

Apuntes HTML y CSS

Profesor: Matías Corti

Requisitos	1
Navegador Web	1
Editores de Código / IDEs	1
Introducción a HTML	2
¿Qué es HTML?	2
Breve Historia	2
¿Cómo funciona?	3
Elementos HTML	3
Elemento tradicional	4
Elemento auto-contenido	4
Etiquetas básicas	6
Primer Página	7
Direccionamiento	7
URL	7
Internas y Externas	8
Relativas y absolutas	9
Peticiónes HTTP	10
Convenciones	10
Ejercicio 1: CV	11
Tipos de Elementos HTML	12
Otras Etiquetas	12
Ejercicio 2: Libro	14
Tablas	14
Formularios	15
Referencias	17
Introducción a CSS	17
¿Qué es CSS?	17
¿Cómo funciona?	17
Reglas CSS	17
Cómo incluir CSS en un documento HTML	18
Estilos En Línea (Desaconsejado)	18
Estilos Internos	19
Estilos Externos (Recomendada)	19
Selectores CSS	20
Selector universal	21
Selector de Tipo o Etiqueta	21
Selector de Clase	22
Selector de ID	22
Selector Descendente	23
Combinación de Selectores	23

Herencia	24
Colisiones de estilos	24
Elementos Comunes	24
Colores	24
Unidades de Medidas	25
Relativas	25
Absolutas	25
Propiedades básicas	25
Ejercicio 3: Tabla de Colores	26
Ejercicio 4: CSS a CV y Libro	27
Referencias	27
HTML y CSS	27
Modelo de cajas	27
Anchura	28
Altura	29
Margen	29
Relleno	29
Posicionamiento	30
Visualización	33
Relación entre display, float y position	33
Estructura o Layout	34
Diseño a 2 columnas con cabecera y pie de página	34
Diseño a 3 columnas con cabecera y pie de página	35
Ejercicio 5: Librería	36
HTML5	37
¿Qué es HTML5?	37
Etiquetas básicas	38
Comparación de Layout entre HTML4 y HTML5	39
Otros Agregados	41
Mejoras en los formularios	41
Ejercicio 6: HTML5	42
JavaScript	43
Ejercicios	43

Requisitos

Navegador Web

Para poder visualizar los ejemplos y ejercicios que escribiremos, es necesario un Navegador Web, si bien es posible utilizar cualquiera, recomiendo el uso de [Google Chrome](#) o [Firefox](#), ya que nos brinda además herramientas extra para trabajar y unificar criterios.

Editores de Código / IDEs

Si bien el código puede escribirse en cualquier editor de texto plano, se recomienda el uso de editores de texto que reconocen el código de los lenguajes utilizados, que además del aspecto visual, nos brindan funciones y herramientas específicas para facilitar el armado de código de calidad.

- VSCode <https://code.visualstudio.com/>
- Sublime Text <https://www.sublimetext.com/3>
- Notepad++ <https://notepad-plus-plus.org/download/>
- Muchos otros...

Introducción a HTML

¿Qué es HTML?

HTML es el lenguaje utilizado para crear y representar la mayoría de las páginas web de Internet.

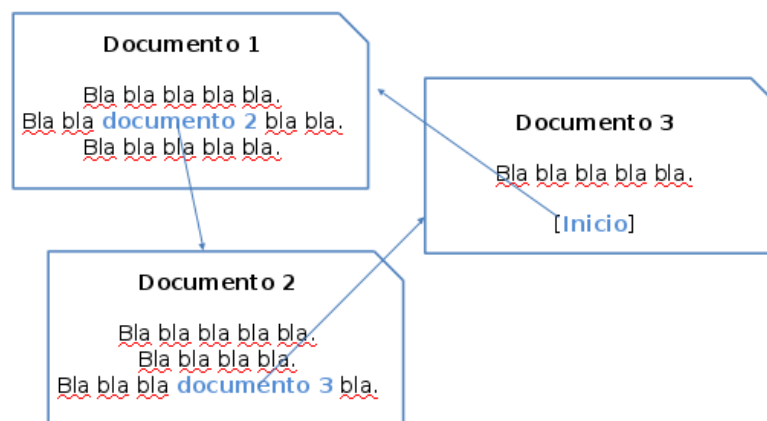
Es muy fácil de aprender y escribir por parte de las personas dado que son “documentos de texto plano” que contienen *Elementos* conformados por *atributos* y *contenido*.

El lenguaje HTML es un estándar reconocido universalmente y que permite publicar información de forma global, esto significa que una misma página HTML se visualiza de forma muy similar en cualquier navegador de cualquier sistema operativo.

HTML son las siglas de HyperText Markup Language, traducido al español como Lenguaje de Marcado de Hipertexto.

Breve Historia

En 1989 Tim Berners-Lee creó un sistema que permitía conectar documentos electrónicos, utilizando un lenguaje que enlazaba un documento con otro, dando algunas características de formato a estos textos.



¿Cómo funciona?



El *Cliente* (por ejemplo un Navegador web) solicita un documento HTML (archivo) al *servidor*.

El *servidor*, en caso de ser posible, se lo envía.

El *Cliente*, al recibirlo, lo *parsea* (interpreta) de arriba hacia abajo y de izquierda a derecha, *renderizando* (mostrando) en pantalla los Elementos que contiene el documento.

Elementos HTML

Son los fragmentos que componen una página web; cada uno puede contener texto, imágenes, otros elementos o incluso nada. Nos permiten estructurar el contenido a mostrar dentro del documento HTML.

Los elementos, según su sintaxis (cómo se escriben), se pueden clasificar en dos clases:

Elemento tradicional

- **Etiqueta (o tag) de apertura**, que puede contar con algunos *atributos*.
Está formada por una "palabra clave" (que indica el nombre o tipo de elemento), encerrada entre los símbolos "<" y ">", puede contener **atributos** antes del cierre.
- **Contenido adjunto**.
Puede ser el texto u otros Elementos HTML que están contenidos dentro del Elemento. Es la parte que se muestra al usuario final.
- **Etiqueta de cierre**.
Una *etiqueta de cierre*, está formada por la misma palabra clave que la etiqueta de apertura, pero anteponiendo una "/" entre ella y el símbolo "<".



Elemento auto-contenido

Constan sólo de un único tag, esto se debe a que no cuentan con Contenido adjunto explícito. Su formato similar a la etiqueta de apertura de los elementos típicos, a diferencia que se pueden escribir con una "/" al final.

De este último grupo son elementos como los saltos de línea (`
`), imagen (``), línea horizontal (`<hr/>`), entre otros.



Notas:

- Es muy común que se hable de forma indistinta de "*elementos*" y "*etiquetas*" (o "*tags*"), aunque como vimos aquí los elementos son mucho más que etiquetas.
- El lenguaje HTML es muy permisivo en su sintaxis, esto puede parecer un aspecto positivo, pero el resultado final son páginas con un código HTML desordenado, difícil de mantener y muy poco profesional. Por lo tanto se deben respetar algunas reglas o [Convenciones](#) básicas que nos permitan generar código de calidad.

Etiquetas básicas

`<html>...</html>`

Define el inicio y fin del documento HTML, le indica al navegador que su contenido debe ser interpretado como código HTML.

`<head>...</head>`

Define la cabecera del documento HTML. Suele contener etiquetas con recursos e información sobre el documento que no se muestra directamente al usuario como. Por ejemplo, el título de la ventana del navegador, enlaces a recursos y meta-información.

`<title>...</title>`

Define el título de la página. Se define dentro del `<head>`. Por lo general, el título aparece en la barra de título encima de la ventana. Siempre debe estar presente.

`<meta />`

Se utiliza para agregar meta-información a la página, con distintos fines. Por ejemplo, la codificación (charset) de los caracteres del documento, información que utilizan los Búscadores para conocer detalles de la página y ordenarlas mejor en las búsquedas, entre otras.

Ejemplos:

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

`<meta name="author" content="Nombre Autor" />`

`<meta name="description" content="Descripción de la página" />`

`<meta name="keywords" content="etiqueta1, etiqueta2, etiqueta3" />`

`<body>...</body>`

Define el contenido principal o cuerpo del documento. Esta es la parte del documento HTML que se muestra en el navegador al usuario.

`<h1>...</h1>`

Muestra el título de la página dentro del `<body>`. Sólo se debe utilizar un único `<h1>` en cada documento.

`...`

Permite generar enlaces a otras páginas, para ello se establece el destino en el atributo `href`, esa dirección puede ser interna (otro documento del mismo sitio) o bien externo (a una dirección de internet).

`
` | `
`

Son dos alternativas del mismo elemento, no se deben usar juntas.

Elemento auto-contenido que representa un salto de línea, es decir, el equivalente a un "Enter" en un procesador de textos cualquiera.

Primer Página

Creamos una carpeta. Luego abrimos un Editor de Textos y creamos un archivo nuevo, que guardamos dentro de esa carpeta, con el nombre "index.html". En el archivo escribimos el siguiente código:

```
<html>
  <head>
    <title>Hola Mundo!</title>
  </head>
  <body>
    <h1>Primer página</h1>
    Hola Mundo!<br/>
    <a href="index.html">Vínculo Interno</a><br/>
    <a href="http://www.google.com">Vínculo a Externo</a>
  </body>
</html>
```

Abrimos el archivo en un navegador web.

Direccionamiento

URL

El acrónimo URL (Uniform Resource Locator) hace referencia al identificador único de cada recurso disponible en Internet. Las URL son esenciales para crear los enlaces, pero también se utilizan en otros elementos HTML como las imágenes, los formularios y otros recursos.

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

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

<http://www.google.com>

El segundo objetivo de las URL es el de permitir la localización eficiente de cada recurso de Internet. Para ello es necesario comprender las diferentes partes que pueden formar las URL.

Cualquier URL sigue el siguiente formato:

[scheme://[user:password@]host[:port]][/]path[?query][#fragment]

Aclaración: En informática se suelen utilizar los corchetes [] para indicar que lo que se encuentra contenido entre ellos es

opcional.

Ejemplo: <http://www.frlp.utn.edu.ar/seccion/novedades.html?page=3&limit=20#comentarios>

Las cinco partes que forman la URL anterior son:

- **Protocolo** (<http://>) *scheme*
- **Servidor** (www.frlp.utn.edu.ar) *host*
- **Puerto** (:80 es el default para web, por eso no suele escribirse) *:port*
- **Ruta** (/seccion/novedades.html) *path*
- **Consulta** (?page=3&limit=20) *?query*
parámetros o información adicional necesaria para que el servidor localice correctamente el recurso que se quiere acceder. Siempre comienza con el carácter ? y contiene una sucesión de palabras separadas por = y &.
- **Sección** (#comentarios) *#fragment*
permite que el navegador se posicione automáticamente en una sección de la página web. Siempre comienza con el carácter #

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

Internas y Externas

Cuando se accede a algunos enlaces, el navegador abandona el sitio web para acceder a páginas que se encuentran en otros sitios (hosts). Estos enlaces se conocen como "**enlaces externos**". Sin embargo, la mayoría de enlaces de un sitio web apuntan a páginas del propio sitio web, por lo que se denominan "**enlaces internos**".

