px;}
        .parrafo2{font-size:32px;}
    </style>
</head>
<body>
    <p class="parrafo1">Párrafo con tamaño de letra 16px</p>
    <p class="parrafo2">Párrafo con tamaño de letra 32px</p>
</body>
</html>
```

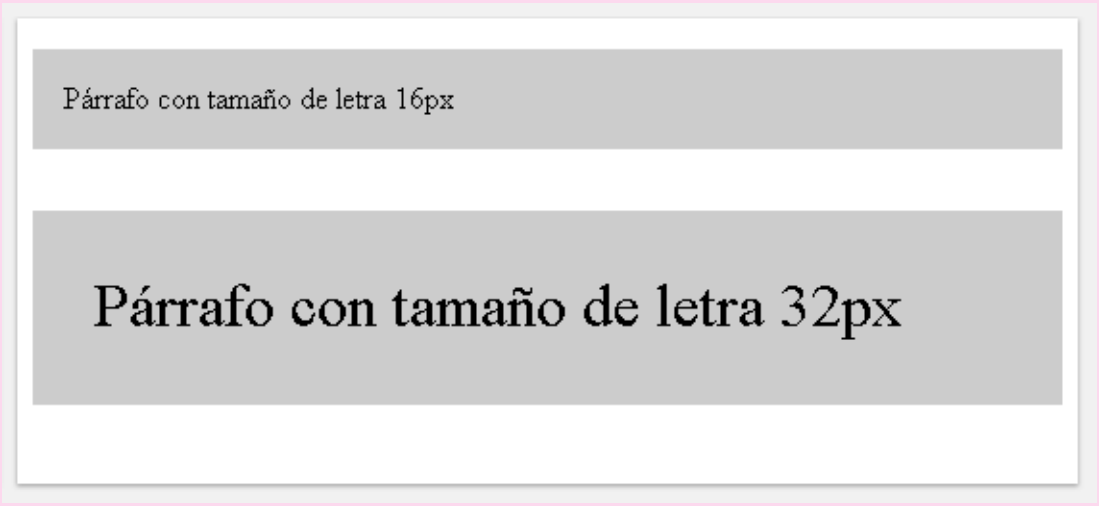


Sin embargo, si se usan medidas relativas para definir el margen, se adapta automáticamente al cambio de tamaño de la fuente:

EJEMPLO PRÁCTICO

Ejemplo similar al anterior, pero cambiando el padding absoluto por uno relativo.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        p{background-color:#ccc;padding:1em;}
        .parrafo1{font-size:16px;}
        .parrafo2{font-size:32px;}
    </style>
</head>
<body>
    <p class="parrafo1">Párrafo con tamaño de letra 16px</p>
    <p class="parrafo2">Párrafo con tamaño de letra 32px</p>
</body>
</html>
```

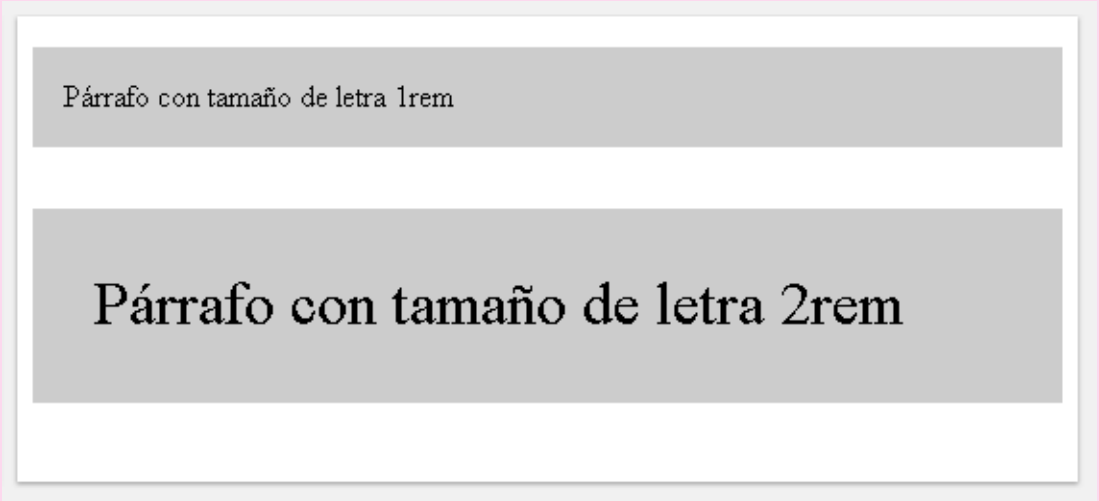


En el caso de rem, las medidas son relativas al tamaño de fuente del elemento raíz, es decir, de <html>. Además, puede combinarse con em para realizar diseños más elaborados. En el siguiente ejemplo se consigue el mismo resultado anterior, pero con todas las medidas relativas.

