CAPÍTULO 8 – ETIQUETAS SEMÁNTICAS Y DE ORGANIZACIÓN EN HTML

INTRODUCCIÓN

Las etiquetas semánticas y de organización nos van a servir para estructurar de una manera lógica nuestra página web y para describir algunos elementos de la página. Con ello facilitaremos que los motores de búsqueda encuentren mejor la información que transmitimos y mejoramos la legibilidad del código al hacerlo más coherente facilitando el reconocimiento de sus distintas secciones.

Estas etiquetas no tienen ninguna influencia sobre la presentación y visualización de la página. Solo se utilizan para mejorar la estructura.

El concepto de web semántica es una idea original y muy defendida por Tim Berners-Lee el inventor de la web.

Si piensa en las páginas web que podemos ver hoy en día en la red, todas tienen una estructura muy semejante, en la que podemos distinguir los siguientes elementos:

1.- Un encabezado que suele contener un logotipo, un banner, el nombre del sitio web, una caja de búsqueda…

2.- Un menú de navegación para pasar a las diferentes secciones del sitio web.

3.- El contenido propiamente dicho de la página web.

4.- Una parte con publicidad u otra información relacionada con la página web.

5.- Un pie de página que suele contener, el copyright, la empresa autora de la web, indicaciones legales, etc, etc.

Para explicarlo visualmente, la mayoría de las páginas web que visitamos en la actualidad suelen tener alguna de estas formas:

A)

Encabezado de página

Navegación

Contenido de la Página

Pie de página.

B)

Encabezado de página

Navegación

Contenido de la Página

Pie de página.

C)

Encabezado de página

Navegación

Contenido de la Página

Pie de página.

Información

adicional

Para indicar a **HTML** la estructura lógica de nuestra página web utilizamos las etiquetas semánticas y de organización.

ETIQUETAS SEMANTICAS DE ORGANIZACIÓN:

HEADER

La etiqueta <**header**> indica al navegador la cabecera de nuestra página web. En ella generalmente podemos encontrar el nombre del sitio web, una imagen, un logotipo, un elemento h1-h6, un formulario de búsqueda, etc, etc.

***Veámoslo en la práctica 8.1.***

NAV

La etiqueta <**nav**> indica al navegador una sección de enlaces o links de navegación.

***En nuestra página web de ejemplo podríamos utilizarla como en la práctica 8.2.***

SECTION

La etiqueta <**section**> se utiliza para indicar una sección general o principal dentro de un documento. El ejemplo más claro para diferenciar una sección sería el capítulo de un libro. Dentro de <**section**> podemos incluir subsecciones y si utilizamos **h1-h6**, podemos estructurar mejor toda la página creando jerarquías en el contenido, algo muy favorable para el buen posicionamiento web.

***Apliquemos esta etiqueta a nuestra página realizando la práctica 8.3.***

ASIDE

La etiqueta <**aside**> especifica una sección de la página que abarca un contenido tangencialmente relacionado con el contenido que lo rodea, por lo que se le puede considerar un contenido independiente. Se puede utilizar para barras laterales, elementos publicitarios, grupos de enlaces de navegación u otro contenido que consideremos separado pero relacionado con el contenido principal de la página.

En nuestra web de práctica crearemos una barra lateral con cierto contenido que no es la información principal de nuestra web. ***¡Veámoslo realizando la práctica 8.4!***

Al igual que en las etiquetas semánticas y de organización precedentes <**aside**> solamente especifica que el texto que abarca es una información adicional o paralela a la ofrecida en el bloque principal de la web. Aunque pretendemos que aparezca en una barra lateral, como esta etiqueta no da formato, el texto sigue apareciendo debajo del resto de la página web. Seguimos esperando a las posibilidades de **CSS** para conseguir la presentación que deseamos.

FOOTER

La etiqueta <**footer**> indica el pie de una página web o una sección de ella. En el pie habitualmente se muestra información acerca de la página que tiene poco que ver con su contenido, como el autor, el copyright, el año, etc, etc…

***En nuestra web de ejemplo está claro lo que pondremos como pie de página… Para incluir el <footer> en nuestra web realizamos la práctica 8.5.***

ARTICLE

La etiqueta <**article**> El elemento de artículo representa un componente de una página que consiste en un texto autónomo en un documento, página, aplicación, o sitio web con la intención de que pueda ser reutilizado y repetido. Habitualmente se utiliza en los post de los foros, las entradas de un blog, un comentario escrito por un usuario, etc, etc.

***En nuestra web de práctica aplicaríamos <article> realizando la práctica 8.6.***

Tenemos que tener en cuenta que nada impide utilizar etiquetas <**nav**> en los <**footer**>, <**header**> dentro de <**section**>, <**header**> dentro de <**article**>, etc, etc, siempre que nos ayuden a tener más clarificado el código **HTML**. Por ejemplo se puede incluir más de un elemento de cabecera de página <**header**>, podemos tener una cabecera de página para todo el sitio web y también una cabecera para una sección concreta. Sin embargo, no se debe abusar de las etiquetas de organización ya que se corre el peligro de “ensuciar” el código más que organizarlo.

DIV

¿Y qué pasa si en algún momento necesitamos hacer una división o crear una sección dentro del contenido de la página web que no coincida exactamente con la cabecera, el pie, <**section**>, <**aside**>, <**nav**> o <**article**>? En esos casos tendremos que utilizar la etiqueta <**div**>.

Podemos considerar a la etiqueta <**div**> como un contenedor genérico que nos permitirá crear un bloque de contenido totalmente independiente. Por sí solo, la etiqueta <**div**> no dice nada. Se utiliza para agrupar contenido en bloque y darle formato mediante el uso de atributos **ID** o identificadores de clase y **CSS**.

En una página web podemos tener todos los elementos <**div**> que consideremos necesarios. De hecho, hasta la llegada de **HTML5** se utilizaba esta etiqueta para crear el armazón visual de la página web.

***Para ver un ejemplo de utilización de la etiqueta <div> realice la práctica 8.7.***

OTRAS ETIQUETAS SEMÁNTICAS

<**abbr**>. Indica una abreviatura, por ejemplo S.A. o IVA. Además sirve para indicar a los programas de síntesis vocal que no traten de leer la palabra tal y como está escrita, sino deletreándola.

<**address**>. Indica una dirección de contacto del autor o propietario de un documento o artículo. No debe usarse para describir una dirección postal a menos que sea parte de la información de contacto. Es uno de los elementos típicos de <**footer**>. Su contenido se visualiza por defecto en cursiva y con letra más pequeña.

<**cite**>. Indica el título de una obra. Por ejemplo: un libro, una canción, una película... Se visualiza en cursiva.

<**code**>. Señala una sintaxis o un código informático. Se visualiza en un tipo de letra de paso fijo.

<**samp**>. Pone de relieve un texto de ejemplo. Se visualiza en un tipo de letra de paso fijo.

<**dfn**>. Indica la definición de un término. Se visualiza en cursiva.

<**kbd**>. Indica al usuario las letras del teclado que tiene que pulsar. Se visualiza en un tipo de letra de paso fijo.

<**var**>. Indica que contiene una variable. Se visualiza en cursiva.

<**strong**>. Ya vista en la página 21.

<**em**>. También vista en la página 21.

<**hgroup**>. Indica que se trata de un grupo de títulos <hx>.

<**hr**>. Ya vista en la página 14.

<**time**>. Define una fecha o una hora. El atributo opcional “datetime” indica la fecha y la hora siguiendo la representación numérica internacional de la norma ISO8601.

<**mark**>. Remarca parte del texto que se muestra subrayado sobre un fondo amarillo.

<**meter**>. Define una medida. Se usa para medidas con valores máximos y mínimos conocidos como por ejemplo uso del disco. Solo es reconocida por Chrome y Opera. Esta etiqueta tiene seis atributos:

* **value**: el valor del dato sobre la escala.
* **min**: el valor mínimo posible.
* **low**: el valor mínimo esperado.
* **high**: el valor máximo esperado.
* **max**: el valor máximo posible.
* **optimum**: el valor máximo ideal.

<**progress**>. Representa el progreso de una tarea. Se utiliza conjuntamente con JavaScript. Tiene dos atributos:

* **max**: Especifica que cantidad de trabajo requiere la tarea total.
* **value**: Especifica la cantidad de tarea que se ha completado.

<**figure**>. Sirve para agrupar elementos tales como imágenes y videos.

<**figcaption**>. Usada de forma conjunta con la etiqueta <**figure**> proporciona una leyenda a los elementos agrupados.

<**details**>. Indica detalles o contenido accesorio. Actualmente solo es soportada por chrome.

<**summary**>. Usada conjuntamente con la etiqueta <**details**>, proporciona un resumen de los elementos accesorios. Actualmente solo es soportada por chrome.

CATEGORÍAS DE CONTENIDO EN HTML

Llegados a este punto de nuestro curso es conveniente hacer una reflexión sobre qué tipo de contenido podemos encontrar en una página web. Esto nos ayudará a entender o clarificar cómo se organiza la información y cómo la entienden los navegadores.

Podemos dividir el contenido de una página web en varias categorías:

* **Metadatos**: Establece el comportamiento del resto del contenido de la página. Un ejemplo de metadatos es la etiqueta <**meta charset**> que hemos utilizado para indicar el juego de caracteres que utilizamos en nuestra página web. Los metadatos se ubican en la sección <**head**> del documento. Veremos más sobre los metadatos más adelante.
* **Flujo**: Son los elementos que constituyen el contenido de una página web. Normalmente será texto o un archivo insertado, como una imagen o un video.
* **Secciones**: Se trata de un subconjunto del contenido de flujo. A las secciones corresponden algunas etiquetas que hemos estudiado en este tema: <**section**>, <**article**>, <**nav**>, <**aside**>, <**header**>, <**footer**>.
* **Títulos**: Se trata de los titulares de la página web, las etiquetas <**h1**>, <**h2**> y sucesivos, además de la etiqueta <**hgroup**>. El contenido de los titulares forma parte del contenido de flujo.
* **Textos**: Engloba todo el texto de la página web. Es un subconjunto del contenido de flujo.
* **Contenidos embebidos**: Son aquellos que se importan de otros recursos dentro de la página, como imágenes, videos, archivos para descargar…
* **Contenidos interactivos**: La etiqueta <**a**> y otros elementos, como los formularios que veremos más adelante, y que permiten interactuar con el usuario de la página web.