Relativas y absolutas

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

Las **URL relativas prescinden de algunas partes de las URL para hacerlas más breves**. Como se trata de URL incompletas, la información faltante se toma del contexto en el que es utilizada.

Por ejemplo una página publicada en la URL Absoluta:

<http://www.ejemplo.com/nivel1/nivel2/pagina1.html>

Quiere incluir en ella un enlace a otra página que se encuentra en la dirección Absoluta:

<http://www.ejemplo.com/nivel1/nivel2/pagina2.html>

Podría simplificar el enlace utilizando la URL Relativa: [/nivel1/nivel2/pagina2.html](#)

La última URL no es absoluta, le falta el protocolo y el servidor, por lo que el navegador supone que son los mismos que los de la página origen (<http://www.ejemplo.com>).

Esto simplifica visualmente el código para los enlaces internos (predominantes en los sitios), pero además hace que los enlaces del sitio sea más fácil de mantener y más independiente del host al que se sube. Pensemos por ejemplo si hubiese que utilizarlo en otro host habría que editar todos los enlaces absolutos de cada documento.

A continuación se describen los diferentes tipos de URL Relativa posibles, para ejemplificar tomaremos el caso de que todas las URL Relativas descritas están definidas en la página cuya URL

Absoluta es `http://www.ejemplo.com/nivel1/nivel2/pagina1.html`

- **"documento.html"**

Para convertir en Absoluta una URL Relativa como ésta, sólo reemplaza el nombre del documento actual por la cadena definida.

Resultado: `http://www.ejemplo.com/nivel1/nivel2/documento.html`

Variante: escribir también un directorio (sin comenzar con "/"), por Ejemplo: **otro_nivel/documento.html**

Resultado: `http://www.ejemplo.com/nivel1/nivel2/otro_nivel/documento.html`

- **"../documento.html"**

Al comenzar con "../" comprende que antes de reemplazar el nombre del documento, debe subir un nivel a partir de la ruta definida, sólo reemplaza el nombre del documento actual por la cadena definida.

Resultado: `http://www.ejemplo.com/nivel1/documento.html`

Variante 1: escribir también un directorio, por Ejemplo: **../otro_nivel/documento.html**

Resultado: `http://www.ejemplo.com/nivel1/otro_nivel/documento.html`

Variante 2: escribir más de un "../", por Ejemplo: **../../documento.html**

Resultado: `http://www.ejemplo.com/documento.html`

Variante 3: combinar ambas variantes: **../../otro_nivel/documento.html**

Resultado: `http://www.ejemplo.com/otro_nivel/documento.html`

- **"/documento.html"**

Al comenzar con "/" el navegador considera que es la ruta completa comenzando desde la raíz del servidor.

Resultado: `http://www.ejemplo.com/documento.html`

Variante: escribir también un directorio, por Ejemplo: **/otro_nivel/documento.html**

Resultado: `http://www.ejemplo.com/otro_nivel/documento.html`

Peticiones HTTP

Cuando nuestro navegador quiere acceder a una página web a través de HTTP (scheme = http), lo que hace es comunicarse con un servidor HTTP. Para ello descompone la URL en diferentes partes que le permite conocer la dirección de la máquina (host) y la ruta (path) del recurso que al que quiere acceder (o al que le va a enviar información), y envía un mensaje al servidor, lo que formalmente se conoce como una petición.

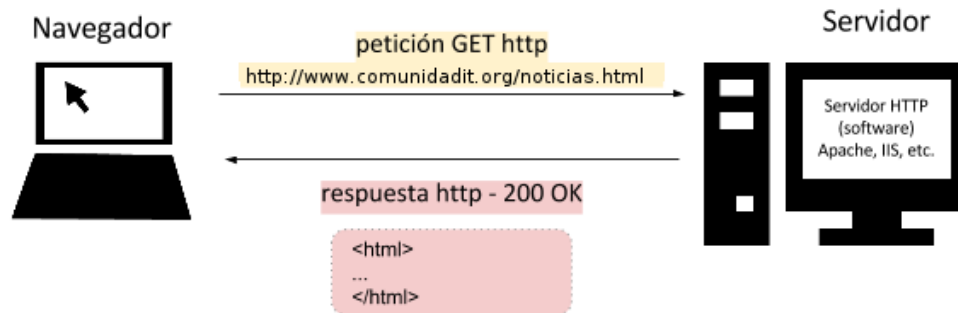
Algunas de las acciones que provocan que un navegador realice una petición HTTP son: escribir una URL en la barra de direcciones, pulsar un enlace, refrescar una pestaña o enviar un formulario.

Existen varios tipos de peticiones, aunque nosotros en este curso trabajaremos con dos tipos:

- **GET:** para solicitar información.
- **POST:** para enviar información.

Los mensajes de respuesta del servidor pueden ser de muchos tipos, aunque nosotros nos encontraremos normalmente tres, que significan:

- **200:** que se ha encontrado correctamente el fichero/recurso.
- **403:** que no tenemos permiso para acceder al fichero/recurso.
- **404:** que el fichero/recurso que le hemos solicitado no se ha podido encontrar en el disco duro (puede ser porque no esté o porque la ruta es incorrecta).



Convenciones

A continuación se describen algunas de las principales convenciones o buenas prácticas que deberemos de tener en cuenta a la hora de escribir código HTML.

En el código

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

```
<!-- MAL -->
<p>Este párrafo tiene una parte <b>destacada</p></b>
<!-- BIEN -->
<p>Este párrafo tiene una parte <b>destacada</b></p>
```

- Los nombres de los elementos HTML y sus atributos se deben escribir en minúsculas

```
<!-- MAL -->
<IMG SRC="html5.gif" ALT="Logo HTML5">
<!-- BIEN -->

```

- Los valores de los atributos en HTML deben ir entre comillas dobles, nunca con comillas simples o sin comillas:

```
<!-- MAL -->
<img src='html5.gif' alt='Logo HTML5'>
<img src=html5.gif alt=Logo HTML5>
<!-- BIEN -->

```

- La indentación se debe hacerse siempre de la misma manera, ya sea con 4 espacios o un "Tab" por cada nivel. Elegir una modalidad y respetarla dentro de todos los archivos del proyecto. (prácticamente todos los editores de código permite configurar este valor).

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

- No introducir espacios antes o después del signo "igual":

```
<!-- MAL -->
<img src = "html5.gif" alt = "Logo HTML5">
```

```
<!-- BIEN -->

```

- Especificar el !DOCTYPE y el atributo lang en el elemento html:

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

- Usar UTF-8 como encoding.

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

- Evitar el uso de estilos en línea (atributo style).

- Evitar el uso de entidades HTML siempre que sea posible, salvo por ejemplo para: < (<) o & (&).

```
<!-- MAL -->
<h1>Página sobre &lt; HTML5 &amp; CSS3</h1>
<!-- BIEN -->
<h1>Página sobre &lt; HTML5 &amp; CSS3</h1>
```

Nombres de Archivos

- Establecer los nombres de los ficheros en minúsculas.
- En caso de usar mayúsculas, respetar el nombre exacto al hacer la referencia en el código. Windows no hace distinción entre mayúsculas y minúsculas pero otros sistemas sí, y esto puede provocar que una ruta funcione en un sistema operativo pero no en otro. Por ejemplo, si tenemos un fichero llamado Logo_HTML5.jpg y una página que hace referencia a él con: Funcionará en Windows pero en un sistema basado en Unix (Linux o Mac) no funcionará.
- Nunca usar acentos ni caracteres especiales: ñ, coma, punto, punto y coma, diéresis, etc.
- En lugar de espacio usa un guión: "-" ó "_".
- Darle nombres que representen al contenido, no sólo por usabilidad sino por posicionamiento web en buscadores (SEO).
- Extensiones de ficheros, es recomendable que cada tipo de fichero tenga una extensión:
 - HTML: ".html"
 - JPEG: ".jpg"
 - GIF: ".gif"
 - PNG: ".png"

Ejercicio 1: CV

Realizar un CV escribiendo un sitio web en HTML que conste de estas cuatro páginas correctamente vinculadas:

- Index (página de inicio), donde aparezca la información personal.
- Estudios.
- Experiencia Laboral.
- Hobbies.

Desde el index, se debe poder entrar a las otras 3 y desde cada una de las 3 se debe poder volver a index.

Utilizar todos los elementos vistos previamente y **respetar las Convenciones** mencionadas.

Tipos de Elementos HTML

El lenguaje HTML clasifica a los elementos en dos grupos según el comportamiento que presentan: elementos **en línea** (**inline** elements en inglés) y elementos **de bloque** (**block** elements en inglés).

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

Otras Etiquetas

`<p>...</p>`

Representa un párrafo dentro del documento.

`<h2>...</h2>` | `<h3>...</h3>` | `<h4>...</h4>` | `<h5>...</h5>` | `<h6>...</h6>`

Son más niveles de título, similares a `<h1>` con la diferencia que estos sí se pueden repetir dentro del documento en caso de ser necesario.

`...` | `...`

juntas.

Alternativas del mismo elemento, no usar

Negrita, sirve para enfatizar en el contenido, denotando algo que es muy importante.

`<i>...</i>` | `...`

Alternativas del mismo elemento, no usar juntas.

Cursiva o *itálica*, sirve para enfatizar en el contenido.

`<hr/>` | `<hr>`

Alternativas del mismo elemento, no usar juntas.

Elemento auto-contenido que renderiza una línea de división horizontal.

`...`

Lista no ordenada. Es un elemento compuesto que se utiliza para agrupar ítems ``. A cada

uno lo representará como una lista con viñetas, uno debajo del otro. Como Contenido directo sólo admite elementos de este tipo, uno o varios.

`...`

Lista ordenada. Es un elemento compuesto que se utiliza para agrupar items ``. A cada uno lo representará como una lista numerada, uno debajo del otro. Como Contenido directo sólo admite elementos de este tipo, uno o varios.

`...`

Item de una lista. Se utiliza tanto para listas ordenadas como no ordenadas.

``

Elemento auto-contenido utilizado para mostrar imágenes en el documento HTML. El atributo `src` define la dirección de la imagen, que puede ser interna (otro archivo del mismo sitio) o bien externo (a una dirección de internet). El texto del atributo `alt` es utilizado en su lugar cuando no se puede cargar la imagen.

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

Es un elemento de bloque que sirve para agrupar elementos HTML, permite organizar y dividir el documento en áreas o secciones.

`...`

Es un contenedor en línea. Sirve para aplicar estilo al texto o agrupar elementos en línea.

