MÓDULO

Diseño de Interfaces Web

UNIDAD 2-1

REPASO HTML Y CSS PREVIOS

1. ¿Qué es HTML5?

HTML5 es la última versión de este lenguaje. No es una reformulación de las versiones anteriores, incluye todos los elementos válidos tanto en HTML 4 como en XHTML 1.0 y además es compatible con los navegadores antiguos.
En la evolución del lenguaje HTML, HTML4, finalmente, dio paso a XHTML, que es en realidad HTML 4 con una sintaxis estricta.
HTML5 incluye nuevos elementos de marcado cuya semántica está asociada con el significado de los contenidos.
En la actualidad, cada vez encontramos más Sitios Web que son semejantes a aplicaciones de escritorio que muestran datos en función de resultados basados en peticiones Ajax y que dependen de scripting del lado del cliente.

1. ¿Qué es HTML5?

Como se ha mencionado, el núcleo de HTML5 son una serie de nuevos elementos semánticos, así como varias tecnologías y APIs relacionadas.
Estos nuevos elementos semánticos, ayudan a que nuestros documentos sean más accesibles lo cual, resulta beneficioso también para el SEO (Search Engine Optimization).
Por otra parte, la introducción de audio y video basado en HTML5 significa que habrá una menor dependencia de software de terceros y plugins.

2. ¿Qué es CSS3?

Otra parte, CSS (Cascading Style Sheets) es un lenguaje de estilo que describe cómo se realiza la presentación de los contenidos HTML.
Y CSS3 es la última versión de este lenguaje.
CSS3 contiene casi todo lo que se incluye en CSS 2.1 (la versión anterior de la especificación) y, además, agrega nuevas características que incluyen soporte para selectores adicionales, sombras, esquinas redondeadas, múltiples fondos, animaciones, transparencias y mucho más.
Por ejemplo, en el pasado, con el fin de crear gradientes, sombras y esquinas redondeadas, los diseñadores de páginas web han tenido que recurrir a técnicas complicadas.

2. ¿Qué es CSS3?

CSS3 permite incluir estos y otros elementos de diseño de una manera sencilla y esto conlleva una serie de beneficios:

- Código HTML limpio y accesible
- Código mantenible
- Menos imágenes extra
- Una carga más rápida.

Podemos tomar cualquier proyecto HTML 4 válido o XHTML, y cambiando el tipo de documento (DOCTYPE) a HTML5, la página seguirá validando y su aspecto será el mismo que antes ya que los cambios y mejoras de HTML5 se han implementado de tal manera que se asegura la compatibilidad hacia atrás con la mayoría de los navegadores.

2. ¿Qué es CSS3?

Pero esto no ocurre con CSS3 en el que algunas versiones antiguas de los navegadores no tienen soporte para muchas de sus nuevas características.
Los desarrolladores han llegado a varias soluciones para proporcionar la experiencia equivalente. Para ello, se hace necesario el uso de scripts que imiten el comportamiento de estas nuevas características. Estas técnicas se conocen como polyfills.

3. Una Plantilla HTML5 básica

Veamos, a continuación, el esqueleto de un documento HTML5:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8" />
    <title>Plantilla HTML5</title>
    <meta name="description" content="HTML5" />
    <meta name="author" content="Nombre del autor" />
    <link rel="stylesheet" href="css/estilos.css" />
  </head>
  <body>
    <script src="js/scripts.js"></script>
  </body>
</html>
```

El doctype

Es el tipo de declaración de documento y es, simplemente, una manera de decirle al navegador, o cualquier otro software que analice nuestro código, de qué tipo de documento se trata.

3. Una Plantilla HTML5 básica

```
<!DOCTYPE html>
```

La validación es el proceso que asegura que un documento escrito en un determinado lenguaje cumple con sus normas y restricciones.
El proceso de validación consiste en probar página a página si el código HTML5 pasa la prueba de validación a través de los validadores. El organismo W3C ha creado una herramienta que se puede utilizar gratuitamente a través de Internet (http://validator.w3.org/)

3. Una Plantilla HTML5 básica

El elemento html

El siguiente paso en cualquier documento HTML es el elemento <html>, hemos incluido el atributo lang con un valor de “es”, que especifica que el documento está en español.

Podríamos prescindir del atributo lang y el documento validaría y funcionaría correctamente.

El elemento head

Define la codificación de caracteres que se utiliza en el documento. HTML5 reduce este meta a la mínima expresión:

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

Tiene que estar antes de cualquier elemento de contenido y dentro de los primeros 512 caracteres del documento.

3. Una Plantilla HTML5 básica

La siguiente parte de nuestro head es la siguiente:

```
<title>Plantilla HTML5</title>
<meta name="description" content="HTML5" />
<meta name="author" content="Nombre del autor" />
<link rel="stylesheet" href="css/estilos.css" />
```

Vemos que en la etiqueta link ya no es necesario incluir un atributo type con un valor de text/css.

Por tanto, hemos visto que se simplifica la sintaxis en HTML5 con respecto a lo que se definía en versiones anteriores.

Mirando el resto del documento básico, vemos que tenemos el elemento <body> y la etiqueta de cierre </html>, las cuales, no varían en relación a versiones anteriores del HTML.

3. Una Plantilla HTML5 básica

Al igual que se señaló anteriormente con el elemento link, la etiqueta <script> no requiere que se declare un atributo type.

Dado que, a todos los efectos prácticos, JavaScript es el único lenguaje de programación real utilizado en la Web, todos los navegadores asumirán que estamos usando JavaScript, incluso cuando no lo declaremos explícitamente, por lo tanto, el atributo de tipo es innecesario en los documentos HTML5:

```
<script src="js/scripts.js"></script>
```

El motivo de poner el elemento script en la parte inferior de nuestra página, es para mejorar la velocidad de carga de la página, cuando el navegador encuentra un script, se hará una

3. Una Plantilla HTML5 básica

pausa en la descarga mientras se ejecuta el código javascript, haciendo que el resto de la página no se descargue hasta que termine la ejecución. Esto se traduce en que la página tarda más en cargar cuando incluimos scripts grandes en la parte superior de la misma, aunque en algunos casos puede ser necesario colocar el script en la cabecera del documento, ya que deseamos que se ejecute antes de que el explorador inicie la presentación de la página.

En HTML5, podemos usar minúsculas, mayúsculas, o mayúsculas y minúsculas en los nombres o atributos de etiqueta, así como introducir valores de atributos sin comillas (siempre y cuando estos valores no contengan espacios u otros caracteres reservados),

3. Una Plantilla HTML5 básica

y todo será correcto para el validador. Estas dos líneas son válidas:

```
<link rel="stylesheet" href="css/styles.css" />
<link rel=stylesheet href=css/styles.css>
```

En HTML5, los atributos simplemente se pueden especificar sin valor. Por ejemplo:

```
<input type="text" disabled>
```

Es muy recomendable mantener las restricciones en cuanto a mayúsculas y minúsculas y/o seguir poniendo los valores de los atributos entre comillas.

4. Etiquetas básicas

Haremos un repaso de los principales elementos básicos ya existentes en HTML4 y XHTML, y que no varían en HTML5:

Cabeceras

Las cabeceras están definidas con los elementos <h1> hasta <h6>

```
<h1>Cabecera de nivel 1</h1>
<h2>Cabecera de nivel 2</h2>
<h3>Cabecera de nivel 3</h3>
```

En HTML5 tienen significado de mayor peso e importancia de tal forma que sólo tendremos un h1 en cada documento que contendrá lo más importante que queremos transmitir. Debemos utilizarlas de forma coherente. No tiene sentido tener un <h2> si no tenemos antes un <h1>, por ejemplo.

4. Etiquetas básicas

Párrafos

Los párrafos de texto los introduciremos mediante la etiqueta <p>

```
<p>Esto es un párrafo</p>
```

No debemos utilizar los párrafos para introducir saltos de línea, para eso ya disponemos de elementos en CSS.

Etiquetas para formatear textos

Aunque se mantengan en HTML5 y debamos conocerlas porque se usaron con anterioridad y siguen en muchas webs “antiguas”, debemos evitar utilizarlas ya que lo correcto es dar formato al documento con CSS.

4. Etiquetas básicas

Vemos marcadas las etiquetas strong y em que si se pueden usar en HTML5 para dar peso y fuerza a lo que encierran (“strong” da mucho peso a su contenido y “em” proporciona énfasis). Pero nunca para dar formato a los textos.

HTML Text Formatting Tags

Tag	Description
<b>	Defines bold text
	Defines emphasized text
<i>	Defines a part of text in an alternate voice or mood
<small>	Defines smaller text
	Defines important text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<mark>	Defines marked/highlighted text

4. Etiquetas básicas

“code” se usa para indicar que es código fuente y pre para texto preformateado. Se usarán para Aportar semántica, pero nunca para dar formato a los textos. Al igual que las siguientes para citas:

HTML "Computer Output" Tags

Tag	Description
<code>	Defines computer code text
<kbd>	Defines keyboard text
<samp>	Defines sample computer code
<var>	Defines a variable
<pre>	Defines preformatted text

HTML Citations, Quotations, and Definition Tags

Tag	Description
<abbr>	Defines an abbreviation or acronym
<address>	Defines contact information for the author/owner of a document
<bdo>	Defines the text direction
<blockquote>	Defines a section that is quoted from another source
<q>	Defines an inline (short) quotation
<cite>	Defines the title of a work
<dfn>	Defines a definition term

4. Etiquetas básicas

Enlaces

Para crear enlaces, utilizaremos la etiqueta <a> con su atributo href para indicar la url de destino del enlace

```
<a href="url">Texto del enlace</a>
```

Si queremos crear un enlace que se dirija a un sitio determinado del documento actual, haremos lo siguiente:

1. Colocaremos un ancla en el lugar al que nos queremos dirigir:
   ```
   <a id="contacto">Contacto</a>
   ```
2. En el enlace pondremos como href el símbolo # seguido del id que le hemos puesto al ancla:
   ```
   <a href="#contacto">Ir a contacto</a>
   ```

4. Etiquetas básicas

Imágenes

Para vincular imágenes a nuestro documento, utilizaremos la etiqueta <img> con su atributo src para indicar el origen de la imagen. <img>. No es necesario que introduzcamos la etiqueta de cierre.

Siempre debemos de poner el atributo alt a todas las imágenes para indicar un texto alternativo (por accesibilidad).

```
<img src="url de la imagen" alt="Texto alternativo de la imagen">
```

4. Etiquetas básicas

Tablas

Las tablas se definen con la etiqueta <table> y está dividida en filas con la etiqueta <tr>.

Una fila se divide en celdas de datos con la etiqueta <td>, aunque también se puede dividir en celdas de cabecera con la etiqueta <th>. Para crear una tabla como la siguiente:

Firstname	Lastname	Points
Jill	Smith	50
Eve	Jackson	94
John	Doe	80
Adam	Johnson	67

4. Etiquetas básicas

Introduciríamos el siguiente código html:

```
<table>
  <tr>
     <th>Firstname</th>
     <th>Lastname</th>
     <th>Points</th>
  </tr>
  <tr>
     <td>Jill</td>
     <td>Smith</td>
     <td>50</td>
   </tr>
   <tr>
     <td>Eve</td>
     <td>Jackson</td>
     <td>94</td>
   </tr>
   <tr>
     <td>John</td>
     <td>Doe</td>
     <td>80</td>
   </tr>
   <tr>
     <td>Adam</td>
     <td>Johnson</td>
     <td>67</td>
   </tr>
</table>
```

4. Etiquetas básicas

Listas

En HTML disponemos de dos tipos de listas: listas desordenadas, cuyos elementos no están numerados, y listas ordenadas, cuyos elementos están numerados de alguna forma, ya sea con números, letras, números romanos, etc.

Listas desordenadas

Una lista desordenada comienza con la etiqueta <ul>, y cada elemento de la lista comienza con la etiqueta <li>.

Los elementos de la lista están marcados con viñetas,normalmente, pequeños círculos negros.

4. Etiquetas básicas

Listas ordenadas

La única diferencia entre una lista desordenada y una ordenada es que la ordenada comienza con la etiqueta <ol>, en lugar de la etiqueta <ul>.

```
<ul>
  <li>Coffee</li>
  <li>Milk</li>
</ul>
```

- Coffee
- Milk

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

1. Coffee
2. Milk

4. Etiquetas básicas

Elementos contenedores

En HTML disponemos de elementos en bloque y elementos en línea.

Un elemento en bloque es aquel que incorpora un salto de línea antes y después del mismo, además tiende a ocupar todo el ancho del contenedor en el que se encuentra desplazando el resto de los elementos abajo en la página.

Mientras que un elemento en línea se colocará a continuación del elemento anterior, sin introducir ningún salto de línea y ocupa sólo el ancho de su contenido dejando a su derecha espacio para otros elementos en línea.

4. Etiquetas básicas

En base a esto, tenemos dos tipos de contenedores: elemento en bloque y elemento en línea.

El elemento se puede utilizar como un contenedor para agrupar otros elementos HTML. Este elemento no tiene ningún significado especial, salvo que, por tratarse de un elemento a nivel de bloque, el navegador mostrará un salto de línea antes y después de él, pero semánticamente no aporta nada al documento.

Cuando se utiliza junto con CSS, el elemento se puede utilizar para establecer atributos de estilo para grandes bloques de contenido. También se usa para aportar diseño al documento.

4. Etiquetas básicas

El elemento puede utilizarse como un contenedor para texto. Al igual que , no tiene ningún significado especial.

Cuando se utiliza junto con CSS, se puede utilizar para establecer los atributos de estilo a partes del texto.

Formularios

Los formularios HTML se utilizan para pasar datos a un servidor. Un formulario HTML puede contener elementos de entrada como campos de texto, casillas de verificación, radio botones, etc.

Para crear un formulario HTML utilizamos la etiqueta <form> y dentro del <form> introduciremos los campos que nos interesen:

4. Etiquetas básicas

En la siguiente tabla podemos ver los principales campos que podemos introducir en un formulario.

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list

Para ver ejemplos de código que utilice formularios se puede consultar la web de w3schools (http://www.w3schools.com/html/html_forms.asp)

4. Etiquetas básicas

Una página web dentro de otra

Para introducir una página web dentro de otra, utilizamos el elemento <iframe>: `<iframe src="URL"></iframe>`

Podemos indicar el alto y ancho del <iframe> mediante los atributos height y width.

```
<iframe src="demo_iframe.htm" width="200‟ height="200"></iframe>
```

Un <iframe> puede ser utilizado como destino de un enlace, para ello, sólo tenemos que indicar el nombre del <iframe> en el atributo target del enlace:

```
<iframe src="demo_iframe.htm" name="iframe_a"> </iframe>
<p><a href=http://www.w3schools.comtarget="iframe_a">W3Schools.com</a></p>
```

5. Definiendo la estructura del cuerpo

La estructura del cuerpo (el código entre las etiquetas <body>) generará la parte visible del documento.
Para los navegadores la correcta interpretación de qué hay dentro del documento que se está procesando (la semántica) puede ser crucial en muchos casos y, por tanto, la identificación de cada parte del documento se ha vuelto más relevante que nunca.
HTML5 incorpora nuevos elementos que ayudan a identificar cada sección del documento y organizar el cuerpo del mismo, pero de momento vamos a ver cómo organizar el diseño de nuestra página utilizando elementos y las nuevas etiquetas semánticas las recordaremos en el siguiente capitulo.
Veamos un diseño común encontrado en la mayoría de las webs:

5. Definiendo la estructura del cuerpo

5. Definiendo la estructura del cuerpo

En la parte superior, descrito como **Cabecera**, se encuentra el espacio donde usualmente se ubica el logo, título, subtítulos y una corta descripción del sitio web o la página.
Inmediatamente debajo, podemos ver la **Barra de Navegación**, en la cual, casi todos los desarrolladores ofrecen un menú o lista de enlaces con el propósito de facilitar la navegación a través del sitio.
El contenido más relevante de una página web se encuentra, en su centro. Esta sección presenta la información principal de la página. La mayoría de las veces está dividida en varias filas y columnas. En nuestro ejemplo se utilizan dos columnas: **Información Principal** y **Barra Lateral**, pero esta sección es muy flexible y cada diseñador la adapta a sus necesidades.

5. Definiendo la estructura del cuerpo

En la parte inferior de un diseño web clásico, siempre nos encontramos con una barra más al pie de la página, que aquí llamamos **Institucional**. Esta es el área donde, normalmente, se muestra información acerca del sitio web, el autor o la empresa, además de algunos enlaces con respecto a reglas, términos y condiciones y toda información adicional que el desarrollador considere importante compartir. Ejemplo del código con divs:

```
<div id="page">
  <div id="header">Cabecera</div>
  <div id="nav">Barra de navegación</div>
  <div id="content">Información principal</div>
  <div id="aside">Barra lateral</div>
  <div id="footer">Institucional</div>
</div>
```

6. CSS

CSS nada tiene que ver con HTML5. CSS fue adoptado como la forma de separar la estructura de la presentación y, por lo tanto, la especificación de HTML5 fue desarrollada considerando que CSS se haría cargo del diseño.

Incorporar estilos al documento

Aplicar estilos a los elementos HTML cambia la forma en que estos son presentados en pantalla. Los navegadores proveen estilos por defecto que, podemos sobrescribirlos con los nuestros usando diferentes técnicas:

Estilos en línea: Una de las técnicas más simples es la de asignar los estilos dentro de las etiquetas por medio del atributo style.