ELEMENTOS EN BLOQUE Y ELEMENTOS EN LÍNEA

Otro concepto que debemos tener claro para ser más eficientes en nuestros diseños web es la distinción entre elementos de bloque y elementos de línea.

Las etiquetas de renderizado en bloque ocupan por defecto toda la longitud de la ventana del navegador o del elemento que las contiene. Se sitúan unas debajo de otras y generalmente el navegador inserta automáticamente un espacio encima y debajo del contenido de la etiqueta.

Así por ejemplo el código:

**<p>Párrafo 1</p><p>Párrafo 2</p>**

aunque está escrito en una sola línea, en el navegador se visualiza en dos líneas puesto que la etiqueta <**p**> es una etiqueta de tipo bloque. Cada párrafo va a ocupar una línea.

Esto es lo que nos ha sucedido en la última modificación del archivo “**turismo.htm**” que nos muestra primero la imagen y debajo de ella el título centrado de la página. De la misma manera, el resto de contenido del archivo, va situándose uno debajo del otro en toda la web.

Algunas etiquetas de bloque son: <**p**>, <**hx**>, <**hr**>, <**ol**>, <**ul**>, <**dl**>, <**div**>, <**header**>, <**footer**>, <**section**>, <**article**>, <**nav**>, etc, etc…

Por el contrario las etiquetas en línea se sitúan siempre uno al lado del otro permaneciendo en el mismo flujo de texto.

Así por ejemplo el código:

**texto en <b>negrita</b> y en <i>cursiva</i>**

se escribe en una sola línea sin interrumpir el flujo del texto.

Algunas etiquetas en línea son: <**b**>, <**i**>, <**br**>, <**a**>, <**img**>, <**span**>…

***Para terminar de dominar este tema realice ahora la ejercitación del capítulo 8.***

CAPÍTULO 9.- PROPIEDADES CSS DE CAJA

Y VISUALIZACIÓN

HOJAS DE ESTILO EXTERNAS, HERENCIA Y HOJAS EN CASCADA

**HOJAS DE ESTILO EXTERNAS**.

En temas anteriores hablamos de las formas de incorporar las hojas de estilo **CSS** a nuestras páginas web. Hasta ahora hemos utilizado estilos incorporados en el mismo archivo de código **HTML**, situados dentro de la sección de cabecera <**head**> de la página. Esta modalidad recibe el nombre de hoja de estilos interna.

Además de hojas de estilo internas, también existen las hojas de estilos externas y los estilos aplicados directamente “**en línea**”, en la propia etiqueta **HTML**.

Una hoja de estilo externa es un documento independiente con la extensión .**css**. Cuando utilizamos un archivo externo, en él únicamente se definen estilos **CSS** y en el de código **HTML** tenemos que insertar un enlace a dicho archivo.

La ventaja de utilizar hojas de estilo externas es que una misma hoja de estilo se puede utilizar para distintos archivos de código **HTML**, mientras que en las hojas de estilo internas, las reglas **CSS** solo se aplican al documento en donde residen.

Por ejemplo, si tenemos un sitio web con 14 páginas y utilizamos hojas de estilo internas, debemos crear una hoja de estilo dentro de cada una de esas 14 páginas. Si tuviéramos que hacer un cambio cualquiera en los estilos tendríamos que modificar todas y cada una de las hojas lo que supone una pérdida de tiempo muy considerable y una posible fuente de errores y equivocaciones. Es lo que hemos tenido que hacer hasta ahora con nuestras webs de prácticas referidas al turismo.

Otra de las ventajas es que este método de utilizar hojas de estilo externas para dar formato a una página web respeta al máximo las prescripciones del **W3C** de separación entre contenido y presentación de las que tantas veces hemos hablado en este curso.

**NOCIÓN DE CASCADA**

Nada impide que para un mismo archivo de código **HTML** utilicemos hojas de estilo en línea, internas y externas a la vez. En este caso se podría dar algún tipo de conflicto u orden de estilo contradictorio entre los distintos formatos que indiquemos en ellas que queremos aplicar. Si fuera así se aplicaría la **NOCIÓN DE CASCADA**, (recuerde que **CSS** es Cascading Style Sheets), u orden de prioridad que de menor a mayor es:

* Propiedades de formato por defecto del navegador.
* Hojas de estilo externas.
* Hojas de estilo internas.
* Hojas de estilo en línea.

Por tanto la regla de prioridad, para visualizar el documento en el navegador, consiste en utilizar la regla de estilo más próxima a la etiqueta **HTML**.

**NOCIÓN DE HERENCIA**

Un último concepto teórico que debemos dominar llegados a este punto, es el de **HERENCIA**. La herencia en **CSS** es el mecanismo mediante el cual determinadas propiedades de un elemento padre se transmiten a sus hijos. Por ejemplo, si aplicamos un tamaño de fuente a la etiqueta <**section**> todo el texto, de todas las etiquetas contenidas dentro de <**section**> heredan ese tamaño de fuente. Podríamos decir por tanto que una etiqueta <**p**> contenida dentro de <**section**> es hija de <**section**>. O al revés que <**section**> es el contenedor padre de <**p**>.

Para entender el porqué de la herencia basta con pensar que pasaría si no existiera… ¡Entonces tendríamos que especificar individualmente todo el formato, familia de fuente, tamaño, color, etc, etc, para todos y cada uno de los elementos de nuestro código **HTML**.!

Pero es hora de ponernos manos a la obra. ***Procederemos a crear una hoja de estilo externa realizando la práctica 9.1.***

HOJAS DE ESTILO DE RESET

Muchos diseñadores web utilizan “archivos de reset de **CSS**”. Se trata de una hoja de estilo externa que “pone a cero” todos los valores de formato aplicados por defecto por los diferentes navegadores.

Efectivamente cada navegador web aplica sus propias reglas para mostrar los elementos **HTML** en pantalla. Esto puede significar que, a pesar de que las diferencias sean mínimas, los diseños web que hayamos creado no se vean igual en distintos navegadores. Por ejemplo, un navegador determinado puede dejar un margen de 10px para un elemento <**h1**> y otro, exagerando un poco, de 15px.

Una manera de homogeneizar las presentaciones en pantalla es crear un archivo **CSS** de reset que elimine los estilos por defecto aplicados a los elementos **HTML** utilizados con más frecuencia. Con este reseteo de valores podemos conseguir una base fiable y coherente sobre la cual empezar nuestro diseño.

Muchos diseñadores experimentados se preparan sus propias hojas de estilo de reset. No obstante existen en internet numerosas distribuciones gratuitas. Nosotros, para ilustrar su uso, utilizaremos el de Eric Mayer que puede consultar en su web <http://meyerweb.com> y que está disponible en su carpeta de prácticas.

***Para ver el resultado de aplicar una hoja de estilos de reset realice la práctica 9.2.***

CONCEPTO DE CAJA O BLOQUE

Otra noción importante que debemos dominar antes de empezar a dar formato a nuestra web es el concepto de caja o bloque. Su dominio nos permitirá situar bien todos los elementos en la web.

Muchas etiquetas **HTML** son tratadas por los navegadores como un elemento de caja o bloque o contenedor, que con todos estos nombres nos lo podemos encontrar. Está constituido por:

* Un contenido
* Un margen interior (padding)
* Un borde (border)
* Un margen exterior (margin)

Margin= margen externo

Border = borde

Padding = margen interno

CONTENIDO

Gráficamente:

Comprendidos los componentes de una caja o contenedor hay que tener en cuenta que a la anchura y/o altura del contenido hay que sumar la anchura del borde y los márgenes para así obtener las medidas finales del contenedor.

Visto con un ejemplo: si tenemos una caja con una anchura del contenido de 200px, un margen interior de 20px, una anchura de borde de 10px y un margen exterior de 20px, la anchura final de todo el contenedor es de 300px (De izquierda a derecha: 20px+10px+20px+200px+20px+10px+20px = 300px).

Las hojas de estilo **CSS** proporcionan al diseñador un control total sobre la altura y anchura del contenido, los márgenes internos y externos y los bordes de un contenedor como vamos a ir viendo.

PROPIEDADES DE ANCHURA Y ALTURA

Ha llegado el momento de ir dando formato a nuestra página web de prueba empezando desde cero.

Las propiedades **width** y **height** fijan respectivamente la anchura y la altura del contenido de un elemento caja sin márgenes ni bordes. Su sintaxis es:

**width: auto / valor / porcentaje;**

y

**height: auto /valor / porcentaje;**

A tener en cuenta:

* El valor **auto** ajusta automáticamente la dimensión del contenido.
* El valor de longitud especifica un valor fijo.
* El porcentaje especifica un valor relativo.

***Veámoslo en la práctica realizando la práctica 9.3.***

MÁRGENES EXTERNOS

La propiedad **margin** permite definir el margen exterior del elemento caja. Su sintaxis es:

**margin: auto / valor / porcentaje;** o,

**margin-top: auto / valor / porcentaje;** o,

**margin-right: auto / valor / porcentaje;** o,

**margin-bottom: auto / valor / porcentaje;**  o,

**magin-left: auto / valor / porcentaje;**

A tener en cuenta:

* El valor **auto** deja que el navegador ajuste automáticamente los márgenes externos.
* Un valor de longitud especifica un valor fijo para los márgenes exteriores.
* Un porcentaje define la longitud respecto a un elemento padre.
* **Margin** especifica los cuatro márgenes pero podemos especificar el margen exterior de un solo lado del contenedor usando respectivamente margin-top, margin-right, margin-bottom y margin-left.
* No obstante también utilizando **margin** podemos especificar de forma abreviada el margen exterior de uno o varios lados del contenedor de la siguiente forma:
  + **margin: 10px;** Si se indica una única cantidad se aplica a los cuatro lados.
  + **margin: 10px 10px;** Si se indican dos cantidades la primera se aplica a los lados superior e inferior y la segunda a los márgenes laterales derecho e izquierdo.
  + **margin: 10px 10px 10px;** Si se indican tres valores el primero se aplica al lado superior, el segundo a los márgenes laterales y el tercero al lado inferior.
  + **margin:** **10px 10px 10px 10px;** Indicando cuatro cantidades se aplican a los cuatro márgenes exteriores en el sentido de las agujas del reloj partiendo del lado superior.
* Se pueden utilizar cantidades negativas.