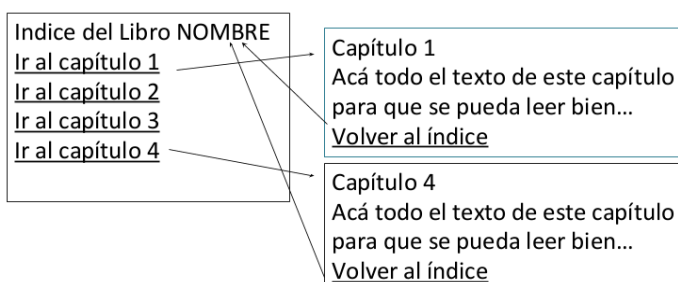
`<!-- Comentario -->`

Comentarios y anotaciones dentro del código. Estos elementos no serán visibles por el usuario final ya que no se renderizan en el navegador. Se utilizan por ejemplo para añadir notas de tareas pendientes, aclaraciones que nos ayuden a nosotros y a otras personas a entender el código.

Ejercicio 2: Libro

Escribir un sitio web en HTML que sea un Libro. Debe contener:

- Index (página de inicio), que tenga el título, una breve descripción y un índice de los capítulos correctamente vinculados.
- Capítulos: crear al menos cuatro capítulos en los que aparezca el título del capítulo y su texto correctamente dispuesto en párrafos. También los vínculos para ir al índice y los capítulos "anterior" y "siguiente" en caso que posean.



*Utilizar todos los elementos vistos previamente y **respetar las Convenciones** mencionadas.*

Tablas

El elemento compuesto `table` nos permite a representar información en filas y columnas.

Etiquetas:

`<table>...</table>` inician y finalizan la tabla respectivamente.

`<tr>...</tr>` ("table row"), inician y finalizan cada fila horizontal.

`<td>...</td>` ("table data"), delimitan una de las celdas que componen las filas de la tabla.

Atributos:

- `colspan` (valor: `número`, default: `1`), establece la cantidad de celdas contiguas que se agruparan horizontalmente. Aplicable a: `<td>`.
- `rowspan` (valor: `número`, default: `1`), establece la cantidad de celdas de una columna que se agruparán verticalmente. Aplicable a: `<td>`.
- `align` (valores: `left|right|center|justify`, default: `left`), indica la alineación horizontal que tendrán los elementos de la fila o columna. Aplicable a: `<tr>` y `<td>`.
- `valign` (valores: `top|middle|bottom`, default: `middle`), indica la alineación vertical que tendrán los elementos de la fila o columna. Aplicable a: `<tr>` y `<td>`.

Ejemplo 1:

```
<table>
  <tr>
    <td>Fila 1, columna 1</td>
    <td>Fila 1, columna 2</td>
  </tr>
  <tr>
    <td>Fila 2, columna 1</td>
    <td>Fila 2, columna 2</td>
  </tr>
</table>
```

Fila 1, columna 1	Fila 1, columna 2
Fila 2, columna 1	Fila 2, columna 2

Ejemplo 2:

```
<table>
  <tr>
    <td align="center">F1-C1</td>
    <td colspan="3" align="right">F1-[C2&C3&C4]</td>
  </tr>
  <tr valign="bottom">
    <td rowspan="2">F2-C1 & F3-C1</td>
    <td>F2-C2</td>
    <td>F2-C3</td>
    <td>F2-C4</td>
  </tr>
  <tr>
    <td>F3-C2</td>
    <td>F3-C3</td>
    <td>F3-C4</td>
  </tr>
</table>
```

F1-C1	F1-[C2&C3&C4]		
F2-C1 & F3-C1	F2-C2	F2-C3	F2-C4
	F3-C2	F3-C3	F3-C4

Formularios

Los formularios son elementos compuestos que permiten que los usuarios interactúen con las aplicaciones web.

```
<form action="tipo" method="post">...</form>
```

Será el elemento padre que anide todos los elementos HTML que representarán los campos de nuestro formulario, incluido el botón de “Enviar”.

action: indica la URL a la que se enviará la petición HTTP con toda la información del formulario.

method: (valores: `get`|`post`), indica el tipo de petición HTTP a realizar.

```
<label for="id_campo">Nombre</label>
```

Se usa para especificar la etiqueta (o nombre) del campo del formulario.

for: tiene que tener el mismo valor que el atributo `id` del campo (`input`, `select` o `textarea`) al que hace referencia la etiqueta.

```
<input type="tipo" id="id_campo" name="nombre_campo" value="valor">
```

Elemento auto-contenido que permite introducir diferentes tipos de campo de formulario en base al valor del atributo `type`. En función del tipo de campo, dispondremos de otros atributos específicos para cada uno.

type: (obligatorio, valores: `text`|`password`|`checkbox`|`radio`|`submit`|`reset`|`file`|`hidden`|y más).

id: es obligatorio si en el elemento `label` tiene un atributo `for`, en tal caso deberá contener un identificador único en la página.

name: representa el nombre asignado al campo cuando se envíe la petición HTTP.

value: representa el valor que se asignará al campo cuando se envíe la petición HTTP.

```
<select id="id_campo" name="nombre_campo">...</select>
```

Elemento compuesto que permite crear una lista desplegable de opciones, cada opción estará contenida como hija dentro de un elemento `<option>`.

id: idem que para elemento `input`.

name: idem que para elemento `input`.

```
<option value="valor_opcion">Opción</option>
```

Define cada una de las opciones posibles del `<select>` que lo contiene. Su contenido será lo que mostrará en pantalla.

value: idem que para elemento `input`.

```
<textarea id="id_campo" name="nombre_campo">Texto</textarea>
```

Permite introducir textos extendidos, con saltos de línea incluidos, normalmente se usa cuando hay que introducir: descripciones, biografías, etc.

id: idem que para elemento `input`.

name: idem que para elemento **input**.

`<button type="tipo">Texto</button>`

Representa un botón que presentará diferente comportamiento con el formulario según su tipo, definido el atributo **type**.

type: (obligatorio, valores: **submit**|**reset**|**button**) define el comportamiento del botón.

Ejemplo:

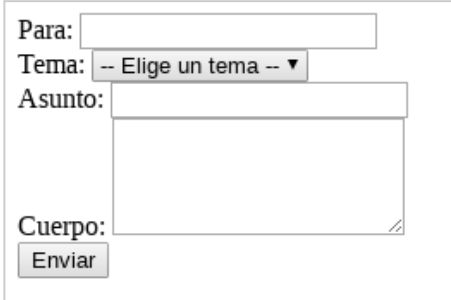
```
<form action="contacto.php" method="post">
  <label for="to">Para:</label>
  <input id="to" type="email">
  <br/>

  <label for="topic">Tema:</label>
  <select id="topic" name="topic">
    <option>-- Elige un tema --</option>
    <option value="proposal">Propuesta</option>
    <option value="report">Reporte</option>
    <option value="other">Otro</option>
  </select>
  <br/>

  <label for="subject">Asunto:</label>
  <input id="subject" name="subject" type="text">
  <br/>

  <label for="body">Cuerpo:</label>
  <textarea id="body" name="body"></textarea>
  <br/>

  <button type="submit">Enviar</button>
</form>
```

A visual representation of the HTML form code. It shows a form with the following elements: a text input field for 'Para:', a dropdown menu for 'Tema:' with the text '-- Elige un tema --', a text input field for 'Asunto:', a large text area for 'Cuerpo:', and a button labeled 'Enviar'.

Referencias

Existen muchos otros Elementos, Etiquetas y Atributos que los aquí mencionados, se puede encontrar mucha información al respecto en Internet.

A continuación dejo algunos enlaces que pueden ser útiles para buscar información o ejemplos:

- **Guía de Tags HTML por Categoría:**
https://www.w3schools.com/tags/ref_byfunc.asp
- **CheatSheets de Tags HTML5:**
https://html.com/wp-content/uploads/html5_cheat_sheet_tags.png

Introducción a CSS

¿Qué es CSS?

Es un lenguaje de marcado que nos permite controlar el aspecto de los documentos con HTML. CSS es la mejor forma de separar los contenidos y su presentación, es imprescindible para crear páginas web complejas.

HTML no fue originalmente pensado para contener los estilos de la interfaz, sino los contenidos de la misma. Para solucionar esto, la World Wide Web Consortium (W3C) creó lo que se llama CSS, que nos permite aplicar estilos al html de una manera más organizada.

CSS es el acrónimo de *Cascading Style Sheet*, traducido al español como Hojas de Estilo en Cascada.

En resumen:

HTML	->	Qué queremos mostrar
CSS	->	Cómo lo vamos a mostrar

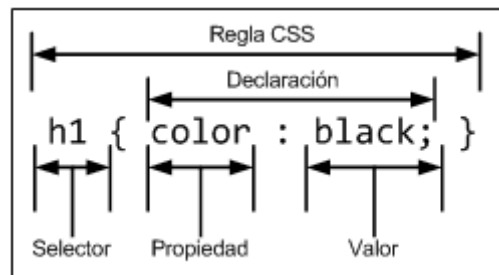
¿Cómo funciona?

A un documento HTML se inserta o vincula un conjunto de reglas CSS (hay varias formas de hacerlo).

El Cliente web (navegador), reconoce las reglas y las aplica a los elementos HTML que cumplen con los criterios de aplicación definidos, mostrando en pantalla el resultado de esa combinación.

Podemos aplicar una misma hoja de estilos CSS a todos los archivos HTML de un mismo sitio, y de esa manera manejar en un único lugar la presentación del sitio completo y el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

Reglas CSS



Regla CSS son los estilos que componen una Hoja de Estilos CSS. Cada regla está compuesta de:

- **Selector:** es un criterio que indica el elemento o elementos HTML a los que se aplica la regla CSS. Hay varios modos y combinaciones posibles.
- **Llave de apertura:** {
- **Declaración:** especifica los estilos que aplican a los elementos HTML. Puede estar compuesta por una o más definiciones CSS, las mismas contienen:

- **Propiedad:** característica que se modifica en el elemento seleccionado, como por ejemplo su tamaño de letra, su color de fondo, etc.
- **Valor:** establece el nuevo valor de la característica modificada en el elemento.
- **Cierre:** Cada definición termina con punto y coma ;
- **Llave de cierre:** }

De esta manera, las *llaves de apertura* y *de cierre* delimitan el conjunto de *Declaraciones* que se aplicarán a cada elemento HTML que coincida con el criterio definido como *Selector*.

Un archivo o bloque CSS puede contener un número ilimitado de reglas CSS, cada regla se puede aplicar a varios selectores diferentes y cada declaración puede incluir tantos pares propiedad/valor como se desee.

Cómo incluir CSS en un documento HTML

Existen tres maneras de incluir CSS dentro de un documento HTML.

Estilos En Línea (Desaconsejado)

Consiste en definir los estilos sobre los propios elementos HTML, utilizando el atributo `style`. En este caso el valor del atributo es directamente la Declaración de la Regla CSS (es decir que no posee ni *Selector*, ni *Llaves de apertura* y *cierre*).