EJEMPLO PRÁCTICO

Ejemplo similar al anterior, pero cambiando las unidades absolutas para el tamaño de las fuentes por relativas.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        html {font-size:16px;}
        p{background-color:#ccc;padding:1em;}
        .parrafo1{font-size:1rem;}
        .parrafo2{font-size:2rem;}
    </style>
</head>
<body>
    <p class="parrafo1">Párrafo con tamaño de letra 1rem</p>
    <p class="parrafo2">Párrafo con tamaño de letra 2rem</p>
</body>
</html>
```



Em y rem son las unidades más usadas para conseguir páginas web escalables, es decir, páginas que adaptan su tamaño a la pantalla. Se trata de una parte del diseño responsive, que se lograría tan sólo cambiando el tamaño de fuente del elemento raíz (como en el ejemplo anterior) según el dispositivo en el que se viera la página.

No obstante, el diseño responsive va más allá de la simple escalabilidad, transformando en ocasiones la disposición de los elementos en la página por completo o incluso añadiendo nuevos.

En primer lugar, es necesario saber qué es el **viewport**. Se trata del área visible de la página web. Por tanto, su tamaño podrá variar según el dispositivo de visualización.

Para usarlo, lo primero que se debe hacer es definirlo. Para ello, se utiliza la etiqueta <meta>, como ya se adelantó en la unidad anterior. Como nombre de la etiqueta, se deberá poner viewport y como contenido, una lista de parámetros separados por comas. Mediante estos parámetros, se puede ajustar el ancho (width), alto (height) y la escala de dicho viewport (initial-scale, minimun-scale y maximum-scale). Por ejemplo:
<meta name="viewport" content="width=device-width, initial-scale=1.0">

En este caso, el ancho se ajusta al ancho del dispositivo, y la escala se ajusta a 1, es decir, sin escalar. También se puede usar el parámetro user-scalable=no si se quiere evitar que el usuario pueda escalar la página.
En el siguiente ejemplo, se define un tamaño de fuente relativo al viewport. De esta manera, el texto ocupará proporcionalmente el mismo tamaño de la ventana, tanto si se ve en un dispositivo como en otro. En el primer caso se usa vw para hacerlo relativo al ancho del viewport, y en el segundo vh para hacerlo relativo a la altura del viewport.

Al usar estas medidas, no se sabe cómo es la forma del viewport, es decir, si es más alto que ancho o viceversa. En algunos casos, puede ser interesante conocer esta información para representar los elementos en la pantalla. En tal caso, se usan las unidades vmin y vmax, especialmente la primera, que cuenta con mayor soporte por parte de los navegadores. Por tanto, si el ancho del viewport es menor que el alto, entonces vmin será igual que vw. Por el contrario, si el alto del viewport es menor que el ancho, entonces vmin será igual que vh. Por ejemplo, si el viewport es de 800px por 600px, vmin valdrá 6px y si es de 600px por 800px, su valor será igualmente 6px.

5.5 Atributos comunes de las hojas de estilos

A continuación, se incluyen una serie de tablas con algunos de los atributos más comunes que se usan en las hojas de estilo, según su categoría.

Atributos para fuentes

Color	Valor o nombre del color	color: #000; color: black;
Define el color del texto.		
font-size	xx-small | x-small | small | medium | large | x-large | xx-large	font-size
Asigna un tamaño a la fuente.		
font-family	Nombre de la fuente.	font-family: arial,sans-serif;
Asigna un tipo de fuente. Si no está instalada en el sistema, debe vincularse. Si se incluyen varias, se toma la primera que se encuentre.		
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900	font-weight: 300;
Establece el grosor de la fuente. Bold se utiliza para poner negritas. Equivale a 700, mientras que normal equivale a 400.		
font-style	normal | italic | oblique	font-style: italic;
Da estilo a la fuente. Se utiliza fundamentalmente para escribir en cursiva (italic).		
font	font-style font-variant font-weight font-size/line-height font-family|caption|icon|menu|message-box|small-caption|status-bar|initial|inherit;	font: italic bold 14px arial, sans-serif;
Ajusta las principales propiedades de la fuente de una sola vez.		

Atributos para texto

line-height	normal Unidades de medida CSS	line-height: 14px;
Ajusta la altura de una línea de texto.		
text-decoration	none | [underline || overline || line-through]	text-decoration: underline;
Añade un efecto al texto (subrayado, tachado, etc.). También se le puede asignar un color y un estilo (sólido, punteado, etc.)		
text-align	left | right | center | justify	text-align: justify;
Alinea el texto.		
text-indent	Unidades de medida CSS	text-indent: 30px;
Define un sangrado para el texto.		
text-transform	capitalize | uppercase | lowercase | none	text-transform: uppercase;

Transforma el texto a mayúsculas o minúsculas, independientemente de cómo esté escrito en el archivo HTML.		
vertical-align	baseline | sub | super | top | text-top | middle | bottom | text-bottom	vertical-align: middle;
Alinea el texto en vertical.		

Atributos para fondo

background-color	Valor o nombre del color	background-color: #000; background-color: black;
Asigna un color al fondo.		
background-image	url(...) | none	background-image: url('imagen.jpg'); background-image: none;
Pone una imagen como fondo.		