***¡Practiquémoslo realizando la práctica 9.4!***

¿ANCHURA FIJA O VARIABLE PARA NUESTRA WEB?

Una de las primeras cosas que hay que decidir a la hora de iniciar el desarrollo de una página web es decidir que anchura va a tener. Existen dos maneras de diseño atendiendo a la anchura de la web:

* **Web de Anchura fija**. En estas páginas todos los elementos se incluyen dentro de un contenedor que tiene un ancho exacto. En nuestro caso vamos a utilizar como contenedor la misma etiqueta <**body**> y como ancho 960 píxeles. Este diseño permite ubicar con precisión todos los elementos de la página. Es la estructura que emplearemos
* **Web de anchura variable**. Están pensadas para adaptarse a la anchura de la ventana del navegador en cualquier momento. Es muy útil para diseños web en los que se prevé que el usuario acceda a la web por dispositivos con resoluciones de pantalla variadas: tablets, dispositivos móviles, etc, etc.

Como hemos comentado en esta práctica nosotros emplearemos una estructura de anchura fija de 960 píxeles.

GROSOR DEL BORDE

La propiedad **border-width** permite definir el grosor de los cuatro bordes simultánea o individualmente. Su sintaxis es:

**border-width: valor / thin / medium / thick;** o,

**border-top-width: valor / thin / medium / thick;** o,

**border-right-width: valor / thin / medium / thick;** o,

**border-bottom-width: valor / thin / medium / thick;** o,

**border-left-width: valor / thin / medium / thick;**

A tener en cuenta:

* El grosor no se muestra si no se ha definido **border-style**.
* El valor **thin** indica un borde fino.
* Un valor **medium** indica un borde medio.
* El valor **thick** ordenada un borde grueso.
* La interpretación de **thin**, **médium** y **thick** puede variar de un navegador a otro.
* **Border-width** especifica los cuatro bordes pero podemos especificar el borde de un solo lado como ya sabemos por el caso de los márgenes exteriores.
* También podemos especificar de forma abreviada el borde de uno o varios lados del contenedor de la siguiente forma:
  + Si se indica un grosor se aplica a los cuatro bordes.
  + Si se indican dos grosores el primero se aplica a los bordes superior e inferior y el segundo a los bordes laterales derecho e izquierdo.
  + Si se indican tres grosores el primero se aplica al borde superior, el segundo a los bordes laterales y el tercero al borde inferior.
  + Indicando cuatro grosores se aplican a los cuatro bordes en el sentido de las agujas del reloj partiendo del lado superior.

***¡Practiquémoslo realizando la práctica 9.5.!***

ESTILO DEL BORDE

La propiedad **border-style** permite definir el grosor de los cuatro bordes simultánea o individualmente. Su sintaxis es:

**border-style: solid / dashed / dotted / double / groove / ridge / inset / outset / hidden / none;**

o,

**border-top-style: “** o,

**border-right-style:**  **“** o,

**border-bottom-style: “** o,

**border-left-style: “**

A tener en cuenta:

* El el estilo del borde no se muestra si no se ha definido **border-width**.
* Los diferentes estilos del borde son:

|  |  |  |
| --- | --- | --- |
| Solid | Un trazo liso | **solid** |
| Dashed | Guiones | **dashed** |
| Dotted | Pequeños puntos | **dotted** |
| Double | Trazos lisos dobles | **double** |
| Groove | Efecto 3D de relieve incrustado en la página. | **groove** |
| Ridge | Efecto 3D de relieve saliendo de la página. | **ridge** |
| Inset | Bordes entrantes, incrustados en la página | **inset** |
| Outset | Bordes salientes, biselados en la página | **outset** |
| Hidden | Sin borde. Influye sobre el borde adyacente. | **hidden** |
| None | Sin borde. | **none** |

* **Border-style** especifica el estilo de los cuatro bordes pero podemos especificar el de un solo lado como ya sabemos.
* También podemos especificar de forma abreviada el estilo del borde de uno o varios lados del contenedor de la siguiente forma:
  + Si se indica un estilo se aplica a los cuatro bordes.
  + Si se indican dos estilos el primero se aplica a los bordes superior e inferior y el segundo a los bordes laterales derecho e izquierdo.
  + Si se indican tres estilos el primero se aplica al borde superior, el segundo a los bordes laterales y el tercero al borde inferior.
  + Indicando cuatro estilos se aplican a los cuatro bordes en el sentido de las agujas del reloj partiendo del lado superior.

***¡Veamos su funcionamiento realizando la práctica 9.6!***

COLOR DEL BORDE

La propiedad **border-color** permite definir el color de los cuatro bordes de un contenedor simultánea o individualmente. Su sintaxis es:

**border-color: nombre / notación / transparent;** o,

**border-top-color: nombre / notación / transparent;** o,

**border-right-color: nombre / notación / transparent;** o,

**border-bottom-color: nombre / notación / transparent;** o,

**border-left-color: nombre / notación / transparent;**

A tener en cuenta:

* El color no se puede aplicar si no se ha definido **border-style** y **border-width**.
* El valor indicado en **border-color** define el color del borde de los cuatro lados del contenedor.
* Podemos especificar el color del borde de un solo lado con las propiedades **border-top-color, border-right-color, border-bottom-color y border-left-color**.
* También podemos especificar de forma abreviada el borde de uno o varios lados del contenedor con la propiedad **border-color** de la siguiente forma:
  + Si se indica un solo color se aplicará a los cuatro bordes.
  + Si se indican dos colores el primero se aplica a los bordes superior e inferior y el segundo a los bordes laterales derecho e izquierdo.
  + Si se indican tres colores el primero se aplica al borde superior, el segundo a los bordes laterales y el tercero al borde inferior.
  + Indicando cuatro colores se aplican a los cuatro bordes en el sentido de las agujas del reloj partiendo del lado superior.

ESCRITURA ABREVIADA DEL BORDE

Podemos escribir todos los valores de los bordes de caja de forma abreviada utilizando la propiedad “**border**”. Un ejemplo de su sintaxis es:

**border: red double 5px;**

A tener en cuenta:

* Esta forma solo vale en el caso de que los cuatro bordes sean idénticos.
* No importa el orden de los diferentes valores. Eso sí, tiene que ir separadas por espacio.
* Para asegurar la compatibilidad óptima entre los distintos navegadores, se recomienda informar las tres propiedades.

Semejante a “**border**” existe también la propiedad “**outline**” que genera un borde exterior pero no ocupa espacio y no es necesariamente rectangular. Además “**outline-offset**” permite establecer una distancia entre el borde interior y el exterior.

***¡Veámoslo en la práctica 9.7.!***

FORMATEANDO EL ELEMENTO <HEADER>

 Vamos a abordar ahora el formato del elemento <**header**>. Dejaremos la foto a la izquierda y pondremos el título de la página centrado a su izquierda con un fondo degradado rojo que recuerde a España. En la siguiente imagen podemos hacernos una idea del resultado final.

SELECTORES JERÁRQUICOS

A parte de los selectores que ya conocemos: de clase, de etiqueta, de identificador, existen otros muchos que iremos explicando a lo largo del curso. Ahora vamos a trabajar con selectores jerárquicos.

Los selectores jerárquicos son aquellos que aprovechan la circunstancia de que unos elementos **HTML** están anidados dentro de otros. Podríamos decir entonces, que unos elementos son hijos de otros. Así en el código que acabamos de escribir la etiqueta <**h1**> es hija de <**div**> pues <**h1**> está dentro de <**div**>. Por la misma razón, <**div**> sería el elemento padre de <**h1**>.

Siguiendo con este ejemplo podemos considerar tres selectores jerárquicos:

|  |  |  |
| --- | --- | --- |
| Div h1 | Selector descendente | Aplicaríamos los estilos a todos los <**h1**> que estén dentro de <**div**>. |
| Div > h1 | Selector hijos | Aplicaríamos los estilos a todos los <**h1**> que sean hijos directos de <**div**>. |
| Div + h1 | Selector hermano adyacente | Aplicaríamos los estilos a cualquier elemento <**h1**> precedido inmediatamente del elemento <**div**> |

***Para ver el funcionamiento de todo esto realice ahora la práctica 9.8.***

FLOTAR

La propiedad **float** permite retirar un elemento de tipo caja del flujo normal para situarlo lo más a la derecha o lo más a la izquierda posible dentro de su elemento padre. Su sintaxis es:

**float: right / left / none;**

A tener en cuenta:

* El valor **right** alinea a la derecha el elemento indicado, empujando a los demás elementos a la izquierda.
* El valor **left** alinea a la izquierda el elemento indicado, empujando a los demás elementos a la derecha.
* El valor **none** deja la gestión al navegador.

***¡Veamos el resultado realizando la práctica 9.9.!***

MÁRGENES INTERNOS

Tenemos que centrar el texto del título verticalmente. Para ello podríamos aplicar la propiedad “**vertical-aling**” que ya vimos en páginas anteriores pero desgraciadamente esta propiedad no funciona en todos los contenedores por lo que, en este caso, tendremos que utilizar un pequeño truco aprovechándonos de los márgenes internos.

La propiedad **padding** permite definir el margen interior del elemento caja. Su sintaxis es:

**padding: auto / valor / porcentaje;** o,

**padding-top: auto / valor / porcentaje;** o,

**padding-right: auto / valor / porcentaje;** o,

**padding-bottom: auto / valor / porcentaje;** o,

**padding-left: auto / valor / porcentaje;**

A tener en cuenta:

* El valor **auto** deja que el navegador ajuste automáticamente los márgenes internos.
* Un valor de longitud especifica un valor fijo para los márgenes interiores.
* Un porcentaje define la longitud respecto a un elemento padre.
* **Padding** especifica los cuatro márgenes pero podemos especificar el margen interior de un solo lado del contenedor usando respectivamente padding-top, padding-right, padding-bottom y padding-left.
* No obstante también utilizando **padding** podemos especificar de forma abreviada el margen interior de uno o varios lados del contenedor de la siguiente forma:
  + **padding: 10px;** Si se indica una única cantidad se aplica a los cuatro lados.
  + **padding: 10px 10px;** Si se indican dos cantidades la primera se aplica a los lados superior e inferior y la segunda a los márgenes laterales derecho e izquierdo.
  + **padding: 10px 10px 10px;** Si se indican tres valores el primero se aplica al lado superior, el segundo a los márgenes laterales y el tercero al lado inferior.
  + **padding:** **10px 10px 10px 10px;** Indicando cuatro cantidades se aplican a los cuatro márgenes interiores en el sentido de las agujas del reloj partiendo del lado superior.