Es el peor y el menos recomendado, ya que tiene problemas a la hora de mantener los estilos y entorpece la legibilidad del código HTML.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de estilos CSS en un elemento HTML</title>
  </head>
  <body>
    <p style="color: red; font-family: Verdana;">Un párrafo de texto.</p>
  </body>
</html>
```

Un párrafo de texto.

Esta forma se utiliza solamente en determinadas situaciones puntuales en las que se debe incluir un estilo muy específico para un solo elemento concreto.

Estilos Internos

Los estilos se definen en una zona específica del propio documento HTML, para ello se emplea el elemento `<style>` de HTML, dentro de `<head>` (solamente se pueden incluir en la cabecera del documento).

Ejemplo:

```

<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de estilos CSS en el propio documento</title>
    <style type="text/css">
      p {
        color: red;
        font-family: Verdana;
      }
    </style>
  </head>
  <body>
    <p>Un párrafo de texto.</p>
  </body>
</html>

```

Un párrafo de texto.

Estilos Externos (Recomendada)

En este caso, todos los estilos CSS se incluyen en un archivo de tipo CSS que las páginas HTML enlazan mediante la etiqueta `<link>`.

Un archivo de tipo CSS no es más que un archivo simple de texto cuya extensión es **.css**

Se pueden crear todos los archivos CSS que sean necesarios y cada página HTML puede enlazar tantos archivos CSS como necesite.

Si se quieren incluir los estilos del ejemplo anterior en un archivo CSS externo, se deben seguir los siguientes pasos:

1) Crea un archivo de texto y añadir solamente el siguiente contenido:

```

p {
  color: red;
  font-family: Verdana;
}

```

2) Se guarda el archivo de texto con el nombre **estilos.css**

Se debe poner especial atención a que el archivo tenga extensión **.css** y NO **.txt**

3) En la página HTML se enlaza el archivo CSS externo mediante el elemento `<link>`, dentro de `<head>`, reemplazando el elemento `<style>`:

```

<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de estilos CSS en un archivo externo</title>
    <link rel="stylesheet" type="text/css" href="estilos.css" />
  </head>
  <body>
    <p>Un párrafo de texto.</p>
  </body>
</html>

```

Un párrafo de texto.

Cuando el navegador carga la página HTML anterior, antes de mostrar sus contenidos también

descarga los archivos CSS externos enlazados mediante la etiqueta `<link>` y aplica los estilos a los contenidos de la página.

Normalmente, la etiqueta `<link>` incluye los siguientes atributos cuando enlaza un archivo CSS:

rel: indica el tipo de relación que existe entre el recurso enlazado (en este caso, el archivo CSS) y la página HTML. Para los archivos CSS, siempre se utiliza el valor `stylesheet`

type: indica el tipo de recurso enlazado. Sus valores están estandarizados y para los archivos CSS su valor siempre es `text/css`

href: indica la URL del archivo CSS que contiene los estilos. La URL indicada puede ser relativa o absoluta y puede apuntar a un recurso interno o externo al sitio web.

Esta es la forma más utilizada y recomendada para incluir CSS en las páginas HTML. La principal ventaja es que se puede incluir un mismo archivo CSS en muchas de páginas HTML, por lo que se garantiza la aplicación homogénea de los mismos estilos a todas las páginas que forman un sitio web.

Con este método, el mantenimiento del sitio web se simplifica al máximo, ya que un solo cambio en un solo archivo CSS permite variar de forma instantánea los estilos de todas las páginas HTML que enlazan ese archivo.

Selectores CSS

Como se vio en el anteriormente, una regla de CSS está formada por una parte llamada "selector" y otra parte llamada "declaración".

La declaración indica "qué hay que hacer" y el selector indica "a quién hay que hacérselo". Por lo tanto, los selectores son imprescindibles para aplicar de forma correcta los estilos CSS en una página.

A un mismo elemento HTML se le pueden aplicar varias reglas CSS y cada regla CSS puede aplicarse a un número ilimitado de elementos. En otras palabras, una misma regla puede aplicarse sobre varios selectores y un mismo selector se puede utilizar en varias reglas.

El estándar de CSS 2.1 incluye una docena de tipos diferentes de selectores, que permiten seleccionar de forma muy precisa elementos individuales o conjuntos de elementos dentro de una página web.

No obstante, la mayoría de páginas de los sitios web se pueden diseñar utilizando solamente los cinco selectores básicos.

Selector universal

Se utiliza para seleccionar todos los elementos de la página. El siguiente ejemplo elimina el margen y el relleno de todos los elementos HTML (por ahora no es importante fijarse en la parte de la declaración de la regla CSS):

```
* {  
    margin: 0;  
    padding: 0;  
}
```

El selector universal se indica mediante un asterisco (*). A pesar de su sencillez, no se utiliza

habitualmente, ya que es difícil que un mismo estilo se pueda aplicar a todos los elementos de una página.

Selector de Tipo o Etiqueta

Selecciona todos los elementos de la página cuya Etiqueta HTML coincide con el valor del *selector*. El siguiente ejemplo selecciona todos los párrafos de la página:

```
p {  
    ...  
}
```

Para utilizar este selector, solamente es necesario indicar el nombre de la etiqueta HTML correspondiente a los elementos que se quieren seleccionar, sin los caracteres < y >.

Ejemplo:

El siguiente ejemplo aplica diferentes colores de fuente a los titulares y a los párrafos de una página HTML:

```
/* Contenido del archivo estilos.css */  
h1 {  
    color: red;  
}  
h2, h3 {  
    color: orange;  
}  
h4, h5, h6 {  
    color: green;  
}  
p {  
    color: blue;  
}  
/* Fin del archivo estilos.css */  
  
<!DOCTYPE html>  
<html lang="es">  
  <head>  
    <meta charset="utf-8">  
    <title>Ejemplo de estilos CSS, Selector de Etiqueta</title>  
    <link rel="stylesheet" type="text/css" href="estilos.css" />  
  </head>  
  <body>  
    <h1>Título 1</h1>  
    <h2>Título 2</h2>  
    <h3>Título 3</h3>  
    <h4>Título 4</h4>  
    <h5>Título 5</h5>  
    <h6>Título 6</h6>  
    <p>Párrafo de texto.</p>  
  </body>  
</html>
```

Título 1

Título 2

Título 3

Título 4

Título 5

Título 6

Párrafo de texto.

Aclaración:

Si se quiere aplicar **las mismas declaraciones a más de un selector** diferente, se pueden **encadenar los selectores separados por una coma**. En el ejemplo es el caso de: h2, h3 (texto color naranja) y

h4, h5, h6 (texto color verde). Esto **es válido para todos los selectores, no sólo los de etiqueta**.

Selector de Clase

Selecciona un conjunto arbitrario de elementos HTML de la página, que poseen como valor del atributo `class` el **nombre de clase** utilizado por el selector luego del punto.

```
.clase {  
    ...  
    <etiqueta class="clase">...</etiqueta>  
}
```

Para distinguir este tipo de selector de los otros, en lenguaje CSS, el *Selector de Clase* se arma prefijando con un punto (.) la **clase** (valor del atributo `class` del HTML).

Ejemplo:

```
/* archivo estilos.css */  
.destacado { color: red; }  
  
<body>  
    <p class="destacado">Párrafo destacado.</p>  
    <p>Párrafo normal.</p>  
    <p>Otro párrafo normal.</p>  
    <p class="destacado">Otro párrafo destacado.</p>  
    <p>Párrafo normal, con <span class="destacado">parte destacada.</span></p>  
</body>
```

Párrafo destacado.

Párrafo normal.

Otro párrafo normal.

Otro párrafo destacado.

Párrafo normal, con parte destacada.

El selector `.destacado` se interpreta como "cualquier elemento de la página cuyo atributo `class` sea igual a `destacado`".

Selector de ID

Permite seleccionar un único elemento de la página a través del valor de su atributo `id` (dado que ese atributo no se debe repetir en dos elementos diferentes de una misma página).

La sintaxis de los selectores de ID es muy parecida a la de los selectores de clase, salvo que se utiliza el símbolo de la almohadilla (#) en vez del punto (.) como prefijo del nombre de la regla CSS.

```
#identificador {  
    ...  
    <etiqueta id="identificador">...</etiqueta>  
}
```

Para distinguir este tipo de selector de los otros, en lenguaje CSS, el *Selector de ID* se arma prefijando con un numeral (#) al **identificador** (valor del atributo `id` del HTML).

Ejemplo:

```
/* archivo estilos.css */  
#parrafo-2 { color: green; }  
  
<body>  
    <p>Primer párrafo.</p>  
    <p id="parrafo-2">Segundo párrafo.</p>  
    <p>Último párrafo.</p>  
</body>
```

Primer párrafo.

Segundo párrafo.

Último párrafo.

Selector Descendente

Permite seleccionar los elementos que se encuentran dentro de otros elementos. Un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento. No necesariamente tienen que ser hijos directos.

```
selector1 selector2 selector3 selectorN {  
    ...  
}
```

Se escriben los selectores separados por un espacio; es importante respetar el orden de anidamiento, el selector de la derecha es el más interno.

Ejemplo:

```
/* archivo estilos.css */  
p span { color: orange; }  
  
<body>  
  <p>  
    Texto Nro 1  
    <span>Texto Nro 2</span>  
    <em>Texto <span>Nro 3</span></em>...  
  </p>  
  <span>Texto Nro 4</span>  
</body>
```

Texto Nro 1 Texto Nro 2 Texto Nro 3...
Texto Nro 4

Aclaración:

NO confundir la *encadenamiento de selectores* con el *selector descendente*:

```
/* Aplica a todos los elementos "p", "a", "span" y "em" por separado */  
p, a, span, em { text-decoration: underline; }  
/* Aplica sólo a los elementos "em" que se estén dentro de "p", "a", "span" */  
p a span em { text-decoration: underline; }
```

Combinación de Selectores

Utilizar más de un selector unidos sin espacios intermedios, permite seleccionar elementos de manera más específica.

Ejemplo:

```
/* archivo estilos.css */  
span.destacado { color: red; }  
  
<body>  
  <p class="destacado">Párrafo destacado.</p>  
  <p>Párrafo normal.</p>  
  <p>Otro párrafo normal.</p>  
  <p class="destacado">Otro párrafo destacado.</p>  
  <p>Párrafo normal, con <span class="destacado">parte destacada.</span></p>  
</body>
```

Párrafo destacado.
Párrafo normal.
Otro párrafo normal.
Otro párrafo destacado.
Párrafo normal, con parte destacada.

El selector `span.destacado` se interpreta como "cualquier elemento de tipo `` que disponga de un atributo `class` con valor `destacado`".

Herencia

Una de las características principales de CSS es la herencia de los estilos definidos para los elementos. Cuando se establece el valor de una propiedad CSS en un elemento, sus elementos descendientes heredan de forma automática el valor de esa propiedad.

Colisiones de estilos

En las hojas de estilos complejas, es habitual que varias reglas CSS se apliquen a un mismo elemento HTML. Si asignamos estilos a un elemento de diferentes formas siempre predominará la regla que tenga un selector más específico. Cuanto más específico sea un selector, más importancia tiene su regla asociada.

A continuación están los tipos de selector, ordenados de menos a más específicos:

- Por nombre de etiqueta
- Por clase (class)
- Por identificador (id)

A igual especificidad, se aplica la última regla indicada.

Elementos Comunes

Colores

Existen múltiples formas de especificar el color, aquí veremos algunas:

Palabra Clave:

aqua | black | blue | fuchsia | gray
| green | lime | maroon | navy |
olive | orange | purple | red |
silver | teal | white | yellow

Las palabras clave indican de modo textual el color aplicado, son mucho más limitadas las opciones que los demás mecanismos.

Valor hexadecimal:

#faf o #ffaaff.

El formato hexadecimal está compuesto por un # y 3 o 6 dígitos del 0 al 9 y de la A la F.

Dónde #000 (ó #000000) es el color negro y #fff (ó #ffffff) es el blanco.

Valor RGB (Red, Green, Blue):

rgb(255, 160, 255) o rgb(100%, 62.5%, 100%).

Dónde rgb(0, 0, 0) es el color negro y por el contrario, rgb(255, 255, 255) es el blanco.

Valor RGBA (RGB + Alpha):

rgba(255, 160, 255, 1) or rgba(100%, 62.5%, 100%, 1)

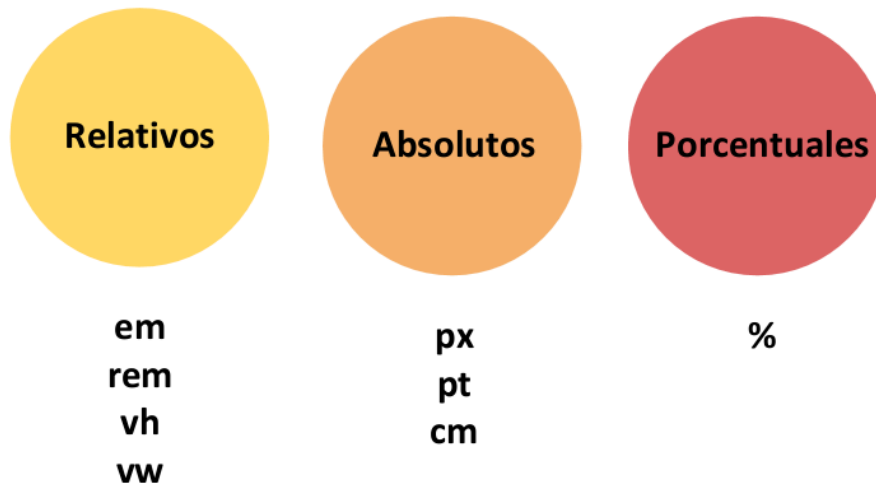
Es similar al anterior pero admite transparencia, mediante el valor Alpha (el último), que tiene

maroon #800000	red #ff0000	orange #ffa500	yellow #ffff00	olive #808000
purple #800080	fuchsia #ff00ff	white #ffffff	lime #00ff00	green #008000
navy #000080	blue #0000ff	aqua #00ffff	teal #008080	
black #000000	silver #c0c0c0	gray #808080		

que estar comprendido entre [0-1], siendo 1 = opaco y 0 = transparente.

Unidades de Medidas

Las medidas en CSS se emplean, entre otras, para definir la altura, anchura y márgenes de los elementos y para establecer el tamaño de letra del texto. Todas las medidas se indican como un valor numérico entero o decimal seguido de una unidad de medida (sin ningún espacio en blanco entre el número y la unidad de medida).



Relativas

Se llaman así porque son unidades relativas al dispositivo sobre el que se está viendo la página web, que dependiendo de cada usuario puede ser distinto, tales como computadoras, tablets o teléfonos móviles.

En principio las unidades relativas son más aconsejables, porque se ajustarán mejor al medio con el que el usuario está accediendo a nuestra web.

Absolutas

Las unidades absolutas son medidas fijas, que deberían verse igual en todos los dispositivos.

Pese a que en principio pueden parecer más útiles, puesto que se verían en todos los sistemas igual, tienen el problema de adaptarse menos a las distintas particularidades de los dispositivos que pueden acceder a una web y restan accesibilidad a nuestro web.

Se aconseja utilizar, por tanto, medidas relativas.

Propiedades básicas

color

Principalmente se usa para modificar el color del texto, se puede aplicar a cualquier elemento.

font

Permite manipular varias propiedades de la fuente a la vez.

font-size

Permite especificar el tamaño de la fuente (en px, em, rem).

font-style

Permite darle estilo a la fuente (ejemplo: normal o italic).

font-family

Establece una lista de fuentes tipográficas (arial, helvetica, sans-serif, etc).

font-weight

Especifica el ancho de la fuente (bold, 400, 600, ...).

text-align

Alinea el texto (left, right, center, justify).

text-decoration

Permite añadir un subrayado, tachar una palabra, etc. (ejemplos: underline, line-through)

text-transform

Permite transformar en mayúsculas, minúsculas, etc. (ejemplos: uppercase, lowercase)

line-height

Ajustar el interlineado usando (unidades: px, em, rem, etc).

background-color

Color del fondo del elemento.

background-image

Permite especificar una imagen de fondo.

background-repeat

Permite usar una imagen a modo de mosaico en diferentes modalidades.

box-shadow

Crear un efecto de sombra para un elemento.

border

Añade un borde a un elemento y establece algunas de propiedades (grosor, estilo de línea, etc.)

border-color

Color del borde.

border-style

Diferentes estilos para el borde (sólido, puntos, etc.)

border-radius

Permite crear esquinas redondeadas para un elemento.

Ejercicio 3: Tabla de Colores

Escribir un sitio web en HTML que contenga una tabla de 4 columnas por 4 filas. Aplicar las siguientes reglas:

- Borde a todos los elementos de la tabla.
- Un color de fondo para todos los <td>.

- Otro color de fondo para la celda de la intersección entre la segunda fila y la tercer columna.
- Otro color de fondo para toda la segunda fila.
- Otro color de fondo para los elementos de la tercer columna.
- Otro color de fondo en común para para la primer y la última celda.
- Otro color de fondo para la celda de la intersección entre la tercer fila y la segunda columna.

Ejercicio 4: CSS a CV y Libro

Para los HTML de los ejercicios 1 y 2, crear una hoja de estilos para cada uno y aplicar estilos.

Referencias

Existen muchos otros Selectores, Propiedades y Atributos que los aquí mencionados, se puede encontrar mucha información al respecto en Internet.

A continuación dejo algunos enlaces que pueden ser útiles para buscar información o ejemplos:

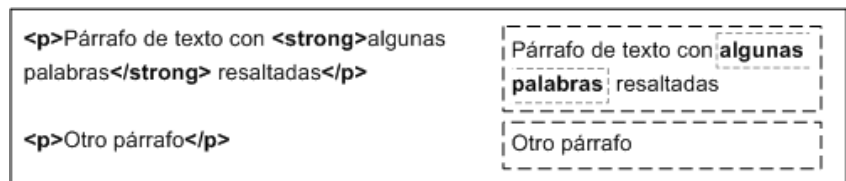
- **Guía de Propiedades CSS por Grupos**
<https://www.w3schools.com/cssref/default.asp>
- **Guía de Selectores CSS**
https://www.w3schools.com/cssref/css_selectors.asp
- **CheatSheets de CSS2**
<https://www.cheatography.com/davechild/cheat-sheets/css2/>
- **CheatSheets de CSS3**
<https://github.com/hhkaos/cursohtml5desdecero/blob/master/images/css3-cheat-sheet.pdf>

HTML y CSS

Modelo de cajas

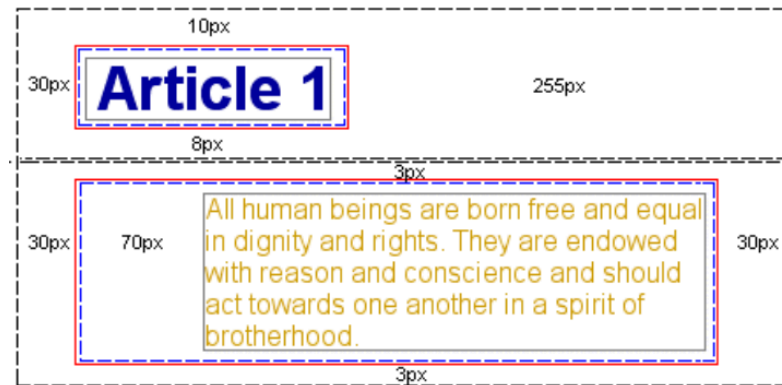
El modelo de cajas o "box model" es seguramente la característica más importante del lenguaje de hojas de estilos CSS, ya que condiciona el diseño de todas las páginas web. El modelo de cajas es el comportamiento de CSS que hace que todos los elementos de las páginas se representen mediante cajas rectangulares.

Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta HTML, se crea una nueva caja rectangular que encierra los contenidos de ese elemento. La imagen muestra tres cajas rectangulares que crean las tres etiquetas HTML que incluye la página.

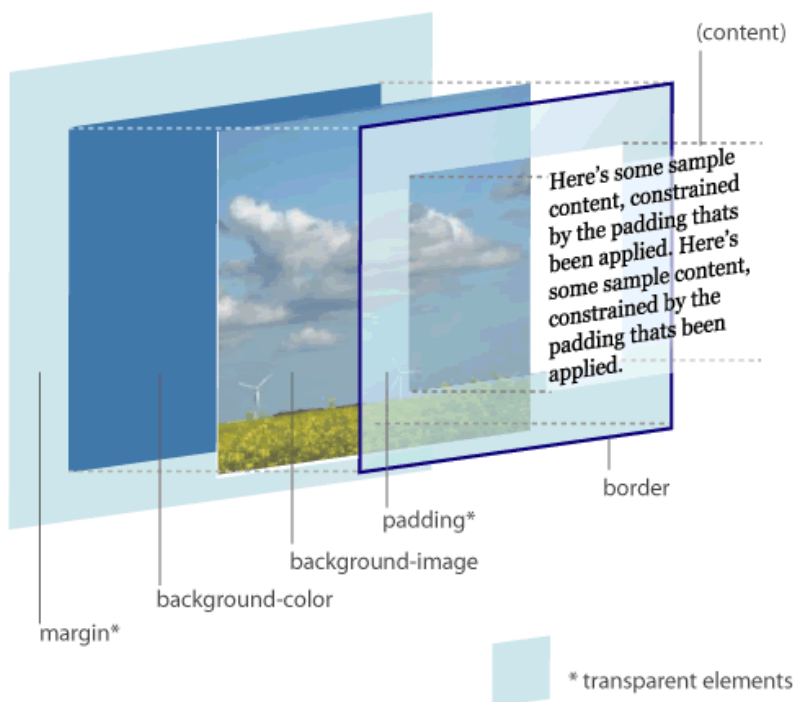


Los navegadores crean y colocan las cajas de forma automática, pero CSS permite modificar todas sus características.

Cada una de las cajas está formada por seis partes, tal y como muestra la imagen de abajo.

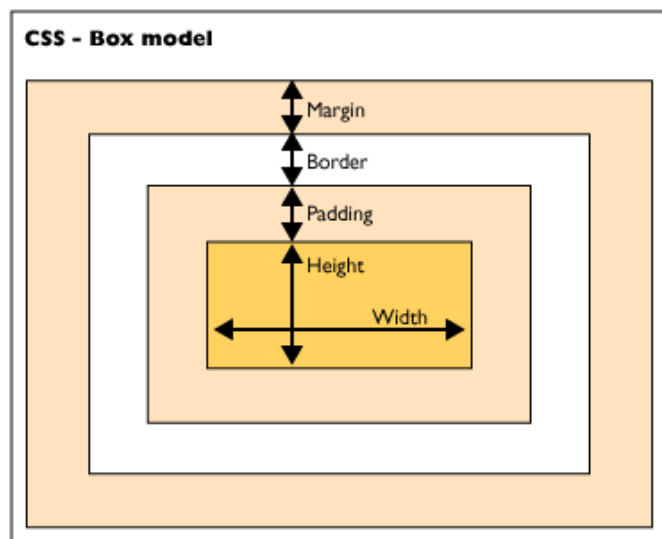


THE CSS BOX MODEL HIERARCHY



Las partes que componen cada caja y su orden de visualización desde el punto de vista del usuario son las siguientes:

- **Contenido (content)**
El contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
- **Relleno (padding)**
Espacio libre opcional existente entre el contenido y el borde. Es transparente.
- **Borde (border)**
Línea que encierra completamente el contenido y su relleno.
- **Imagen de fondo (background image)**
Imagen opcional que se muestra por detrás del contenido y el espacio de relleno.
- **Color de fondo (background color)**
Color opcional que se muestra por detrás del contenido y el espacio de relleno.
- **Margen (margin)**
Separación opcional existente entre la caja y el resto de cajas adyacentes. Transparente.



EJEMPLO

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: lightgrey;
  width: 300px;
  border: 25px solid green;
  padding: 25px;
  margin: 25px;
}
</style>
</head>
<body>
```

```
<h2>CURSO COMUNIDAD IT!</h2>
```