background-repeat	repeat | repeat-x | repeat-y | no-repeat	background-repeat: repeat;
Indica si la imagen de fondo debe repetirse como un mosaico. Repeat indica que lo haga tanto en horizontal como en vertical. Repeat-x sólo en horizontal y repeat-y en vertical.		
background-position	left | right |center + top | bottom |center Unidades de medida CSS	background-position: right center; background-position: 40px 20px;
Establece la posición de la imagen de fondo. Por defecto, se coloca arriba a la izquierda.		
background	bg-color bg-image position/bg-size bg-repeat bg-origin bg-clip bg-attachment	background: black url("imagen.jpg") repeat right center;
Para agrupar las propiedades del fondo de una sola vez.		

Atributos para cajas

margin-left	Unidades de medida CSS	margin-left: 2em;
Define el tamaño del margen izquierdo.		
margin-right	Unidades de medida CSS	margin-right: 2em;
Define el tamaño del margen derecho.		
margin-top	Unidades de medida CSS	margin-top: 2em;
Define el tamaño del margen superior.		
margin-bottom	Unidades de medida CSS	margin-bottom: 2em;
Define el tamaño del margen inferior.		
margin	Unidades de medida CSS	margin: 20px 10px;
Agrupa todas las propiedades de margen en una sola. El orden para especificarlos es arriba-derecha-abajo-izquierda.		
padding-left	Unidades de medida CSS	padding-left: 3%;
Define el tamaño del margen interno izquierdo.		
padding-right	Unidades de medida CSS	padding-right: 3%;
Define el tamaño del margen interno derecho.		
padding-top	Unidades de medida CSS	padding-top: 3%;
Define el tamaño del margen interno superior.		
padding-bottom	Unidades de medida CSS	padding-bottom: 3%;
Define el tamaño del margen interno inferior.		
padding	Unidades de medida CSS	padding: 10px 20px 10px 30px;
Agrupa todas las propiedades de margen interno en una sola. El orden para especificarlos es arriba-derecha-abajo-izquierda.		
border-left	Unidades de medida CSS [none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset] Color	border-left: 1px solid black;
Define las propiedades del borde izquierdo.		
border-right	Unidades de medida CSS [none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset] Color	border-right: 1px solid black;
Define las propiedades del borde derecho.		

border-top	Unidades de medida CSS [none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset] Color	border-top: 1px solid black;
Define las propiedades del borde superior.		
border-bottom	Unidades de medida CSS [none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset] Color	border-bottom: 1px solid black;
Define las propiedades del borde inferior.		
border	Unidades de medida CSS [none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset] Color	border: 1px solid black;
Agrupa todas las propiedades de borde en una sola. El orden para especificarlos es arriba-derecha-abajo-izquierda. Afecta a los 4 por igual.		
border-width	Unidades de medida CSS	border-width: 3px;
Define el ancho del borde.		

Atributos para formato visual

float	none | left | right	float:none;
Hace que un elemento se comporte como flotante, alineándose a izquierda o derecha.		
clear	none | left | right | both	clear: both;
Posiciona el elemento donde no haya ningún otro flotante alineado al lado que se especifique.		
width	Unidades de medida CSS	width: 300px;
Especifica el ancho de un elemento.		
height	Unidades de medida CSS	height: 4vw;
Especifica la altura de un elemento.		
position	static | relative | absolute | fixed	position: fixed;
Establece el tipo de posicionamiento del elemento.		
top	Unidades de medida CSS	top: 40px;
Especifica la posición del elemento con respecto a la parte superior.		
bottom	Unidades de medida CSS	bottom: 40px;
Especifica la posición del elemento con respecto a la parte de abajo.		
left	Unidades de medida CSS	left: 40px;
Especifica la posición del elemento con respecto a la izquierda.		

right	Unidades de medida CSS	right: 40px;
Especifica la posición del elemento con respecto a la parte derecha.		
display	inline | block | list-item | run-in | inline-block | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none	display: block;
Especifica el comportamiento de un elemento. Inline se comporta como un elemento en línea y block como un elemento en bloque. None hace que el elemento no se muestre.		
z-index	Auto | número	z-index: 100;
Controla el nivel de la capa. Cuanto mayor sea, más cerca estará.		

EJEMPLO PRÁCTICO

Anabel pertenece al equipo de desarrollo web del departamento de informática de su empresa, que están actualmente en pleno desarrollo del nuevo sitio web y se encarga de cumplir con las especificaciones requeridas por el equipo de diseño gráfico del sitio.

Le han solicitado la inclusión de unos bloques en la web, pero que permanezcan de manera estática en la misma y que el código se le pase al resto de desarrolladores para que lo tengan de referencia en otras páginas web del sitio corporativo.

¿Cómo sería el código fuente para la creación de dos bloques estáticos?

Solución.

El código para el posicionamiento estático de bloques sería:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>ejemplo</title>
    <meta charset="utf-8">
    <style>
        div {width:100px;height:100px;}
    </style>
<body>
    <div style="background-color:blue;">
    </div>
    <div style="background-color:green;">
    </div>
</body>
</html>
```