***¡Apliquémoslo realizando la práctica 9.10.!***

BORDES REDONDEADOS

La propiedad **border-radius** permite formatear contenedores con bordes redondeados. Su sintaxis es:

**border-radius: x y;**

A tener en cuenta:

* **X** e **Y** pueden ser un valor o un porcentaje.
* Los valores de **X** e **Y** determinan los radios horizontal y vertical de un cuarto de elipse, lo cual provocará la curvatura de la esquina.
* Es posible indicar un único valor. En ese caso el valor de **X** será igual al valor de **Y**.
* Como en otras propiedades **border-radius** especifica las cuatro esquinas pero podemos especificar el borde redondeado de una sola esquina del contenedor usando respectivamente border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius y border-top-left-radius.
* También es posible definir el borde redondeado para cada esquina a partir de una escritura abreviada que se lee, como de costumbre, según el sentido de las agujas del reloj comenzando por la parte superior.

***Para poner los bordes redondeados de la parte derecha de nuestro contenedor <div> realice la práctica 9.11.***

Con referencia a los bordes simplemente dejaremos aquí anotado que existe la propiedad **border-image** que permite asignar una imagen como borde de un contenedor.

EFECTO SOMBREADO SOBRE UN ELEMENTO CAJA

La propiedad **box-shadow** añade sombra a un elemento de caja. Su sintaxis es:

**box-shadow: x y z color inset;**

A tener en cuenta:

* **X** es el margen de sombra hacia la derecha.
* **Y** es el margen de sombra hacia abajo.
* **Z** es la intensidad de degradado o desenfocado. (Opcional con valor por defecto 0).
* **Color** es el color de la sombra.
* **Inset.** Indica que la sombra sea interna.
* Los parámetros **X** e **Y** admiten valores negativos. En este caso la sombra se desplaza hacia la izquierda y hacia arriba.
* Es importante tener en cuenta que la sombra se aplica sobre la caja del elemento, no sobre su borde por lo que la sombra no afecta al tamaño de la caja.

***¡Vayamos a la práctica y pongamos sombra al contenedor <div> del <header> realizando la práctica 9.12!***

FORMATEANDO EL ELEMENTO <NAV>

 En nuestra zona de navegación utilizaremos un fondo negro sobre el colocaremos los enlaces tal y como se ve en la siguiente imagen.

***Para ello realice la práctica 9.13.***

DESPEJAR

La propiedad “**clear**” anula el efecto introducido por la propiedad “**float**”. Su sintaxis es:

**clear: right / left / both /none;**

A tener en cuenta:

* **Right** anula los elementos flotantes a la derecha.
* **Left** anula los elementos flotantes a la izquierda.
* **both** anula los elementos flotantes a ambos lados.
* **None** anula los elementos flotantes.

***¡Apliquémoslo realizando la práctica 9.14.!***

VISUALIZAR ELEMENTOS EN BLOQUE O EN LÍNEA

Vamos a dar formato a los elementos <**li**> correspondientes a los enlaces de <**nav**>.

Los cuatro enlaces que tenemos están uno debajo de otro porque son elementos en bloque que como ya sabemos ocupan todo el espacio horizontalmente. Para que aparezcan en una sola línea uno detrás de otro debemos indicar que se comporten como elementos en línea. Esto lo podemos hacer con la propiedad “**display**”. Su sintaxis es:

**display: block / inline / none;**

A tener en cuenta:

* **Block** redefine un elemento en línea como un elemento en bloque.
* **Inline** redefine un elemento en bloque como un elemento en línea.
* **None** define un elemento que no se va a mostrar y por lo tanto se retira del documento y de la visualización de la página. Es como si no existiera dicho elemento.
* Esta propiedad es muy rica en posibilidades de presentación y se utiliza a menudo en hojas de estilo **CSS** complejas.

Si observa la hoja de estilos de reset verá que muchos elementos en bloque se les pone en línea o viceversa. ¡Compruébelo abriendo la hoja de estilos “**reset.css**”!

***Para efectuar los cambios de formato en los elementos <li> realice la práctica 9.15.***

FORMATEANDO EL ELEMENTO <SECTION>

 Vamos a maquetar el cuerpo principal de nuestra web que está indicado bajo la etiqueta <**section**>. Muchas de las propiedades de **CSS** que manejaremos ya las hemos aprendido pero veremos alguna nueva. Al terminar este proceso la web tendrá un aspecto parecido al de la siguiente imagen:

***Abordemos el formateo realizando la práctica 9.16.***

FORMATEANDO LOS ELEMENTOS <ARTICLE>

Los elementos <**article**> los vamos a colocar horizontalmente como si se tratara de celdas de una tabla. Para ello vamos a utilizar un valor de la propiedad “**display**” que no comentamos en su momento y que estudiaremos más adelante.

***Para ello realice ahora la práctica 9.17.***

DAR FORMATO AL ELEMENTO <ASIDE>

Aunque el elemento <**aside**> puede situarse en cualquier parte de una página web, en la mayoría de los casos se utiliza para las barras laterales, como es nuestro caso.

***Realice la práctica 9.18.***

FORMATEANDO EL ELEMENTO <FOOTER>

Con todo lo que hemos practicado hasta ahora dar formato al pie de página va a resultarnos muy sencillo. **Realice para ello la práctica 9.19.**

Con esta última práctica hemos terminado de dar formato a nuestra página web. Aún podemos realizar algunas mejoras y cambios que iremos viendo en lo que queda de curso.

***Por ahora para terminar de dominar esta parte del tema realice la primera ejercitación del capítulo 9.***

POSICIONAMIENTO DE ELEMENTOS HTML CON CSS

Las hojas de estilo **CSS** proporcionan las herramientas adecuadas para posicionar con una precisión de pixel, cualquier elemento de un documento **HTML**. Aplicando la propiedad “**position**”, un elemento puede posicionarse de cuatro formas:

A.- **Posicionamiento estático**: Es el posicionamiento normal del elemento. La sintaxis es:

**position: static;**

A tener en cuenta:

* El diseñador no tiene el control.
* El elemento no puede posicionarse ni reposicionarse.
* La visibilidad, propiedad que veremos más adelante, no puede modificarse.

B.- **Posicionamiento relativo**: Es el posicionamiento de un elemento respecto a su posición normal o estática. Su sintaxis es:

**position: relative; left: valor ó %;**

**Top: valor ó %;**

**Right: valor ó %;**

**Bottom: valor ó %;**

A tener en cuenta:

* El elemento permanece en el flujo de los datos aunque está, en cierta medida, descentrado respecto a su posición normal.
* La posición se define mediante las coordenadas (**x,y**) donde:
  + **X** es la distancia respecto al borde izquierdo del elemento padre o de la ventana del navegador (eje horizontal). De esta manera:
    - “**left**” determina la distancia entre la parte izquierda del elemento y el borde izquierdo de la página.
    - “**right**” determina la distancia entre la parte derecha del elemento y el borde derecho de la página.
  + **Y** es la distancia respecto al borde superior del elemento padre o de la ventana del navegador (eje vertical). De este modo:
    - “**top**” determina la distancia entre el borde superior del elemento y el borde superior de la página.
    - “**bottom**” determina la distancia entre el borde inferior del elemento y el borde inferior de la página.
* Basta con indicar una sola especificación en el eje horizontal (“**left**” o “**right**”) y otra sobre el eje vertical (“**top**” o “**bottom**”).

C.- **Posicionamiento absoluto**: El posicionamiento absoluto crea un elemento independiente del resto del documento. Su sintaxis es:

**position: absolute; left: valor ó %;**

**Top: valor ó %;**

**Right: valor ó %;**

**Bottom: valor ó %;**

A tener en cuenta:

* El elemento se retira del flujo normal y se posiciona en el lugar exacto definido por el diseñador.
* La posición se define mediante las coordenadas (**x,y**) donde:
  + **X** es la distancia respecto al borde izquierdo del elemento padre o de la ventana del navegador (eje horizontal). De esta manera:
    - “**left**” determina la distancia entre la parte izquierda del elemento y el borde izquierdo de la página.
    - “**right**” determina la distancia entre la parte derecha del elemento y el borde derecho de la página.
  + **Y** es la distancia respecto al borde superior del elemento padre o de la ventana del navegador (eje vertical). De este modo:
    - “**top**” determina la distancia entre el borde superior del elemento y el borde superior de la página.
    - “**bottom**” determina la distancia entre el borde inferior del elemento y el borde inferior de la página.
* Basta con indicar una sola especificación en el eje horizontal (“**left**” o “**right**”) y otra sobre el eje vertical (“**top**” o “**bottom**”).
* El posicionamiento absoluto tiene el riesgo de no mostrar correctamente la página en resoluciones de pantalla diferentes a la utilizada en tiempo de diseño puesto que los elementos que se hayan posicionado mediante esta propiedad pueden superponerse sobre los elementos del flujo normal de la página.
* Conviene recordar que los navegadores agregan por defecto un margen al cuerpo de la página **HTML** y que este margen por defecto varía de un navegador a otro. Por ello es prudente, e caso de usar posicionamiento absoluto, especificar los márgenes de la etiqueta <**body**>.

D.- **Posicionamiento fijo**: El posicionamiento fijo crea un elemento independiente del que se puede definir su posición exacta en relación a la ventana del navegador. Su sintaxis es:

**position: fixed;**

***Para ver el funcionamiento de esta propiedad realice la práctica 9.20.***

SUPERPOSICIÓN

La propiedad “**z-index**” permite posicionar los elementos unos por encima o por debajo de otros: Su sintaxis es:

**z-index: nº entero positivo;**

A tener en cuenta:

* Si un elemento tiene la propiedad “**z-index: 2;**” aparecerá por delante de otro elemento que tenga la propiedad “**z-index: 1;**”.
* Esta propiedad solo funciona con un posicionamiento absoluto de los elementos.