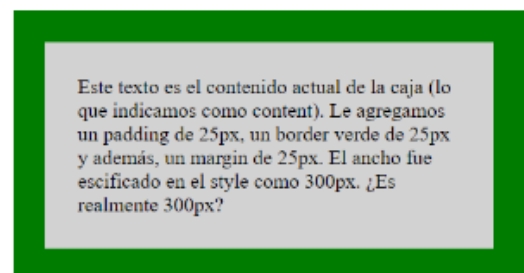
```
<p>A continuación se muestra un ejemplo del modelo de cajas.</p>
```

```
<div>Este texto es el contenido actual de la caja (lo que indicamos como content). Le agregamos un padding de 25px, un borde verde de 25px y además, un margen de 25px. El ancho fue especificado en el style como 300px. ¿Es realmente 300px?</div>
```

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

CURSO COMUNIDAD IT!

A continuación se muestra un ejemplo del modelo de cajas.



Anchura

La propiedad CSS que controla la anchura de la caja de los elementos se denomina **width**.

No admite valores negativos y los valores en porcentaje se calculan a partir de la anchura de su elemento padre.

El siguiente ejemplo establece el valor de la anchura del elemento `<div>` lateral:

```
...
.lateral { width: 200px; }
</div>

<div class="lateral">
```

Altura

La propiedad CSS que controla la altura de los elementos se denomina **height**.

No admite valores negativos y los valores en porcentaje se calculan a partir de la altura de su elemento padre.

El siguiente ejemplo establece el valor de la anchura del elemento `<div>` lateral:

```
...
.cabecera { height: 60px; }
</div>

<div class="cabecera">
```

Margen

CSS define cuatro propiedades para controlar cada uno de los márgenes horizontales y verticales de un elemento: **margin-top**, **margin-right**, **margin-bottom** y **margin-left**.

La propiedad que permite definir de forma simultánea los cuatro márgenes se denomina **margin**.

Ejemplos:

```
/* Código CSS original */
div {
  margin-top: .5em;
  margin-bottom: .5em;
  margin-left: 1em;
  margin-right: .5em;
}

/* Otra alternativa */
div {
  margin: .5em .5em .5em 1em;
}

/* Alternativa directa */
div {
  margin: .5em .5em .5em 1em;
}
```

Relleno

CSS define cuatro propiedades para controlar cada uno de los espacios de relleno horizontales y verticales de un elemento: **padding-top**, **padding-right**, **padding-bottom** y **padding-left**.

La propiedad que permite definir de forma simultánea los cuatro márgenes se denomina **padding**.

Ejemplos:

```
/* Código CSS original */
div {
  padding-top: .5em;
  padding-bottom: .5em;
  padding-left: 1em;
  padding-right: .5em;
}
```

```
/* Alternativa directa */
div {
  padding: .5em .5em .5em 1em;
}
```

```
/* Otra alternativa */
div {
  padding: .5em;
  padding-left: 1em;
}
```

Posicionamiento

Los navegadores crean y posicionan de forma automática todas las cajas que forman cada página HTML. No obstante, CSS permite al diseñador modificar la posición en la que se muestra cada caja.

Utilizando las propiedades que proporciona CSS para alterar la posición de las cajas es posible realizar efectos muy avanzados y diseñar estructuras de páginas que de otra forma no serían posibles.

El estándar de CSS define cinco modelos diferentes para posicionar una caja:

- **Posicionamiento normal o estático**
Se trata del posicionamiento que utilizan los navegadores si no se indica lo contrario.
- **Posicionamiento relativo**
Variante del posicionamiento normal que consiste en posicionar una caja según el posicionamiento normal y después desplazarla respecto de su posición original.
- **Posicionamiento absoluto**
La posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.
- **Posicionamiento fijo**
Variante del posicionamiento absoluto que convierte una caja en un elemento inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.
- **Posicionamiento flotante**
Se trata del modelo más especial de posicionamiento, ya que desplaza las cajas todo lo posible hacia la izquierda o hacia la derecha de la línea en la que se encuentran.

position

static: corresponde al posicionamiento normal o estático. Si se utiliza este valor, se **ignoran** los valores de las propiedades **top**, **right**, **bottom** y **left** que se verán a continuación.

relative: corresponde al posicionamiento relativo. El desplazamiento de la caja se controla con las propiedades **top**, **right**, **bottom** y **left**.

absolute: corresponde al posicionamiento absoluto. El desplazamiento de la caja también se controla con las propiedades **top**, **right**, **bottom** y **left**, pero su interpretación es mucho más compleja, ya que el origen de coordenadas del desplazamiento depende del posicionamiento de su elemento contenedor.

fixed: corresponde al posicionamiento fijo. El desplazamiento se establece de la misma

forma que en el posicionamiento absoluto, pero en este caso el elemento permanece inamovible en la pantalla.

float

(Valores: `left` | `right` | `none`) Permite posicionar de forma flotante una caja, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.

clear

(Valores: `none` | `left` | `right` | `both`) Permite modificar el comportamiento por defecto del posicionamiento flotante para forzar a un elemento a mostrarse debajo de cualquier caja flotante.

top, right, bottom, left

Indican el desplazamiento horizontal y vertical del elemento respecto de su posición original.

Visualización

Además de las propiedades que controlan el posicionamiento de los elementos, CSS define otras cuatro propiedades para controlar su visualización: `display`, `visibility`, `overflow` y `z-index`.

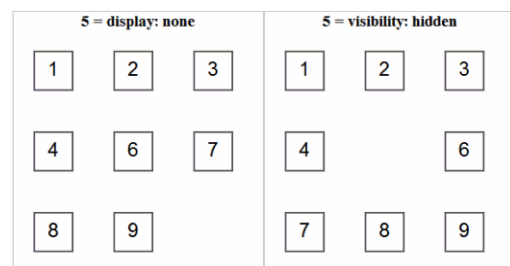
Las propiedades `display` y `visibility` controlan la visualización de los elementos. Las dos propiedades permiten ocultar cualquier elemento de la página. Habitualmente se utilizan junto con JavaScript para crear efectos dinámicos como mostrar y ocultar determinados textos o imágenes cuando el usuario pincha sobre ellos.

display

(Valores: `inline` | `block` | `none` | y otras). Permite controlar la forma de visualizar un elemento e incluso ocultarlo completamente, haciendo que desaparezca de la página. En ese caso resto de elementos de la página se mueven para ocupar su lugar.

visibility

(Valores: `visible` | `hidden` | `collapse`). Permite hacer visible e invisible un elemento. En modo invisible el navegador crea la caja del elemento pero no la muestra, el resto de elementos de la página no modifican su posición, porque aunque la caja no se ve, sigue ocupando sitio.

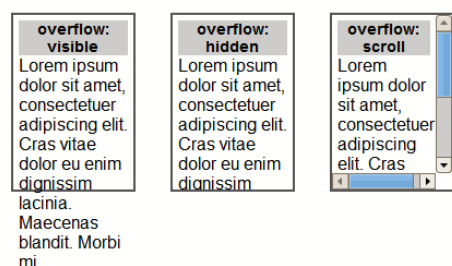


overflow

Permite controlar los contenidos sobrantes de un elemento.

visible: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.

hidden: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la

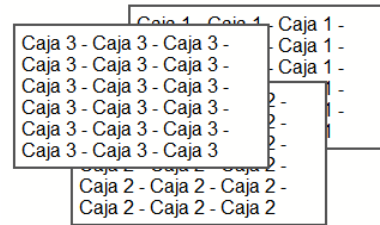


zona reservada para el elemento.

scroll: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de scroll que permiten visualizar el resto del contenido.

z-index

Establece el nivel tridimensional en el que se muestra el elemento. Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja.



Relación entre display, float y position

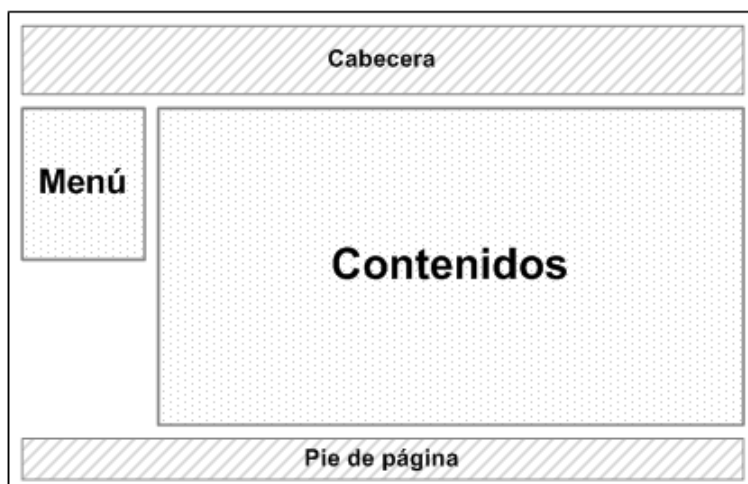
Cuando se establecen las propiedades **display**, **float** y **position** sobre una misma caja, su interpretación es la siguiente:

- Si **display** vale **none**, se ignoran las propiedades **float** y **position** y la caja no se muestra en la página.
- Si **position** vale **absolute** o **fixed**, la caja se posiciona de forma absoluta, se considera que **float** vale **none** y la propiedad **display** vale **block** tanto para los elementos en línea como para los elementos de bloque. La posición de la caja se determina mediante el valor de las propiedades **top**, **right**, **bottom** y **left**.
- En cualquier otro caso, si **float** tiene un valor distinto de **none**, la caja se posiciona de forma flotante y la propiedad **display** vale **block** tanto para los elementos en línea como para los elementos de bloque.