```
<p style="font-size: 20px">Mi texto</p>
```

6. CSS

Estilos embebidos: Una mejor alternativa es insertar los estilos en la cabecera del documento con el elemento <style> y utilizando selectores:

```
<style>
   p {font-size: 20px; }
</style>
```

Este método sería bueno si sólo tuviéramos un documento en nuestra página, pero habitualmente tendremos páginas formadas por varios documentos.

Archivos externos: La solución es mover todos los estilos a un archivo externo, y luego utilizar el elemento <link> para insertar este archivo dentro de cada documento que los necesite. Este método nos permite cambiar los estilos por completo, simplemente

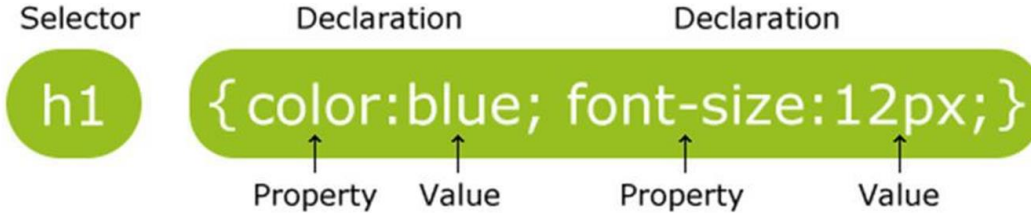
6. CSS

incluyendo un archivo diferente. También nos permite modificar o adaptar nuestros documentos a cada circunstancia o dispositivo.

```
<link rel="stylesheet" href="misestilos.css">
```

7. Funcionamiento básico de las reglas CSS

CSS define una serie de términos que permiten describir cada una de las partes que componen los estilos CSS. El siguiente esquema muestra las partes que forman un estilo CSS muy básico:



Los diferentes términos se definen a continuación:

- **Regla**: cada uno de los estilos, está compuesta por una parte denominada "selectores", un símbolo de "llave de apertura" ({), otra parte denominada "declaraciones" y, por último, un símbolo de "llave de cierre" (}).

7. Funcionamiento básico de las reglas CSS

- **Selector**: indica el elemento o elementos HTML a los que se aplica la regla CSS.
- **Declaración**: especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
- **Propiedad**: permite modificar el aspecto de una característica del elemento.
- **Valor**: indica el nuevo valor de la característica modificada en el elemento.

8. Selectores

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".
Una misma regla puede aplicarse sobre varios selectores y un mismo selector se puede utilizar en varias reglas.
A continuación, veremos los selectores más importantes:

Selector universal *

Para seleccionar todos los elementos del documento:

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

Con la regla indicada anteriormente, nos aseguramos de que todo elemento tendrá un margen interno y externo de 0 píxeles.

8. Selectores

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 { ... }
```

Si se quiere aplicar los mismos estilos a dos etiquetas diferentes, se pueden encadenar los selectores. Para ello, se incluyen todos los selectores separados por una coma (,).

```
h1, h2, h3 { color: #8A8E27; font-weight: normal; }
```

Selector descendente

Selecciona los elementos que se encuentran dentro de otros elementos.

8. Selectores

El selector del siguiente ejemplo selecciona todos los elementos de la página que se encuentren dentro de un elemento <p>:

```
p span { color: red; }
```

En el selector descendente, un elemento no tiene que ser "hijo directo" de otro. La única condición es que un elemento debe estar dentro de otro elemento.

Selector de clase .

Selecciona todos los elementos de la página con una misma propiedad class. Por ejemplo:

```
<p class="destacado">Lorem ipsum dolor sit amet...</p>
```

Se prefija el valor del atributo class con un punto (.):

```
.destacado { color: red; }
```

8. Selectores

Combinando el selector de tipo y el selector de clase, se obtiene un selector mucho más específico:

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

Se interpreta como "aquellos elementos de tipo <p> que dispongan de un atributo class con valor destacado".

Es posible aplicar los estilos de varias clases CSS sobre un mismo elemento, para esto, los diferentes valores del atributo class se separan con espacios en blanco

```
<p class="especial destacado error">Párrafo de texto...</p>
```

Al párrafo anterior se le aplican los estilos definidos en las reglas .especial, .destacado y .error:

```
.especial { font-size: 20px; }
.destacado { color: red; }
```

8. Selectores

Selector de ID #

Permite seleccionar un elemento de la página a través del valor de su atributo id. Este tipo de selectores sólo seleccionan un elemento de la página, ya que el valor del atributo id no se puede 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 lugar del punto (.):

```
#destacado { color: red; }

<p>Primer párrafo</p>
<p id="destacado">Segundo párrafo</p>
<p>Tercer párrafo</p>
```

8. Selectores

En el ejemplo anterior, el selector #destacado, solamente selecciona el segundo párrafo (cuyo atributo id es igual a destacado).
La recomendación general es la de utilizar el selector de ID cuando se quiere aplicar un estilo a un solo elemento específico de la página, y utilizar el selector de clase cuando se quiere aplicar un estilo a varios elementos diferentes de la página HTML.