***Para ilustrar esta propiedad realice la práctica 9.21.***

DESBORDAMIENTO

La propiedad “**overflow**” determina lo que debe hacer el navegador cuando un elemento es más grande que el elemento padre que lo contiene. Su sintaxis es:

**overflow: hidden / scroll / visible / auto;**

A tener en cuenta:

* Con el valor “**hidden**”, la parte que desborda se oculta, sin posibilidad de verla ni acceder a ella.
* Con el valor “**scroll**”, la parte que desborda se oculta, pero aparecen barras de desplazamiento para acceder a ella.
* Con el valor “**visible**”, la parte que desborda se muestra, ignorando las especificaciones del elemento padre que lo contiene.
* Con el valor “**auto**”, se deja la gestión al navegador.

***¡Veamos un ejemplo realizando la práctica 9.22.!***

RECORTE

La propiedad “**clip**” determina la parte visible del elemento, generalmente una imagen. Su sintaxis es:

**clip: rect (sup\_izquierda sup\_derecha inf\_derecha inf\_izquierda / auto;**

A tener en cuenta:

* Las coordenadas del rectángulo recortado vienen dadas por las esquinas superior izquierda, superior derecha, inferior derecha e inferior izquierda respecto de la imagen inicial.
* Se recomienda incluir el elemento que se quiere recortar dentro de una etiqueta <**div**> o <**span**>.
* Solamente funciona con un posicionamiento absoluto del elemento.

***Para ver su funcionamiento realice la práctica 9.23.***

VISIBILIDAD

La propiedad “**visibility**” determina si un elemento se está visible u oculto. Su sintaxis es:

**visibility: visible / hidden;**

A tener en cuenta:

* El valor “**visible**” muestra el elemento.
* El valor “**hidden**” oculta el elemento.
* Si se establece el valor “**hidden**” el elemento ocupa todavía su hueco en el documento y en consecuencia se muestra un rectángulo en blanco donde estuviera situado el elemento. Es decir se conserva el formato de página a pesar de la ausencia, aparente, del elemento, de tal manera que los demás elementos de la página conservan su posición relativa inicial.
* No confundir el valor “**hidden**” con la propiedad “**display: none;**” que elimina por completo el elemento de la página y no se reserva sitio alguno.

***Para ver el efecto que produce esta propiedad realice la práctica 9.24.***

CURSOR DEL RATÓN

La propiedad “**cursor**” permite modificar el cursor del ratón. Su sintaxis es:

**cursor: palabra clave;**

Las distintas palabras clave son:

|  |  |
| --- | --- |
| URL | Especifica un archivo donde se encuentra la imagen que se desea usar como cursor. El archivo de imagen especificado en la URL debe tener el formato .cur o .ani. **Nota:** Se pueden indicar varios cursores separados por comas. En este caso especifique siempre un cursor genérico al final de la lista por si ninguno de los cursores definidos por URL se pueden utilizar. |
| auto | El navegador establece el cursor según el contexto. Es la opción predeterminada. |
| crosshair | Cursor en forma de cruz. |
| default | Cursor por defecto del sistema operativo. |
| e-resize | Cursor apuntando hacia el este. |
| help | El cursor indica que hay ayuda disponible. A menudo se muestra con un signo de interrogación. |
| move | El cursor indica un objeto que se puede trasladar. |
| n-resize | Cursor apuntando hacia el norte. |
| ne-resize | Cursor apuntando hacia el noreste. |
| nw-resize | Cursor apuntando hacia el noroeste. |
| pointer | El cursor presenta un dedo que indica un enlace. |
| progress | El cursor indica que el programa está ocupado, en curso, pero se puede seguir trabajando. Normalmente se muestra una flecha con un reloj de arena. |
| s-resize | Cursor apuntando hacia el sur. |
| se-resize | Cursor apuntando hacia el sureste. |
| sw-resize | Cursor apuntando hacia el suroeste. |
| text | El cursor indica que es posible escribir o seleccionar texto. |
| w-resize | Cursor apuntando hacia el oeste. |
| wait | El cursor indica que el programa está ocupado y debemos esperar. Suele representarse con un reloj de arena o un círculo dando vueltas. |
| not-allowed | El cursor muestra una señal de prohibido. |
| no-drop | El cursor tiene forma de mano con un dedo desplegado y una señal de prohibido. |
| col-resize | Cursor compuesto por dos trazos verticales con una flecha a cada lado. |
| row-resize | Cursor compuesto por trazos horizontales. |

***Para ver el funcionamiento de esta propiedad realice la práctica 9.25.***

COLUMNAS PERIODÍSTICAS

La propiedad “**column-count**” permite presentar contenido en columnas de estilo periodístico. Su sintaxis es:

**-moz-column-count: valor entero que determina el nº de columnas.**

**-moz-column-width: valor del ancho de cada columna. (Es opcional).**

**-moz-column-gap: valor del espacio entre columnas.**

**-moz-column-rule: Define el borde entre las columnas.**

***Para ver su funcionamiento realice la práctica 9.26.***

CAPÍTULO 10.- PSEUDOCLASES EN CSS

¿QUÉ ES UNA PSEUDOCLASE?

Hasta el momento, para declarar los estilos en las hojas de estilo, hemos utilizado sobre todo selectores de etiqueta y en pocas ocasiones selectores de clase, (si es necesario recuerde estos conceptos explicados en el capítulo 4), pero **CSS** nos permite aplicar estilos a elementos de un documento **HTML** que no podemos especificar a través de una etiqueta o de una clase como, por ejemplo, un determinado enlace o la primera letra de un párrafo. Para referenciar esos elementos más específicos, que luego indicaremos, utilizamos pseudoclases.

Una pseudoclase permite tener en cuenta diferentes condiciones o eventos a la hora de definir un estilo para una etiqueta **HTML**.

La sintaxis de las psuedoclases es:

**Selector:pseudoclases {**

**…**

**declaraciones de estilo**

**…**

**}**

Las pseudoclases también se pueden utilizar con las clases. En ese caso su sintaxis sería:

**Selector.clase:pseudoclase {**

**…**

**declaraciones de estilo**

**…**

**}**

¡Veamos y practiquemos con las pseudoclases!

PSEUDOCLASES DE ENLACE

Los enlaces que indicamos con la etiqueta <**a**> pueden tener diferentes estados y podemos asignar estilos diferentes, dependiendo del estado en que se encuentre, utilizando las siguientes pseudoclases:

**A.-** **a:link**. Permite definir la apariencia de un enlace que aún no se ha seleccionado o visitado.

**B.-** **a:visited**. Permite definir la apariencia de enlaces que dirigen a páginas que el usuario ya ha visitado. ***¡Veamos un ejemplo práctico realizando la práctica 10.1!***

**C.-** **a:hover**. Permite definir la apariencia de un enlace en el momento que se pasa el ratón sobre él. ***¡Veamos cómo funciona realizando la práctica 10.2!***

**D.-** **a:active**. Permite definir la apariencia de un enlace en el momento preciso en que se hace clic sobre él. ***Un ejemplo de su uso podría ser como en la práctica 10.3…***

A tener en cuenta:

* La pseudoclase **:active** se utiliza, generalmente, de forma conjunta con **:visited**, **:hover**, y **:link**.
* Para obtener un resultado correcto es indispensable enunciar las apariencias para los enlaces en el orden siguiente: **:link**, **:visited**, **:hover** y **:active**.
* La pseudoclase **:hover** puede utilizarse con otras etiquetas **HTML** además de con los enlaces como ya hemos visto al utilizarlo con la etiqueta <**li**>.

Llegados a este punto, es necesario hacer una reflexión sobre el subrayado de los enlaces. Hasta ahora, en todas nuestras páginas de ejemplos y ejercicios hemos eliminado todos los subrayados de enlaces con la propiedad “**text-decoration: none**”. Debemos tener cuidado con esta práctica ya que los usuarios de internet están acostumbrados al subrayado azul de los enlaces de las páginas web y saben que pueden hacer clic en ellos. Si quitamos dicho subrayado existe la posibilidad de confundir a los usuarios y disminuir la usabilidad de la web. El uso de las pseudoclases de enlace ayudarán en este cometido.

***Realice ahora la primera parte de la ejercitación del tema 10.***

PSEUDOCLASES DE PÁRRAFO

**A.-** **:first-letter**. Permite definir la apariencia de la primera (y solo de la primera) letra de un párrafo. Se logra un efecto de letra capital. ***¡Veamos un ejemplo realizando la práctica 10.4!***

**B.-** **:first-line**. Permite definir la apariencia de la primera (y solo de la primera) línea de un párrafo. ***¡Vayamos al ejemplo realizando la práctica 10.5!***

PSEUDOCLASES PARA INSERTAR CONTENIDO

**A.-** **:before**. Asociada a la propiedad “**content**”, permite insertar un contenido antes de un elemento. ***¡Vayamos a la práctica realizando la 10.6!***

**B.-** **:after**. Asociada a la propiedad “**content**”, permite insertar un contenido después de un elemento. ***¿Lo probamos realizando la práctica 10.7?***

PSEUDOCLASES DE SELECCIÓN

Además de las pseudoclases que hemos utilizado en los pasos anteriores existen otras pseudoclases que nos permiten indicar elementos **HTML** dependiendo de la posición que ocupen dentro de la página web, padres o hijos, y en relación con sus elementos hermanos. Recordar el concepto de “**herencia**” que aprendimos en temas anteriores nos ayudará a entender estas pseudoclases. Las llamadas pseudoclases de selección son:

|  |  |
| --- | --- |
| **:root** | Representa el elemento raíz o el elemento en el nivel más alto del documento. Por ejemplo, en **HTML5**, la etiqueta <**html**>. |
| **:empty** | Corresponde a los elementos vacíos y que no tienen hijos. |
| **:only-child** | Devuelve el hijo único. Este elemento no tiene elementos hermanos. |
| **:first-child** | Selecciona el primer elemento hijo. |
| **:last-child** | Selecciona el último elemento hijo. |
| **:nth-child(n)** | Designa e enésimo elemento hijo, donde n es un número o bien las palabras clave “**even**” (pares) y “**odd**” (impares). |
| **:nth-last-child(n)** | Selecciona el enésimo hijo a partir del último elemento. |
| **:only-of-type** | Devuelve el único elemento de este tipo. |
| **:first-of-type** | Representa el primer elemento de este tipo. |
| **:last-of-type** | Representa el último elemento de este tipo. |
| **:nth-of-type(n)** | Designa al enésimo elemento de este tipo, donde n es un número o bien las palabras claves “**even**” (pares) y “**odd**” (impares). |
| **:nth-last-of-type(n)** | Selecciona el enésimo elemento de este tipo partiendo del último elemento. |
| **not()** | Selecciona todos los elementos excepto el indicado entre paréntesis. |

***¡Veamos algunos ejemplos en la práctica 10.8!***

SELECTORES DE ATRIBUTOS

Ya conocemos que las etiquetas en **HTML** pueden tener atributos tales como: “**alt**”, “**title**”, “**href**”, etc… Pues bien los llamados “**selectores de atributo**”, aprovechan dichos atributos para convertirlos en selectores **CSS** y así poder aplicar un estilo a los elementos que tienen dichos atributos. Esto nos librará de poner clases e identificadores a muchas etiquetas **HTML**.