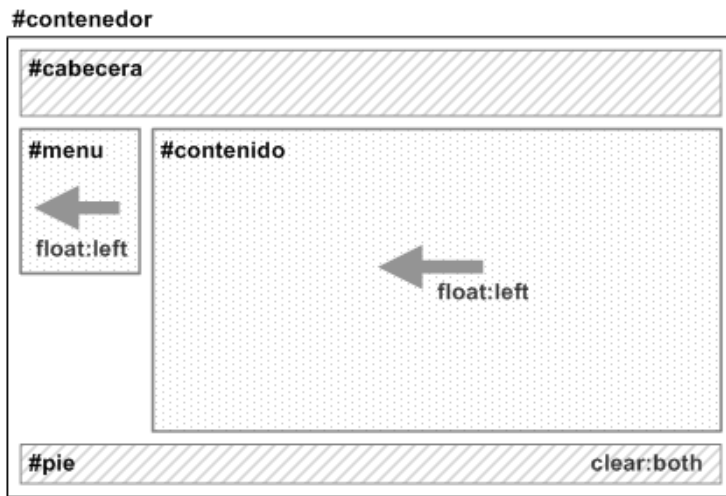
Estructura o Layout

El diseño de las páginas web habituales se divide en bloques: cabecera, menú, contenidos y pie de página. Visualmente, los bloques se disponen en varias filas y columnas.

Diseño a 2 columnas con cabecera y pie de página



El objetivo de este diseño es definir una estructura de página con cabecera y pie, un menú lateral de navegación y una zona de contenidos. La anchura de la página se fija en **700px**, la anchura del menú es de **150px** y la anchura de los contenidos es de **550px**.



La solución CSS se basa en el uso de la propiedad **float** para los elementos posicionados como el menú y los contenidos y el uso de la propiedad **clear** en el pie de página para evitar los solapamientos ocasionados por los elementos posicionados con **float**.

El código HTML y CSS mínimos para definir la estructura de la página sin aplicar ningún estilo adicional son los siguientes:

```

<!-- fragmento de HTML a correspondiente al body del documento -->
<body>
  <div id="contenedor">
    <div id="cabecera">
    </div>

    <div id="menu">
    </div>

    <div id="contenido">
    </div>

    <div id="pie">
    </div>
  </div>
</body>

/* Contenido del archivo externo CSS */
#contenedor {
  width: 700px;
}
#cabecera {
}
#menu {
  float: left;
  width: 150px;
}
#contenido {
  float: left;
  width: 550px;
}
  
```

```
#pie {
    clear: both;
}
```

En los estilos CSS anteriores se ha optado por desplazar tanto el menú como los contenidos hacia la izquierda de la página (`float: left`). Sin embargo, en este caso también se podría desplazar el menú hacia la izquierda (`float: left`) y los contenidos hacia la derecha (`float: right`).

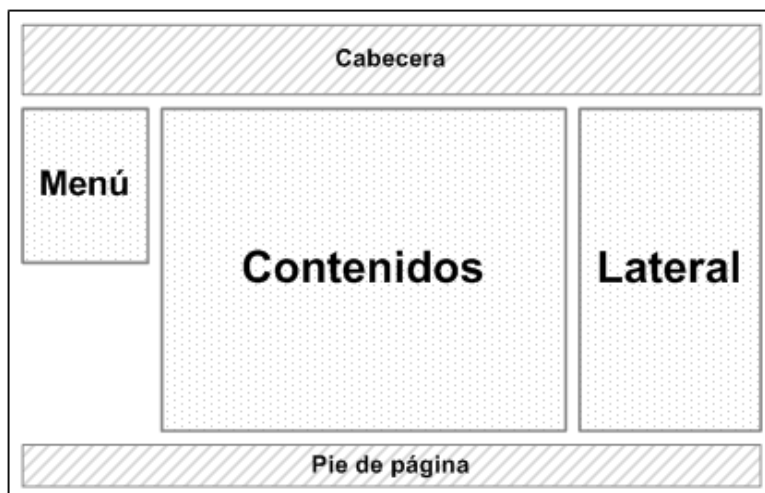
El diseño anterior es de anchura fija, lo que significa que no se adapta a la anchura de la ventana del navegador.

Para conseguir una página de anchura variable y que se adapte de forma dinámica a la ventana del navegador, se deben aplicar las siguientes reglas CSS:

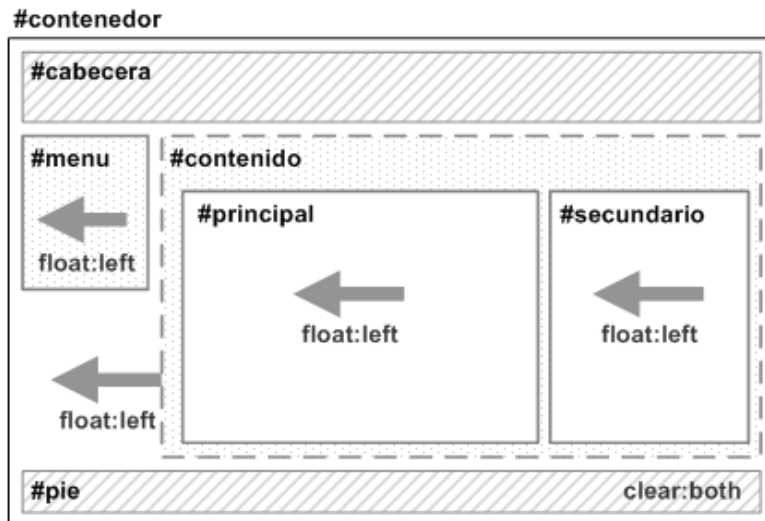
```
/* Contenido del archivo externo CSS */
#contenedor {
}
#cabecera {
}
#menu {
    float: left;
    width: 15%;
}
#contenido {
    float: left;
    width: 85%;
}
#pie {
    clear: both;
}
```

Si se indican la anchuras de los bloques que forman la página en porcentajes, el diseño final es dinámico. Para crear diseños de anchura fija, basta con establecer las anchuras de los bloques en píxel.

Diseño a 3 columnas con cabecera y pie de página



Además del diseño a dos columnas, el diseño más utilizado es el de tres columnas con cabecera y pie de página. En este caso, los contenidos se dividen en dos zonas diferenciadas: zona principal de contenidos y zona lateral de contenidos auxiliares.



La solución CSS emplea la misma estrategia del diseño a dos columnas y se basa en utilizar las propiedades `float` y `clear`.

El código HTML y CSS mínimos para definir la estructura de la página sin aplicar ningún estilo adicional son los siguientes:

```

<!-- fragmento de HTML a correspondiente al body del documento -->
<body>
  <div id="contenedor">
    <div id="cabecera">
    </div>

    <div id="menu">
    </div>

    <div id="contenido">

      <div id="principal">
      </div>

      <div id="secundario">
      </div>

    </div>

    <div id="pie">
    </div>
  </div>
</body>

/* Contenido del archivo externo CSS */
#contenedor {
}
#cabecera {

```

```

}
#menu {
    float: left;
    width: 15%;
}
#contenido {
    float: left;
    width: 85%;
}
#contenido #principal {
    float: left;
    width: 80%;
}
#contenido #secundario {
    float: left;
    width: 20%;
}
#pie {
    clear: both;
}

```

El código anterior crea una página con anchura variable que se adapta a la ventana del navegador. Para definir una página con anchura fija, solamente es necesario sustituir las anchuras en porcentajes por anchuras en píxel.

Al igual que sucedía en el diseño a dos columnas, se puede optar por posicionar todos los elementos mediante `float: left` o se puede utilizar `float: left` para el menú y la zona principal de contenidos y `float: right` para el contenedor de los contenidos y la zona secundaria de contenidos.

Ejercicio 5: Librería

Escribir un sitio web en HTML para una Librería. Debe contener:

- **Layout:** unificado para todo el sitio con cabecera, menú, bloque de contenido y pie.
 - Hoja de Estilos centralizada y estructura de directorios organizada separando los css e imágenes en directorios aparte.
 - Cabecera: con imagen de fondo (background-image), logo y nombre de la librería.
 - Menú: acceso a Inicio, cada sección, registración y contacto.
 - Pie de página: al menos copyright, redes sociales y términos y condiciones.
- **Página principal:** debe mostrar bloques para cuatro temas (título con enlace a la sección del tema) y un libro destacado en cada uno (imagen de la tapa, título y autor).
- **Secciones:** una para cada una de los cuatro temas, debe mostrar 6 libros cada una (título, autor, imagen y descripciones de libros).
- **Registración:** Formulario completo solicitando datos personales.
 - Nombre (*text*)
 - Apellido (*text*)
 - Fecha de Nacimiento (*date*)
 - Email (*email*)
 - Contraseña (*password*)
 - Sexo (*radio*)
 - Tema Favorito (*select* de los temas)
 - Botón (*submit*)

- Contacto: Dirección y datos de contacto (teléfonos, email). Formulario de contacto.

IMPORTANTE: Repasar requerimientos y hacer **wireframes** antes de comenzar con el código.

HTML5

¿Qué es HTML5?

El HTML5 (HyperText Markup Language, versión 5) es la quinta revisión del lenguaje “básico” de la World Wide Web, el HTML.

Esta nueva versión pretende reemplazar a (X)HTML, corrigiendo problemas con los que los desarrolladores web se encuentran, así como rediseñar el código actualizándolo a nuevas necesidades que demanda la web de hoy.

Etiquetas básicas

```
<!DOCTYPE html>
<html lang="es">
...
</html>
```

Para iniciar un documento con soporte para tags de HTML5, se debe incluir el elemento `<!DOCTYPE html>` como elemento inicial. Eso le indica al navegador que su contenido debe ser interpretado como código HTML5.