6. VALIDACIÓN DE DOCUMENTOS HTML Y CSS

Una vez que se has terminado la elaboración del documento que recoge el código fuente de la página web, debes llevar a cabo la validación del mismo para comprobar que es correcto y puede implementarse de manera que pueda verse en un navegador web, con todos sus elementos, etiquetas, atributos y CSS que hayas utilizado.

En el código HTML de una página web podemos encontrar diferentes tipos de errores relacionados con:

1. Sintaxis
   Error en la escritura del código, por lo que el navegador no podrá mostrar la página correctamente.
2. Accesibilidad
   A pesar de poder estar correctamente escrito el código, no se presenta de forma adecuada en todos los dispositivos, o personas con deficiencias visuales, por ejemplo, no pueden acceder al contenido.
3. SEO
   Cuando los buscadores web no interpretan bien el código y afecta al posicionamiento web de la página.
4. Rendimiento
   Cuando afecta a la velocidad de carga o realización de alguna tarea, ralentizando las acciones.
5. Usabilidad
   Dificulta un uso correcto de la web por una forma incorrecta en la distribución de sus elementos.

ENLACE DE INTERÉS

Centrándonos en la importancia de un código fuente correcto de una página web, dispones de un servicio de validación HTML o CSS que ofrece W3C de manera gratuita y a través de la url.

Servicio de validación de marcado W3C.
Fuente: https://validator.w3.org/#validate_by_uri

Ofrece tres formas distintas de validar la página, pinchando sobre la correspondiente pestaña podremos validarla por:

- URI, debiendo insertar la url de la página a validar, con el requisito de que debe estar publicada en internet.
- Carga de archivos (File upload), donde se subirá el archivo .html a analizar.
- Entrada directa (Direct input), insertando directamente el código HTML que queremos analizar en la casilla correspondiente.

La salida de información después del análisis nos ofrece datos que podemos configurar como:

- El código fuente numerado por líneas para facilitar la corrección de errores.
- Un esquema o árbol de la estructura de la página.
- Informe de las imágenes de la página web con resumen de las características de cada una de ellas asociadas a una imagen de pequeño tamaño.

En el caso de la validación CSS es posible llevar a cabo una validación más específica mediante:

- Ajustando el nivel de detalle entre:
  - Errores, cuando las CSS que se comprueban no respetan las recomendaciones sobre CSS de la W3C.

- Advertencias (warnings), no están relacionadas con las especificaciones, pero si alertan sobre los posibles puntos que pueden ser susceptibles de tener un comportamiento no controlado.

- Comprobando los perfiles, que contienen las características necesarias para una correcta implementación en la plataforma elegida.
- Verificando la aplicación de la regla @media en todo el documento sobre los tipos de medios para las CSS.

VÍDEO DE INTERÉS

En este vídeo se explica la validación HTML y CSS. Véase a partir del minuto 1.37

7. LENGUAJE DE MARCAS PARA LA SINDICACIÓN DE CONTENIDOS

Entre las funcionalidades que encontramos dentro de los lenguajes de marcas tenemos la sindicación de contenidos, tendrás que llevar a cabo una sindicación de contenidos de la página que estás desarrollando con las sentencias o instrucciones necesarias dentro del documento web y comprobar qué resultados se ofrecen a los distintos usuarios.