Los selectores de atributos son:

|  |  |
| --- | --- |
| **elemento[atributo]** | Los estilos se aplicarán a todas las etiquetas que tengan ese atributo. |
| **elemento[atributo=”valor”]** | Los estilos se aplicarán a todas las etiquetas que tengan ese atributo con ese “**valor**”. |
| **elemento[atributo~=”valor”]** | Los estilos se aplicarán a las etiquetas que tengan ese atributo y que uno de sus valores separados por espacios sea el “**valor**”. |
| **elemento[atributo|=”valor”]** | Selecciona los elementos que tienen establecido un atributo llamado “**atributo**” que contenga una lista de valores separados por guiones, pero que comienza por “**valor**”. |
| **elemento[atributo^=”valor”]** | Los estilos se aplicarán a todos los elementos que tengan ese atributo y cuyo valor comienza por la cadena de texto indicada en “**valor**”. |
| **elemento[atributo$=”valor”]** | Los estilos se aplicarán a todos los elementos que tengan ese atributo y cuyo valor termina por la cadena de texto indicada en “**valor**”. |
| **elemento[atributo\*=”valor”]** | Los estilos se aplicarán a todos los elementos que disponen de ese atributo y cuyo valor contiene la cadena de texto indicada en “**valor**”. |

***Para ver un ejemplo de este tipo de selectores realice la práctica 10.9***

***A continuación realice la segunda parte de la ejercitación del capítulo 10.***

CAPÍTULO 11.- FORMULARIOS CON HTML

Los formularios ocupan un lugar relevante en el diseño y explotación de una aplicación o un sitio web. Nos sirven para recibir de vuelta la información que proviene directamente de los visitantes de nuestra web.

**HTML5** ha supuesto enormes mejoras en el tratamiento de los formularios convirtiendo en sencillas tareas que antes necesitaban de lenguajes de programación como **JavaScript**.

El propósito último de los formularios es el procesamiento automático de los datos escritos por los usuarios. Para esta tarea será preciso usar lenguajes de programación de gestión de bases de datos como **PHP** o **MySQL** que estudiará más adelante en nuestro curso de “**Creación y Edición de Páginas Web**”.

FORMULARIOS: DECLARAR UN FORMULARIO

La etiqueta <**form**> no muestra nada en la web. Tiene como única función indicar al navegador que a continuación se va a mostrar un formulario. Engloba el resto de etiquetas y elementos que constituyen el formulario.

Los atributos de la etiqueta <**form**> son:

* **Name**. Asigna un nombre al formulario.
* **Action**. Le indica al navegador la acción que vamos a realizar con los datos introducidos en el formulario. Para el **W3C** la presencia de este atributo es obligatoria. Pueden ser:
  + Llamar a un programa de procesamiento de datos como veremos en el curso del lenguaje **PHP**. Por ejemplo la etiqueta podría ser como: **<form action=”http://servidor/datos.php”>**.
  + Llamar al programa cliente de correo instalado en el ordenador del usuario para enviar los datos recogidos en el formulario a un correo electrónico. En este caso la etiqueta quedaría como: **<form action=”mailto:info@academiafrayluis.es”>**.
  + Procesar internamente los datos del formulario mediante **JavaScript**. En este caso el atributo “**action**” quedaría vacío como: **<form action=””>**.
* **Enctype**. Especifica el formato informático, (mime type), en que se transmitirán los datos del formulario.
* **Method**. Especifica el método de transmisión de los datos recogidos en el formulario. Puede ser **GET** o **POST**. En la práctica se utiliza más este último.

***¡Vayamos a la práctica realizando la 11.1!***

FORMULARIOS: CAMPO DE TEXTO

La etiqueta <**input**> genera un campo de texto que permite introducir datos tanto alfabéticos como numéricos. No tiene etiqueta de cierre aunque se acepta la sintaxis de **XHTML** <**input /**>.

Los atributos de la etiqueta <**input**> son:

* **Type**. Especifica el tipo de dato que se va a almacenar en el campo de texto. Este atributo tiene bastantes valores que estudiaremos en este capítulo.
* **Name**. Asigna un nombre único para este elemento.
* **Size**. Indica el número de caracteres visibles del campo de texto y por tanto la longitud del campo. Su valor por defecto es 20 pero, aunque no se vean, el usuario puede introducir tantos caracteres como quiera.
* **Maxlength**. Determina el número máximo de caracteres que el usuario puede introducir en el campo.
* **Value**. Indica el valor por defecto del campo texto. Aparece como valor del campo de texto cuando se carga la página.
* **Readonly**. Indica que el usuario no puede modificar el valor por defecto definido en el atributo anterior.
* **Placeholder**. Permite incluir una sugerencia acerca del valor que se debe introducir en el campo. Esta sugerencia aparece de color gris en el campo tras cargar la página y desaparece cuando el usuario sitúa el foco en el elemento afectado.
* **Autofocus**. Sitúa el foco sobre el campo una vez cargada la página.
* **Required**. Convierte al campo en obligatorio para poder enviar los datos recogidos en el formulario. El navegador muestra un bocadillo indicando que el campo se debe rellenar.
* **Pattern**. Se utiliza junto con JavaScript.
* **Height**. Indica la altura del campo en píxeles o porcentaje. Se puede reemplazar por la propiedad **CSS** que ya conocemos.
* **Width**. Indica la anchura del campo en píxeles o porcentaje. Se puede reemplazar por la propiedad **CSS** que ya conocemos.
* **Autocomplete**. Especifica si se activa o no la función autocompletar en el campo. Cuando esta función está activa el navegador recuerda lo introducido en el mismo campo anteriormente y lo ofrece mediante un menú desplegable ahorrando al usuario el tener que escribirlo de nuevo.

FORMULARIOS: ETIQUETADO DE LOS CAMPOS.

La etiqueta <**label**> asocia a un campo de un formulario un título. El funcionamiento de esta etiqueta es el siguiente:

**A.-** En primer lugar el texto asignado como título de un campo de formulario debe situarse entre las etiquetas <**label**> … <**/label**>. Por ejemplo:

**<label>Nombre</label>**

**B.-** Luego hay que asociar esta etiqueta <**label**> a su correspondiente control. Para ello el campo del formulario debe definirse mediante un identificador de tipo “**id**”.

**<label>Nombre</label>**

**<input type=”text” id=”nombre1”>**

**C.-** El atributo “**for=”…”**” dentro de la etiqueta <**label**> permite asociar directamente la etiqueta al campo haciendo referencia a ese identificador.

**<label for=”nombre1”>Nombre</label>**

**<input type=”text” id=”nombre1”>**

¡Vamos a crear nuestro primer campo del formulario! ***Para ello realice la práctica 11.2.***

CAMPO DE TEXTO PARA DIRECCION DE CORREO ELECTRÓNICO

Utilizando el atributo “**type=”email”**” en la etiqueta <**input**> podemos pedir la dirección de correo electrónico al usuario de nuestra web. El navegador se encarga de comprobar o validar que el usuario introduzca correctamente, con su formato correspondiente, una dirección de correo electrónico. ***¡Veámoslo en la práctica 11.3!***

CAMPO DE TEXTO PARA DIRECCION WEB

Si lo que queremos es pedir una dirección web utilizaremos el atributo “**type=”url”**”. El navegador también se encarga de validar que la dirección web introducida sea correcta***. ¡Vayamos a la práctica 11.4!***

BOTON DE ENVÍO

Hasta ahora hemos recogido los datos de nuestro visitante de nombre, apellidos, e-mail y web ¿Pero cómo le decimos al navegador que nos envíe los datos o que realice la acción que hemos indicado en la etiqueta <**form**>? Para eso tenemos el atributo “**type=”submit”**”. ***¡Veámoslo en la práctica 11.5!***

CAMPO DE TEXTO DE BÚSQUEDA

Es muy habitual, en muchas de las páginas web que visitamos, encontrarnos con campos de búsqueda. Para ello podemos utilizar el atributo “**type=”search”**” ***¡Tal como lo puede comprobar en la práctica 11.6!***

CAMPOS DE TRANSFERENCIA DE ARCHIVOS

La etiqueta <**input type=file**> permite transferir un archivo desde el ordenador del usuario que visita nuestra web hasta otro ordenador de tipo servidor. ***¡Vamos a probarlo realizando la práctica 11.7!***

A tener en cuenta:

* Por supuesto pueden utilizarse los atributos habituales de la etiqueta <**input**>, pero hay que reseñar que el atributo “**maxlength**” permite fijar el tamaño máximo del archivo a transferir.
* Como hemos visto en la declaración de la etiqueta <**form**>, es preciso usar los atributos “**method=”post”**” y “**enctype=”multipart/form-data”**” según el formato correcto del archivo.
* Existe también el atributo “**accept**” que permite limitar la transferencia a ciertos tipos de archivos concretos.
* Observe que esta etiqueta no sirve más que para seleccionar el archivo que se quiere enviar. Para enviarlo realmente se necesita un programa en **PHP** u otro lenguaje de programación.