El atributo `lang="es"` indica el idioma utilizado, en nuestro caso **Español**.

```
<head>...</head>
```

Define la cabecera del documento HTML, se mantiene tal como en versiones anteriores a HTML5.

```
<title>...</title>
```

Define el título de la página. Se define dentro del `<head>`. Siempre debe estar presente.

```
<meta />
```

Tal como en versiones anteriores de HTML, se utiliza para agregar meta-información a la página, con distintos fines. Uno muy importante en HTML5 es la definición de la visualización del viewport para smartphones y tablets:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
<body>...</body>
```

Define el contenido principal o cuerpo del documento. En HTML5, el elemento `<div>` pasa a ser un contenedor genérico y aparecen nuevas **etiquetas semánticas** que son más adecuadas, se mantiene tal como en versiones anteriores a HTML5.

```
<header>...</header>
```

Representa el encabezado de la página. Por lo general contiene el logotipo, el título principal y la barra de navegación (`<nav>`). Ésta información hace posible que los buscadores anclen la temática de la página con respecto al sitio.

```
<nav>...</nav>
```

Alberga el menú principal de navegación, con el cual los spider de los buscadores

pueden hacer un mejor análisis de la arquitectura.

`<section>...</section>`

Tiene la tarea de agrupar las piezas de contenido de la página. Cada página debería tener un `<section>` y varias piezas de contenido `<article>`.

`<article>...</article>`

En estos tags se agregan los contenidos primordiales de la página. Nos ayuda a tener mayor control sobre lo que queremos resaltar para la indexación.

`<aside>...</aside>`

Por lo general se utiliza para colocar información adicional a la página. Su uso más común es como una columna a uno de los lados de la página. Por llevar contenido secundario, los spiders le dan menor relevancia.

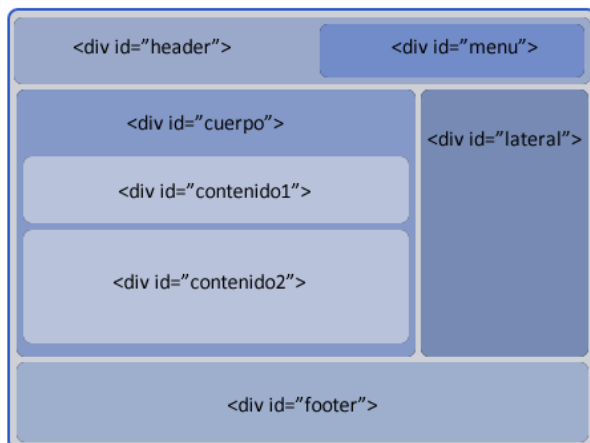
`<footer>...</footer>`

Indica el pie de la página, por lo general contiene información como autor, enlaces, copyright, redes sociales, términos y condiciones, entre otros.

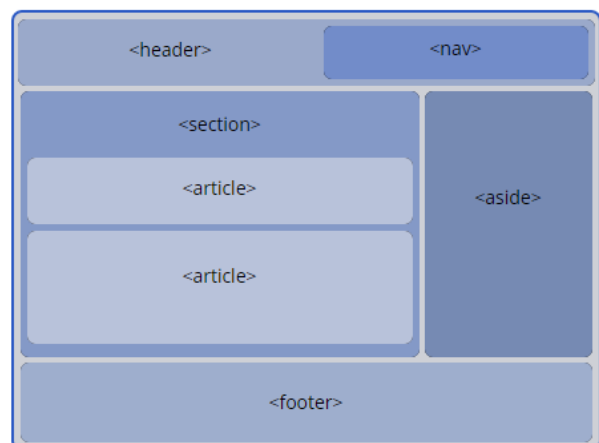
Comparación de Layout entre HTML4 y HTML5

A continuación se expone un ejemplo, en el cual se hace una comparación de dos esquemas de maquetación para un sitio básico de una página, uno en HTML4 y el otro en HTML5. Se supone una página con encabezado, pie, una sección con múltiples contenidos y una barra lateral; podría tratarse por ejemplo de un blog o una página de noticias.

HTML4



HTML5



El armado en HTML4 está basado en divs, que identifican su función mediante su atributo id. El mismo sitio en HTML5 está completamente planteado con **etiquetas semánticas**, lo que clarifica cada bloque.

Otros Agregados

`<audio>` y `<video>`

Nuevos elementos que permiten incrustar contenidos multimedia de sonido o de vídeo, respectivamente. Es una de las novedades más importantes e interesantes en este HTML5, ya que permite reproducir y controlar vídeos y audio sin necesidad de plugins externos, como Flash o Java.

Ejemplos: https://www.w3schools.com/html/html5_audio.asp
https://www.w3schools.com/html/html5_video.asp

`<canvas>...</canvas>` (Lienzo)

El navegador reserva una parte de la ventana para el uso de gráficos, los cuales serán dibujados ahí. El elemento en sí es fácil de implementar en términos de HTML, pero su verdadera manipulación se hace totalmente por medio del uso de JavaScript.

Ejemplos: <http://corehtml5canvas.com/code-live/>
https://www.w3schools.com/html/html5_canvas.asp

`<svg>...</svg>` (Gráficos Vectoriales)

Permite construir gráficos vectoriales. Utiliza una estructura XML, es decir, también utiliza etiquetas, pero éstas son propias de `<svg>` y no se pueden utilizar fuera de su ámbito. Estas etiquetas propias tienen atributos para permitir posición y apariencia, pero se puede utilizar JavaScript y CSS sin ningún problema.

La característica más popular de `<svg>` es que permite escalar imágenes sin perder calidad.

Ejemplos: https://www.w3schools.com/html/html5_svg.asp

Mejoras en los formularios

Los formularios es uno de los puntos que tuvieron mejoras considerables en HTML5. Se incorporaron **nuevos tipos de input** y **nuevos atributos** que permiten un mayor nivel de personalización y funcionalidad.

Aunque se agregaron campos de mucha utilidad, como los relacionados con las fechas, quizá el punto más importante sea la incorporación de **validación propia para poder verificar los datos que el usuario introduce**. En versiones anteriores de HTML, esta verificación había que hacerla usando algún lenguaje de programación como JavaScript, en HTML5 es posible hacerlo sin escribir una sola línea de código.

Los nuevos input type que se añadieron en HTML5 son: `number`, `range`, `date`, `time`, `datetime`, `datetime-local`, `week`, `month`, `color`, `search`, `email`, `tel` y `url`.

Ejercicio 6: HTML5

Escribir un sitio web completo cuyo layout esté armado en HTML5 y además respete las convenciones y recomendaciones que vimos.

Pueden tomar como base el ejercicio de la Librería o bien un sitio institucional que contenga varias páginas:

- Layout: unificado para todo el sitio con cabecera, menú, bloque de contenido y pie.
 - Hoja de Estilos centralizada y estructura de directorios organizada separando los css e imágenes en directorios aparte.
 - Cabecera: con imagen de fondo (background-image), logo y nombre de la librería.
 - Menú: acceso a Inicio, cada sección, registración y contacto.

- Pie de página: al menos copyright, redes sociales y términos y condiciones.
- Página principal.
- Secciones: al menos cuatro.
- Registración: Formulario completo solicitando datos personales (utilizar al menos 4 de los type agregados en HTML5) .
- Contacto: Dirección y datos de contacto (teléfonos, email). Formulario de contacto.

IMPORTANTE: Repasar requerimientos y hacer **wireframes** antes de comenzar con el código.

JavaScript

Ejercicios

1. Definir una variable numérica, asignarle un valor y sumarle 5.
2. Definir dos variables de cadenas, asignarles valores y concatenarlas.
3. Evaluar si dos números son iguales, diferentes, mayor o menor. Resolver utilizando "if"/"else".
4. Utilizando "switch". Definir una variable numérica. Asignarle un valor entre 1 y 10; mostrar a qué grupo pertenece:
 - Grupo 1: del 1 al 3
 - Grupo 2: del 4 al 6
 - Grupo 3: del 7 al 10
 - Modifiquemos el ejercicio para que el número lo ingrese el usuario (con "prompt").
5. Realizar la sumatoria de 0 a 10 y devolver el valor de la misma.
6. Generar un array con 10 números, recorrerlo e ir multiplicando todos los elementos, finalmente obtener el producto total.
7. Crear una función que reciba dos valores y retorne el producto de los mismos.
8. Crear una función que reciba dos cadenas y retorne la concatenación de la misma.
9. Crear una función, a partir de la lógica aplicada en ejercicio 3, que reciba dos valores y muestre cuál es el mayor. En caso de ser iguales, deberá indicarlo.
10. Crear una función que reciba un número y muestre tantos asteriscos como la cantidad de veces que se pasó como parámetro.
11. Crear una función que reciba el monto de un producto, y el medio de pago: C (tarjeta de crédito), E (efectivo) y D (tarjeta de débito).

Si el monto del producto es menor a \$200 no se aplicará ningún descuento, pero si el monto a abonar es entre \$200 y \$400 se aplicará un descuento del 30% si el medio de pago es efectivo, 20% si se realiza con débito y 10% con tarjeta de crédito.

Para montos mayores a \$400, el descuento es el mismo sin importar el medio de pago, dicho descuento es del 40%.

La función deberá retornar el monto final a abonar.
12. Crear una función que reciba un número que represente la altura de un medio-árbol. Deberá generar de manera escalonada el mismo. Ejemplo: si la altura es 5 deberá **mostrar**:

```
*
* *
* * *
* * * *
* * * * *
```
13. Crear una función que reciba un número que indica el día de la semana y retorne una cadena de texto indicando a qué día corresponde. Ejemplo: si es 1, deberá retornar lunes, 2 retornará martes, y así siguiendo. Si el día es 6 o 7 deberá retornar "fin de semana". En caso de un valor que no represente un día de la semana deberá retornar un mensaje de error.
14. Crear una función que genere un array de varios elementos numéricos y muestre por pantalla el promedio de esos números. El tamaño y los valores deben ser ingresados por el usuario (comando prompt) en dicho orden.

TIP: El dato ingresado con prompt es de tipo string, usar split() para quitar los espacios y usar la función Number para transformarlo.
15. Utilizar la función que genera el medio-árbol (ejercicio 12): crear un campo de entrada que le permita al usuario ingresar la altura del mismo. Incluir un botón que al presionarlo,

invoque a la función generada previamente con el valor ingresado por el usuario para que la misma muestre el medio-árbol.

Deberá incluir validación de datos ingresados por el usuario.

16. Desarrollemos un portero eléctrico:

- Tendrá dos visores, de dos posiciones el piso y una posición para dpto. Los pisos van del 00 al 48. Los dptos, del 1 al 6.
- El botón llamar, muestra el mensaje de abajo. El botón borrar limpia los visores y el mensaje de abajo.
- Si se hace referencia a un piso y/o dpto que no existe, mostrar el error en el visor de abajo.

PISO		DPTO
1	2	3
4	5	6
7	8	9
0	Llamar	Borrar
Llamando al piso NN, dpto NN		

17. Desarrollemos un teclado en pantalla:

- Cada línea del teclado debe hacerse en un array.
- Al presionar cada tecla (botón) deberá mostrarse en el display.
- La muestra estará centralizada en una sola función.
- Debe existir un botón para borrar el display.
- Botón Backspace.