Al hablar de sindicación de contenidos con la utilización de lenguaje de marcas, debemos comenzar por hablar de las RSS (Really Simple Sindication) como tecnología que se utiliza para la distribución de información de una web a los usuarios.

Habitualmente se requiere disponer de un software específico o lector de contenidos RSS, donde la información puede ser compartida mediante la suscripción a un servicio de agregador de noticias, o bien, compartiendo la información existente en otros sitios web, haciendo la labor de emisor.

Entre la gran variedad de lectores de RSS (agregadores) distinguimos los siguientes tipos:

- De escritorio, instalables en el propio equipo del usuario.
- En línea, no requiere instalación, sólo el alta en el sitio web para su utilización.
- Plugins, extensiones de algunos navegadores como Opera o FireFox que permiten la suscripción a RSS.

Esta técnica de distribuir contenidos web está derivada del lenguaje XML, permitiendo la construcción de archivos que almacenan el contenido a compartir, bien sea propio o mostrado en otras páginas web, dando lugar a la redifusión de contenidos web.

Ventajas

- Actualización de contenidos sin necesidad de intervención del usuario.
- Poco peso y facilidad de transmisión al tener un formato de texto plano.
- Posibilidad de filtrado de información sin necesidad de agrupación.

Pasos a seguir en la creación de RSS

1. Creación de un canal de contenido

Va a requerir únicamente un editor XML y una página web, y algún tipo de herramienta como pueden ser Drupal o Joomla, que permiten la creación de una estructura y soporte para su creación y administración, con las ventajas de ser un software de código abierto, implementados en php con servidores apache y gestión de bases de datos a través de Mysql.

2. Validación de archivo RSS

W3C pone a nuestra disposición la herramienta de validación desde la url https://validator.w3.org/feed/

3. Publicación del archivo RSS

El siguiente paso es la subida del archivo a un servidor web, siguiendo unas sencillas pautas que consistirán en:

- Insertar en la página de inicio el enlace al archivo RSS.
- Habilitar el link que los lectores RSS seguirán para su lectura, suscribiéndose al feed.
- Envío de la dirección URI del archivo RSS al sitio web, donde se catalogan y almacenan los feeds para los distintos navegadores web.

Elementos principales de un RSS

1. <channel>

Describe la fuente RSS e integra tres elementos obligatorios, <tittle> nombre del canal, <link> especifica el enlace al canal y <description> describe el contenido del canal.

Puede contener otra serie de elementos opcionales con los que definir las categorías de para el canal <category> o la dirección de la documentación del formato utilizado <docs>, entre otros.

2. <item>

Son elementos que cuentan con la posibilidad de definir un determinado artículo RSS, pueden existir varios dentro del <cannel>, y contarán con elementos fundamentales como el elemento <channel>: Tittle, link y description. Además de opcionales como <author>, <category> o <comments>, por ejemplo.

RECUERDA

Una vez seleccionado el agregador de RSS, es necesario añadir qué feeds o archivos RSS son necesarios para lograr la sindicación de contenidos.

RESUMEN FINAL

Centrándonos en la utilización de HTML, es imprescindible conocer en qué consiste y una breve historia de su evolución en el tiempo, para entrar de lleno en lo que es su estructura y partes como <head> y <body>, así como la utilización de etiquetas y atributos, haciendo una parada en las herramientas de diseño web que podemos encontrar, desde los sencillos editores de texto, a un paso más con aplicaciones como Notepad ++ o más completas como WordPress o Wix.

Para dar formato a una página web mediante hojas de estilo, se ha estudiado cómo definir reglas CSS y su forma de aplicación. Así mismo, se ha visto cómo definir clases e identificadores y usarlos para asignar estilos diferentes a los mismos elementos HTML.

Se ha explicado el modelo de caja y cómo realizar la maquetación de una página web en base a éste. También se han visto las distintas unidades de medida que se pueden usar con CSS y su efecto en el diseño adaptativo, y se ha presentado un listado a modo de resumen con las propiedades CSS más comunes y sus posibles valores.

Como finalización del proceso de desarrollo de una página web, hemos destacado la importancia de la validación de documentos HTML y CSS, haciendo referencia al validador que pone a nuestra disposición W3C desde su propia web.

Para finalizar, hemos tratado el tema de la sindicación de contenidos web a través de las RSS, qué son, cómo se elaboran y se hacen visibles en una web.