Selector de hijos >

Se utiliza para seleccionar un elemento que es hijo directo de otro elemento y se indica mediante el "signo de mayor que" (>):

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

<p><span>Texto1</span></p>
<p> <a href="#"><span>Texto2</span></a> </p>
```

8. Selectores

p > span se interpreta como "cualquier elemento que sea hijo directo de un elemento <p>", por lo que el primer elemento cumple la condición del selector. Sin embargo, el segundo elemento no la cumple porque es descendiente, pero no es hijo directo de un elemento <p>

Selector adyacente +

El selector adyacente utiliza el signo + y su sintaxis es:

```
elemento1 + elemento2 { ... }
```

Selecciona todos los elementos de tipo elemento2 que cumplan las dos siguientes condiciones:

- elemento1 y elemento2 deben ser hermanos (mismo padre).
- elemento2 debe estar inmediatamente después de elemento1

8. Selectores

En el siguiente ejemplo:

```
h1 + h2 { color: red }

<body>
   <h1>Titulo1</h1>
   <h2>Subtítulo</h2>
   ...
   <h2>Otro subtítulo</h2>
   ...
</body>
```

Los estilos del selector h1 + h2 se aplican al primer elemento <h2> de la página, pero no al segundo <h2>

8. Selectores

Selector de atributos

Los selectores de atributos permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.
Los cuatro tipos de selectores de atributos son:

- [nombre_atributo], selecciona los elementos que tienen establecido el atributo llamado nombre_atributo.
- [nombre_atributo=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo con un valor igual a valor.
- [nombre_atributo~=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y al menos uno de los valores del atributo es valor.

8. Selectores

- [nombre_atributo|=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo, cuyo valor es una serie de palabras separadas con guiones, pero que comienza con valor.

A continuación, se muestran algunos ejemplos de estos tipos de selectores:

```
/* Se muestran de color azul todos los enlaces que tengan un
atributo "class", independientemente de su valor */
      a[class] { color: blue; }
/* Se muestran de color azul todos los enlaces que tengan un
atributo "class" con el valor "externo" */
     a[class="externo"] { color: blue; }
/* Se muestran de color azul todos los enlaces que apunten al sitio
"http://www.ejemplo.com" */
     a[href="http://www.ejemplo.com"] { color: blue; }
```

8. Selectores

```
/* Se muestran de color azul todos los enlaces que tengan un
atributo "class" en el que al menos uno de sus valores sea "externo"
*/
      a[class~="externo"] { color: blue; }
/* Selecciona todos los elementos de la página cuyo atributo "lang"
sea igual a "en", es decir, todos los elementos en inglés */
      *[lang="en"] {
      ...;
      }
/* Selecciona todos los elementos de la página cuyo atributo "lang"
empiece por "es", es decir, "es", "es-ES", "es-AR", etc. */
      *[lang|="es"] {
      color: red;
      }
```

8. Selectores

Pseudo-clases

El concepto pseudo-clase se introdujo para permitir la selección de elementos sobre la base de la información que se encuentra fuera de la estructura del documento.

Una pseudo-clase se compone siempre de "dos puntos" (:) seguido del nombre de la pseudo-clase y, opcionalmente, por un valor entre paréntesis.

- :link permite aplicar estilos para los enlaces que aún no han sido visitados.
- :visited aplica estilos a los enlaces que han sido visitados.
- :focus estilos que se aplican al enlace cuando este tiene el foco (acepta eventos de ratón o de teclado).

8. Selectores

- :hover estilos cuando el usuario posiciona el puntero del ratón sobre el elemento.
- :active estilos que se aplican cuando el usuario está pinchando sobre el elemento.

Las pseudo-clases :link y :visited solamente están definidas para los enlaces, pero las pseudo-clases :hover y :active se definen para todos los elementos HTML.

```
a:hover { text-decoration: none; }
```

En este ejemplo se elimina el subrayado del enlace cuando se sitúa el ratón sobre él.

8. Selectores

Pseudo-elementos

Permiten aplicar estilos a ciertas partes de un texto.

- El pseudo-elemento **:first-line** permite aplicar estilos a la primera línea de un texto.
- El pseudo-elemento **:first-letter** permite aplicar estilos a la primera letra del texto.

CSS también define dos pseudo-elementos **:before** y **:after** que nos permiten insertar contenidos antes o después de un elemento determinado. Para ello utilizaremos la propiedad CSS **content**, en la que indicaremos los contenidos que serán insertados.

```
a:after { content: " (" attr(href) ") "; }
```

8. Selectores

El código CSS anterior añade después de cada enlace de la página un texto formado por la dirección a la que apunta el enlace y mostrado entre paréntesis. Si se quiere añadir las direcciones antes de cada enlace, se puede utilizar el pseudo-elemento :before