CAMPOS DE CONTRASEÑA

Se trata de un simple campo de texto, <**input type=”password”**>, en donde el texto introducido se sustituye, en la visualización, por puntos o asteriscos.

Este campo de contraseña no protege los datos ya que se transmiten sin cifrar. Únicamente protege de las personas que pudieran estar mirando la pantalla en el momento de escribir la contraseña. ***¡Veamos su funcionamiento en la práctica 11.8!***

CAMPOS OCULTOS

Este tipo de campo, <**input type=”hidden”**>, permite capturar datos que no estarán visibles al visitante de la página. Se utilizan junto con programas en **JavaScript** o **PHP** entre otros. Si escribimos su código veremos que no aparecen en la web.

***¡Compruébelo realizando la práctica 11.9!***

BOTON DE ANULACION

Este campo, <**input type=”reset”**>, genera un botón que borra todo lo que hayamos escrito en los diferentes campos del formulario. ***Como siempre, veámoslo en la práctica 11.10.***

Como en otros casos, el texto “**Restablecer**” que muestra el botón, lo selecciona automáticamente el navegador, pero es posible cambiarlo utilizando el atributo “**value**” de la etiqueta <**input**>.

CAMPO DE TEXTO CON FORMATO NUMÉRICO

La etiqueta <**input type=”number”**> genera un campo de tipo numérico semejante a los controles numéricos típicos de Windows. Tiene el siguiente formato:

**<input type="number" name="q" min="2" max="10" step="2" value="2" />**

Los atributos:

* **Min**. Indica el valor mínimo del contador.
* **Max**. Indica el valor máximo del contador.
* **Step**. Determina el incremento del contador cada vez que se hace clic en las flechitas.
* **Value**. Indica el valor inicial del contador.

Esta etiqueta solo es reconocida por ahora en el navegador “**Opera**”. El resto de navegadores no la tienen en cuenta y simplemente incluyen un campo de texto normal y corriente que debe completarse manualmente.

CAMPO DE TEXTO CON FORMATO DE FECHA

Al igual que en el caso anterior las etiquetas <**input**>, “**type=”date”**”, “**type=”month”**”, “**type=”week”**” y “**type=”time”**” solo son reconocidas por el navegador “**Opera**” y muestran los típicos recuadros de fechas para seleccionar un día determinado que podemos encontrar por ejemplo en las páginas webs de hoteles o reservas de vuelos.

CAMPO DE TEXTO DE COLOR

Y la etiqueta <**input type=”color”**> permitirá escoger un color. A día de hoy no está aún implementada en ningún navegador.

CAMPO DE RANGO

La etiqueta <**input type=”range”**> permite crear controles deslizantes iguales a los de Windows. Tiene el siguiente formato:

**<input type="range" name=”deslizate" min="1" max="5" step="1" value="3" />**

Los atributos:

* **Min**. Indica el valor mínimo del contador.
* **Max**. Indica el valor máximo del contador.
* **Step**. Determina el paso de avance al deslizar la pestaña.
* **Value**. Indica el valor inicial del contador.

***Para ver su funcionamiento realice la práctica 11.11.***

AREA DE TEXTO

En ocasiones es necesario proveer de un campo con suficiente espacio para que el usuario pueda escribir con más libertad. Es el caso de campos para comentarios, sugerencias, etc. En estos casos se utiliza la etiqueta <**textarea**> con su correspondiente cierre <**/textarea**>.

Los atributos de la etiqueta <**textarea**> son:

* **Name**. Asigna un nombre para este elemento.
* **Maxlength**. Determina el número máximo de caracteres que el usuario puede introducir en el área de texto.
* **Cols**. Define el número de caracteres visibles en anchura para el área de texto. Puede reemplazarse por la propiedad **CSS** “**width**”.
* **Rows**. Define el número de caracteres visibles en altura para el área de texto. Puede reemplazarse por la propiedad **CSS** “**height**”.
* **Readonly**. Indica que el usuario no puede modificar el valor por defecto definido en el área de texto.
* **Placeholder**. Permite incluir una sugerencia acerca del valor que se debe introducir en el campo. Esta sugerencia aparece de color gris en el campo tras cargar la página y desaparece cuando el usuario sitúa el foco en el elemento afectado.
* **Autofocus**. Sitúa el foco sobre el campo una vez cargada la página.
* **Required**. Convierte al campo en obligatorio.
* **Wrap**. Especifica la forma en que se gestionan los saltos automáticos de la línea de texto a la hora de enviar el contenido. Con **wrap=”hard”** se inserta un carácter de salto de línea junto con el texto. Con **wrap=”soft”**, que es el valor por defecto, no se inserta ningún carácter de salto de línea a la hora de transmitir la información.

Es posible indicar un valor por defecto en el área de texto pero no mediante el atributo “**value**” como en otros tipos de campos sino directamente escribiendo el texto entre las etiquetas <**textarea**>.

***¡Veámoslo en la práctica 11.12!***

BOTONES DE SELECCIÓN ÚNICA – BOTONES DE OPCIÓN

También conocidos como botones de radio se pueden mostrar mediante la etiqueta <**input type=”radio”**>. Tienen la particularidad de que son mutuamente excluyentes entre sí. Dicho de otro modo solo puede estar uno seleccionado. Tiene los siguientes atributos:

* **Name**. Asigna un nombre para este elemento. Es obligatorio. Debe ser igual para todos los botones de radio.
* **Checked**. Permite preseleccionar un botón de radio.
* **Value**. Permite atribuir un valor a cada botón de radio lo que permitirá realizar algún procesamiento posterior.

***¡Veamos cómo funcionan realizando la práctica 11.13!***

BOTONES DE SELECCIÓN MÚLTIPLE – CASILLAS DE VERIFICACIÓN

Si lo que queremos es seleccionar varias opciones a la vez utilizaremos los campos de tipo **checkbox**, <**input type=”checkbox”**>, conocidas en el entorno Windows como “casillas de verificación”. Tiene los siguientes atributos:

* **Name**. Asigna un nombre para este elemento. Es obligatorio. No es necesario que el nombre sea el mismo para todas las casillas. Si más adelante queremos utilizar las casillas en un programa de **PHP** o **JavaScript** tendremos que dar un nombre diferente a cada casilla.
* **Checked**. Permite preseleccionar una casilla.
* **Value**. Permite atribuir un valor a cada casilla lo que permitirá realizar algún procesamiento posterior.

***¡Veamos su funcionamiento realizando la práctica 11.14!***

LISTA DESPLEGABLE

La etiqueta <**select**> indica al navegador el uso de una lista desplegable. Los elementos de la lista se introducen mediante etiquetas <**option**>. Sus atributos son los siguientes:

* **Name**. Define un nombre para la lista desplegable.
* **Size**. Define el nº de elementos de la lista que serán visibles.
* **Multiple**. Habitualmente, el usuario no puede seleccionar más de un elemento de la lista desplegable. Este atributo permite seleccionar más de un elemento a la vez manteniendo pulsada la tecla “**Ctrl**”.
* **Selected**. En una lista desplegable, por defecto aparece seleccionado el primer elemento. Podemos conseguir que sea otro indicando este atributo en la etiqueta <**option**> correspondiente.
* **Value**. Permite atribuir un valor a cada elemento de la lista para realizar algún procesamiento posterior. Si no se utiliza este atributo se recogerá el texto situado tras la etiqueta <**option**>.

***¡Practiquémoslo realizando la práctica 11.15!***

CAMPO DE TEXTO CON LISTA DE SUGERENCIAS

La etiqueta <**datalist**>, agregada a un campo de texto, muestra una lista de sugerencias para facilitar la introducción del texto al usuario. El efecto es el mismo que cuando realizamos una búsqueda en Google. Además, a medida que el usuario escribe letras en el campo de texto, las sugerencias van reduciéndose a aquellas que contienen las letras que el usuario escribe.

La etiqueta <**datalist**> contiene una lista de etiquetas <**option**> con las sugerencias. Estas etiquetas <**option**> deben poseer un atributo “**value**”. ***¡Veámoslo en la práctica 11.16!***

Observe que la etiqueta <**datalist**> está ligada al campo de texto del formulario mediante un identificador “**id**” que se asocia con el atributo “**list**” de aquél.

BOTON DE ACCIÓN – BUTTON

Llegados a este punto nos parece oportuno hacer referencia a la etiqueta <**button**>. Esta etiqueta muestra un botón que al hacer clic sobre él permite ejecutar una acción específica generalmente mediante código **JavaScript**.

Al tener una etiqueta de apertura y otra de cierre permite incluir texto, imágenes u otro contenido **HTML**.

Esta etiqueta no tiene por qué estar anidada a un formulario pudiéndose usar en múltiples contextos.

Sus atributos son los siguientes:

* **Name**. Define un nombre para el botón.
* **Type**. Este atributo admite tres argumentos:
  + **Button:** se utiliza cuando el botón inicia un script. Es el valor por defecto del atributo en la mayoría de los navegadores.
  + **Submit:** para enviar el formulario (en caso de utilizar la etiqueta dentro de un elemento formulario).
  + **Reset:** para reinicializar el formulario (en caso de utilizar la etiqueta dentro de un elemento formulario).
* **Disabled**. Permite desactivar el botón cuando se carga la página. En este caso el botón aparece sombreado en gris. Es posible activarlo (enabled) con un script mediante una acción del usuario, por ejemplo después de haber aceptado las condiciones de uso.

***Un ejemplo del uso de <button> podría ser el que puede ver en la práctica 11.17:***

ORGANIZACIÓN DE CAMPOS DE FORMULARIO

En el caso de formularios largos y complejos, en ocasiones resulta útil reagrupar gráficamente ciertos elementos para organizar la página de forma lógica. Las etiquetas <**fieldset**> y <**legend**> permiten mejorar sensiblemente la ergonomía y la usabilidad de los formularios.

La etiqueta <**fieldset**> engloba los campos de formulario que usted determine. Estos campos se muestran en la pantalla rodeados por un borde.

La etiqueta <**legend**> situada justo detrás de la etiqueta <**fieldset**>, permite agregar una leyenda a modo de título de los campos que hemos agrupado.

***Aplique estas etiquetas a nuestra web realizando la práctica 11.18.***

***Para terminar de dominar este tema realice ahora la ejercitación del capítulo 11.***

CAPÍTULO 12.- TABLAS EN HTML y CSS

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

Hasta hace poco las tablas en **HTML** se utilizaban para maquetar o estructurar la disposición de las páginas web colocando los textos, las imágenes y demás contenidos dentro de celdas de tablas que a su vez estaban dentro de otras tablas y así sucesivamente hasta conseguir la presentación web deseada. Ahora con **CSS**, esta técnica ya no se utiliza y las tablas vuelven a su función: presentar la informacion, especialmente la numérica.

CREACIÓN DE UNA TABLA

Para crear tablas en **HTML** se utilizan básicamente estas tres etiquetas: <**table**> que indica el comienzo de una tabla, que se compone de filas <**tr**>, que contienen celdas <**td**>. Los pasos a seguir son:

* Se declara la apertura de la tabla con la etiqueta <**table**>.
* Esta tabla se componen de una primera fila, introducida por la etiqueta <**tr**>.
* Esta fila se compone de una primera celda, indicada con la etiqueta <**td**>.
* Escribimos el contenido de la primera celda.
* Se cierra la primera celda con la etiqueta <**/td**>.
* Abrimos la segunda celda y sucesivas con <**td**> y <**/td**>.
* Indicamos la el fin de la primera fila con la etiqueta <**/tr**>.
* Comenzamos la segunda fila con una nueva etiqueta <**tr**>.
* Especificamos las celdas contenidas en esta fila con sus correspondientes <**td**> y <**/td**>.
* Indicamos el fin de la segunda fila con la etiqueta <**/tr**>.
* Por último, después de todas las filas y celdas, cerramos la tabla con <**/table**>.

***¡Veamoslo en la práctica 12.1!***

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

Para dar formato a las tablas utilizaremos las propiedades **CSS** como ya conocemos. ***Compruébelo realizando la práctica 12.2.***

CELDAS DE LA TABLA

Las celdas pueden contener cualquier elemento **HTML**, bien sea un texto, imágenes, enlaces, fondos o incluso otras tablas. Y por supuesto, pueden tener su propio formato.

***Compruébelo realizando la práctica 12.3.***

COMBINAR CELDAS

Es posible combinar celdas tanto horizontal como verticalmente:

* Para **COMBINAR CELDAS HORIZONTALMENTE** aplicaremos a la etiqueta **HTML** <**td**>, el atributo “**colspan=”x”**”, donde “**x**” indica el nº de columnas a combinar. ***¡Veámoslo en un ejemplo realizando la práctica 12.4!***
* De igual forma **PARA COMBINAR VARIAS CELDAS EN VERTICAL** utilizaremos en la etiqueta **HTML** <**td**>, el atributo “**rowspan=”x”**”, donde “**x**” indica el nº de filas a combinar.

Resumiendo, para no tener lío, tenga en cuenta que en **HTML** todas las tablas deben disponer de una estructura regular, es decir, todas las filas de una tabla **HTML** deben tener el mismo número de filas y columnas. Para mostrar menos columnas en una fila se fusionan mediante el atributo “**colspan**”. Y si se quiere fusionar filas utilizaremos el atributo “**rowspan**”.

CELDAS DE ENCABEZADO

En una tabla podemos indicar al navegador las celdas que constituyen el encabezado o título de cada columna mediante la etiqueta <**th**> y su correspondiente de cierre <**/th**>.

Esta etiqueta es de bastante utilidad para los lectores de pantalla usados por personas invidentes. ***¡Veamos su funcionamiento realizando la práctica 12.5!***

LEYENDA DE LA TABLA

La etiquea <**caption**> asocia un título o leyenda a una tabla. Debe situarse justo después de la etiqueta de apertura <**table**> y debe aparecer una sola vez en la tabla. **¡Apliquémoslo en la práctica 12.6!**

Por defecto, la etiqueta <**caption**> muestra la leyenda de la tabla de forma centrada y en la parte superior de la tabla. Cada tabla solo puede contener una etiqueta <**caption**>.

Como siempre podemos cambiar esto aplicando estilos **CSS** con la propiedad “**caption-side**” que tiene dos valores: “**top**” y “**bottom**”.

¿Sería capaz de conseguir que la leyenda de la tabla aparezca debajo de la tabla? ¡Pruébelo! Cuando lo haya probado vuelva a poner la leyenda encima de la tabla.

AGRUPAR COLUMNAS

La etiqueta <**colgroup**> se usa para agrupar **columnas** con el objetivo de aplicar el mismo formato al conjunto de etiquetas del grupo mediante hojas de estilo **CSS**.

El atributo “**span=”x”**” determina el nº de columnas agrupadas. Por defecto el valor de “**x**” es 1.

La etiqueta <**colgroup**> se posiciona justo tras la etiqueta <**table**> o justo tras la etiqueta <**caption**> si se está usando, y delante de cualquier etiqueta <**tr**> o <**td**>. ***¡Veamos un ejemplo realizando la práctica 12.7!***

Semejante a la etiqueta <**colgroup**>, **HTML** también define la etiqueta <**col**> que se utiliza para asignar las mismas propiedades a varias columnas de forma simultánea. La diferencia es que la etiqueta <**col**> no agrupa columnas, sino que sólo asigna estilos comunes a varias columnas.

También tiene el atributo “**span=”x”**” que determina el nº de columnas a las que se les asignarán las propiedades.

Esta etiqueta no tiene su correspondiente de cierre, por lo que utillizaremos la notación de **XHTML**: <**col /**>. Se puede utilizar dentro o fuera de la etiqueta <**colgroup**>. ***Veamos un ejemplo realizando la práctica 12.8.***

ESTRUCTURA LÓGICA DE UNA TABLA

Además de filas y celdas de datos, en tablas más complejas, podemos indicar una sección de cabecera de tabla, una sección de pie de tabla y varias secciones de datos. **HTML** proporciona las siguientes etiquetas para estructurar el contenido de una tabla de una forma lógica:

* <**thead**> indica o agrupa a las celdas consideradas como cabecera de tabla.
* <**tbody**> indica o agrupa el cuerpo de la tabla, es decir el conjunto de datos de la misma.
* <**tfoot**> indica o agrupa a las celdas consideradas como pie de tabla.

Cada tabla puede contener solamente una cabecera y un pie, pero puede incluir un número ilimitado de secciones de datos.

Si en una tabla definimos una cabecera y/o un pie, las etiquetas <**thead**> y/o <**tfoot**> deben colocarse inmediatamente antes que cualquier etiqueta <**tbody**> de modo que, al dibujar la tabla, el navegador pueda preveer el pie de página antes de la recepción de las filas con las celdas de datos. Si la etiqueta <**tfoot**> aparece después de un elemento <**tbody**>, la página no se considera válida.

Los navegadores pueden usar estas etiquetas para habilitar el desplazamiento del cuerpo de la tabla independientemente de la cabecera y el pie. Además, al imprimir una tabla grande que abarque varias páginas, estos elementos pueden permitir que la cabecera y pie de la tabla se imprima en la parte superior e inferior de cada página.

Por otra parte la etiqueta <**tbody**> es indispensable para acceder al contenido de una tabla usando **JavaScript** por lo que conviene acostumbrarnos a incluirla en todas las tablas que creemos por muy sencillas que sean.

PROPIEDADES CSS PARA APLICAR A TABLAS

Además de la propiedad ya vista, “**caption-side**”, **CSS** dispone de las siguientes propiedades para modificar el aspecto de las tablas:

**Border-collapse**

La propiedad “**Border-collapse**” indica como muestra el navegador los bordes en las tablas. Su sintaxis es:

**border-collapse: collapse / separate;**

A tener en cuenta:

* **Collapse** fusiona los bordes de las celdas adyacentes y muestra un solo borde.
* **Separate** muestra los bordes de las celdas de la tabla de manera separada, es decir, fuerza a que cada celda muestre sus cuatro bordes. Es la opción por defecto.

Generalmente, los diseñadores prefieren el modelo collapse porque es el más parecido a como tradicionalmente conocemos las tablas en Word.

En realidad los bordes de la tabla están separados pero con un espacio entre los bordes de 0px. De ahí que se muestren los bordes con una anchura doble, ya que en realidad se trata de dos bordes iguales sin separación entre ellos.

**Border-spacing**

La propiedad “**Border-spacing**” indica al navegador el espacio que tiene que dejar entre los bordes de las celdas. Su sintaxis es:

**border-spacing: px / pt / %;**

A tener en cuenta:

* Si solamente se indica como valor una medida, se asigna ese valor como separación horizontal y vertical. Si se indican dos medidas, la primera es la separación horizontal y la segunda es la separación vertical entre los bordes de las celdas.
* Lógicamente esta propiedad solo funciona si previamente hemos indicado el valor “**separate**” de la propiedad “**border-collapse**”.

**Empty-cells**

La propiedad “**empty-cells**” determina el comportamiento del navegador cuando se encuentra con una celda vacía en una tabla. Su sintaxis es:

**empty-cells: show / hide;**

A tener en cuenta:

* El valor **Show** muestra la celda vacía. Los bordes y el fondo también son visibles.
* El valor **Hide** no muestra la celda vacía ni los bordes, ni el fondo.
* Esta propiedad sólo se aplica cuando el modelo de bordes de la tabla es “**separate**”.

***¡Comprobémoslo con un ejemplo realizando la práctica 12.9!***

**Table-layout**

La propiedad “table-layout” indica el algoritmo que debe utilizar el navegador para mostrar la tabla. Su sintaxis es:

**table-layout: fixed / auto;**

A tener en cuenta:

* El valor **fixed** le indica al navegador que calcule una anchura igual y proporcional para las celdas de la tabla.
* Con el valor **auto** la anchura de las columnas se adapta a su contenido. El navegador debe disponer de todos los contenidos de la tabla entera para poder aplicar un cálculo complejo que determina la anchura de cada columna. Por lo tanto la carga de la página es más lenta.