```
a:before { content: " (" attr(href) ") ";
```

9. Propiedades CSS

La potencia de CSS la encontramos en las propiedades disponibles para modificar el aspecto de los elementos a los que se las aplicamos. Veamos las propiedades de uso más habitual:

CSS que afecta al texto	**color**: red; **font-family**: 20px; **font-size**: 20px; **font-weight**: bold; **text-align**: center; **line-height**: 50px; **letter-spacing**: 2px; **word-spacing**: 5px;	Color del texto Tipografía utilizada. Tamaño del texto (expresado en píxeles "10px", en tamaño original de la fuente "2em"..) Estilo de la fuente ("bold", "normal", "italic"..) Alineación de un texto dentro de un bloque (center, left, right o justify). Altura de una línea de texto. Sirve para centrar un texto verticalmente, indicando como valor la altura del bloque. Espacio que hay entre cada letra (se puede definir en píxeles "px" o en cantidades relativas "em"). Espacio que hay entre palabras (se puede definir en píxeles "px" o en cantidades relativas "em").
CSS que afecta a los bloques	**background-color**: red; **padding**: 20px; **margin**: 20px; **position**: relative; **left**: 10px; **top**: 10px; **float**: left **clear**: both	Color de fondo de un bloque. Espacio que hay entre el contenido de un bloque y su borde (sería el espacio interno). Espacio que hay entre bloques (espacio externo). Define el tipo de posicionamiento de un bloque (relative, absolute o fixed). Posición horizontal de un elemento. Posicion vertical de un elemento. Posiciona bloques a la izquierda (left) o derecha (right) de otros. Elimina los float declarados con anterioridad y que aún perduran.

CSS que afectan a los enlaces	**text-decoration**: none;
	cursor: pointer;
	a {
	a:hover {
		Elimina el subrayado de los enlaces (o la viñetas en las listas).
		Transforma el cursor en la mano con el dedo extendido (al usar junto a **a:hover**).
		Selector de CSS que identifica a los enlaces que contenga la página.
		Selector de CSS que identifica el momento en el que el cursor se coloca encima de un enlace.

9. Propiedades CSS

Los colores en CSS

Estas propiedades permiten establecer el color y transparencia del texto o del fondo del elemento. Se pueden expresar colores mediante:

- Códigos de colores RGB:
  - rgb(rojo verde azul), donde rojo, verde y azul son números enteros desde 0 a 255
  - rgb(rojo verde azul), donde rojo, verde y azul son porcentajes desde 0% hasta 100%.
  - #RRGGBB, donde RR, GG y BB son números hexadecimales desde 00 hasta FF.
  - #RGB, donde R, G y B son números hexadecimales desde 0 hasta F (el navegador transforma esos números en números de dos cifras repitiendo el valor, es decir, F se convierte en FF, 6 en 66, etc.
- Códigos de colores HSL (poco usado)
- Códigos de colores HWB (poco usado)

9. Propiedades CSS

- Nombres de colores (poco recomendable)
- Nombres de colores del sistema (poco recomendable)

(más información en https://www.mclibre.org/consultar/htmlcss/css/css-color.html)

Opacidad (transparencia): opacity

La propiedad opacity permite hacer que un elemento sea parcialmente transparente. El valor de esta propiedad debe ser un valor decimal entre 0 y 1 (el valor 0 hace que el elemento sea completamente transparente y el valor 1 hace que el elemento sea completamente opaco).

Si un elemento parcialmente transparente se superpone a otro elemento de otro color, el navegador mezcla los colores de los elementos.

9. Propiedades CSS

Transparencia alfa

El canal alfa es un valor numérico que se puede añadir a los colores para expresar que el color tiene cierto grado de transparencia y que se puede ver lo que hay detrás de él. El canal alfa se escribe como porcentaje o como valor decimal entre 0 y 1, en el que el 0 significa completamente transparente y el 1 completamente opaco.

El canal alfa se añade como un cuarto valor separado con una barra de los tres componentes del color rgb(), hsl() o hwb().

También se puede añadir como un cuarto valor en las notaciones rgb(rojo verde azul transp) y #RRGGBBTT

9. Propiedades CSS

Los tamaños en CSS

Unidades de medida absolutas (que no cambian):

- in: hace referencia a las pulgadas, que son iguales a 2.54cm.
- cm: se refiere a los centímetros.
- mm: hace referencia a los milímetros.
- q: se refiere a un cuarto de la unidad mm. 1q=0.248mm.
- pt: un punto es igual a 1/72 de una pulgada o 0.35mm.
- pc: una pica es igual a 12 puntos, o sea 4.23mm.
- px: esta etiqueta se refiere a los píxeles que, aunque son absolutos (0.26mm), también son relativos a la densidad de la pantalla.

9. Propiedades CSS

Unidades de medida relativas (dependen de otras):

- em: es relativa al tamaño de letra o font size establecida en el navegador (normalmente 16px).
- ex: relativa a la altura de la «x» del elemento.
- ch: es relativa al tamaño del ancho del cero en la fuente del navegador.
- %: relativa al tamaño del elemento padre.
- rem: relativa al tamaño base del documento HTML. Esta es una unidad de medida comúnmente utilizada para el tamaño de fuente en css
- vw: es relativa al ancho del viewport.
- vh: es relativa a la altura del viewport.

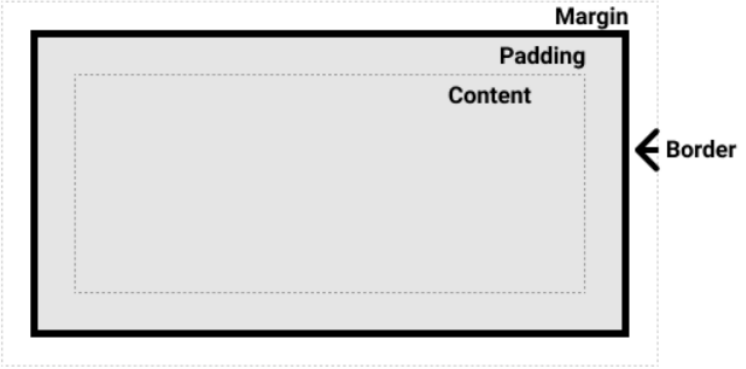
9. Propiedades CSS

Modelo de cajas

Los navegadores consideran cada elemento HTML como una caja. Una página web es en realidad un grupo de cajas ordenadas siguiendo ciertas reglas.

El modelo de cajas CSS completo se aplica a cajas que presentan comportamiento en bloque; las cajas con comportamiento en línea solo usan una parte del comportamiento definido en el modelo de cajas.

Al hacer una caja de tipo bloque en CSS tenemos los elementos siguientes:



9. Propiedades CSS

- El contenido de la caja (o content box): El área donde se muestra el contenido, cuyo tamaño puede cambiarse utilizando propiedades como **width** y **height**.
- El relleno de la caja (o **padding** box): El espacio en blanco alrededor del contenido.
- El borde de la caja (o **border** box): El borde de la caja envuelve el contenido y el relleno.
- El margen de la caja (o **margin** box): El margen es la capa más externa. Envuelve el contenido, el relleno y el borde como espacio en blanco entre la caja y otros elementos.

Veámoslos con más detalle:

9. Propiedades CSS

Margen

Podemos controlar todos los márgenes de un elemento a la vez usando la propiedad margin o cada lado individualmente usando las propiedades equivalentes sin abreviar: (margin-top, margin-right, margin-bottom, margin-left).

Acepta uno, dos, tres o cuatro valores de unidades de longitud (como px, em, etc.), porcentajes, o la palabra clave “auto”.

- Un único valor se aplicará para todos los cuatro lados .
- Dos valores se aplicarán: El primer valor para arriba y abajo , el segundo valor para izquierda y derecha .
- Tres valores aplicarán: El primero para arriba , el segundo para izquierda y derecha , el tercero para abajo .
- Cuatro valores se aplicarán en el sentido de las manecillas del reloj comenzando desde arriba. (Arriba, derecha, abajo, izquierda)

9. Propiedades CSS

El valor auto, intentará adaptar los márgenes por igual en horizontal
Ejemplos:

```
margin: 0px; /* todos los márgenes 0px*/
margin: 0px 10px; /* arriba y abajo 0px, izq y derecha 10px*/
margin: 0px 10px 20px 30px; /*agujas reloj*/
margin: 1em auto; /* 1em arriba y abajo, centrado
horizontalmente */
margin: 0 auto; /* 0px de margen vertical, centrado
horizontalmente */
margin: auto; /* 0px de margen vertical, centrado
horizontalmente */
```

Colapso de margen CSS: cuando los márgenes superior e inferior de los elementos se contraen en un solo margen (el mayor de los dos)

9. Propiedades CSS

Relleno

El relleno o padding es equivalente al margen en cuanto a su uso.

Borde

Podemos usar la propiedad border para, de forma abreviada, definir el color del borde, el estilo de borde y el ancho del borde

Sintaxis:

```
border: [border-width || border-style || border-color | inherit] ;
```

Ejemplo:

```
element {
    border: 1px solid #000;
}
```

https://developer.mozilla.org/es/docs/Web/CSS/border

9. Propiedades CSS

Centrando elementos inline

Para contenido inline, como textos o imágenes podemos usar:

```
text-align: center;
```

La propiedad text-align es hereditaria.

10. Maquetando mediante float

Usando la **propiedad float** podemos posicionar las cajas del lado izquierdo o derecho de acuerdo a nuestras necesidades.
La propiedad de CSS float hace que el elemento flote hacia un lado o al otro en el espacio disponible. Los elementos afectados por float actúan como elementos block (con la diferencia de que son ubicados de acuerdo al valor de esta propiedad y no el flujo normal del documento).
Los elementos son movidos a izquierda o derecha en el área disponible, tanto como sea posible, respondiendo al valor de float.

Por ejemplo:

10. Maquetando mediante float

```
#content {
    float: left;
    width: 660px;
    margin: 20px;
}
#aside {
    float: left;
    width: 220px;
    margin: 20px 0px;
    padding: 20px;
    background: #cccccc;
}
```

El contenido del elemento estará situado a la izquierda de la pantalla con un tamaño de 660 píxeles, más 40 píxeles de margen, ocupando un espacio total de 700 pixeles de

10. Maquetando mediante float

ancho. La propiedad float del elemento también tiene el valor left (izquierda). Esto significa que la caja generada será movida al espacio disponible a su izquierda. Debido a que la caja previa creada por el elemento fue también movida a la izquierda de la pantalla, ahora el espacio disponible será solo el que esta caja dejó libre. La nueva caja quedará ubicada en la misma línea que la primera, pero a su derecha, ocupando el espacio restante en la línea, creando así una segunda columna de nuestro diseño.

El tamaño declarado para esta segunda caja fue de 220 píxeles. Con un fondo gris y un margen interno de 20 píxeles. Como resultado final, el ancho de esta caja será de 220 píxeles más 40 los píxeles.

10. Maquetando mediante float

Propiedad clear

Después de haber usado float, tenemos que aplicar la propiedad "clear" al siguiente elemento si queremos que nos devuelva al documento su flujo normal y nos permita posicionar el siguiente elemento debajo del anterior, en lugar de a su lado.

Por ejemplo:

```
#footer {
   clear: both;
   text-align: center;
   padding: 20px;
   border-top: 2px solid #999999;
}
```

Esta propiedad, simplemente, restaura las condiciones normales del área ocupada por el elemento, no permitiéndole posicionarse

10. Maquetando mediante float

adyacente a una caja flotante. El valor usualmente utilizado es both, que significa que ambos lados del elemento serán restaurados y el elemento seguirá el flujo normal (este elemento ya no es flotante como los anteriores) y se posicionará en la siguiente línea (si es block).

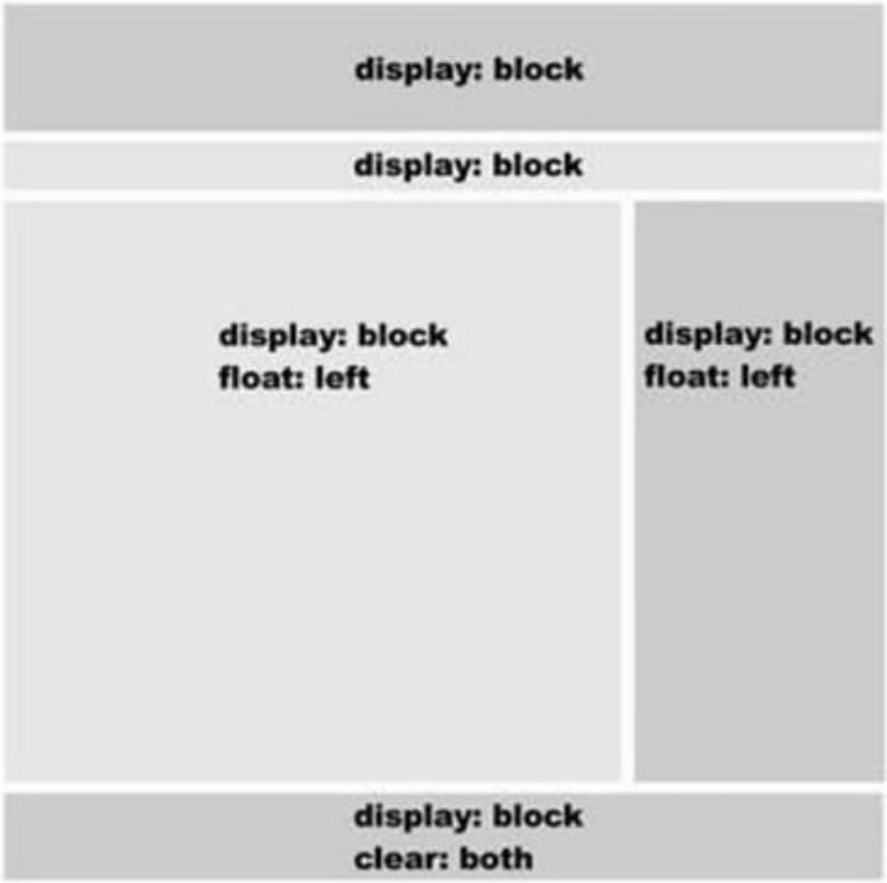
Los valores left (izquierda) y right (derecha) de la propiedad float no significan que las cajas deben estar necesariamente posicionadas del lado izquierdo o derecho de la ventana. Lo que los valores hacen es volver flotante ese lado del elemento, rompiendo el flujo normal del documento.

Si el valor es left, por ejemplo, el navegador tratará de posicionar el elemento del lado izquierdo en el espacio disponible. Si hay

10. Maquetando mediante float

espacio disponible a continuación de otro elemento, este nuevo elemento será situado a su derecha, porque su lado izquierdo fue configurado como flotante. El elemento flota hacia la izquierda hasta que encuentra algo que lo bloquea.
Cuando queremos crear varias columnas en la pantalla. Cada columna tendrá el valor left en la propiedad float para asegurar que cada columna estará contigua a la otra en el orden correcto.

10. Maquetando mediante float



11. El uso de display (inline, block e inline-block)

Ahora vamos a repasar el modelo de cajas (inline y block) para proponer otra forma de maquetar que es complementaria (no excluyente) a lo visto en el apartado anterior con float.

Cambiando el comportamiento con display

Recordemos que hay dos tipos de cajas: cajas en bloque y cajas en línea. Podemos cambiar el comportamiento de ellas mediante la propiedad display asignándole el valor inline, block o inline-block

Si una caja se define como un bloque, se comportará de forma:

- La caja fuerza un salto de línea al llegar al final de la línea.
- La caja se extenderá para llenar todo el espacio disponible que haya en su contenedor. Esto significa que la caja será tan ancha como su contenedor, y llenará el 100% del espacio disponible.

11. El uso de display (inline, block e inline-block)

- Se respetan las propiedades width y height.
- El relleno, el margen y el borde mantienen a los otros elementos alejados de la caja.

Si una caja tiene una visualización externa de tipo inline, entonces:

- La caja no fuerza ningún salto de línea al llegar al final de la línea.
- Las propiedades width y height no se aplican.
- Se aplican relleno, margen y bordes verticales, pero no mantienen alejadas otras cajas en línea.
- Se aplican relleno, margen y bordes horizontales, y mantienen alejadas otras cajas en línea.

11. El uso de display (inline, block e inline-block)

Si la caja tiene una visualización externa de tipo inline-block:

- El elemento genera una caja de elemento de bloque que fluye con el contenido circundante como si fuera una caja en línea. Y, por tanto, aplica width y height y se ubica una junto a otra.

Ejemplo:

```
#nav {
   background: #cccccc;
   padding: 5px 15px;
}
#nav li {
   display: inline;
   list-style: none;
   padding: 5px;
   font: bold 14px verdana, sans-serif;
   cursor: default;
}
```

11. El uso de display (inline, block e inline-block)

Por defecto, los ítems de una lista son posicionados unos sobre otros. Para cambiar este comportamiento y colocar cada opción del menú una al lado de la otra, cambiamos a inline los elementos <li> Esta es una forma más correcta de implementar los menús.

Por sus características, los elementos de bloque no pueden insertarse dentro de elementos en línea y tan sólo pueden aparecer dentro de otros elementos de bloque. En cambio, un elemento en línea puede aparecer tanto dentro de un elemento de bloque como dentro de otro elemento en línea.

11. El uso de display (inline, block e inline-block)

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

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

Tiene un pequeño inconveniente de márgenes que no se quitan fácilmente. Más adelante veremos que lo mejor será usar FlexBox.
(https://escss.blogspot.com/2012/03/display-inline-block-y-sus-empeno-en.html)

12. Uso de variables en CSS

Una práctica bastante recomendable es hacer uso de variables en CSS para almacenar valores como, por ejemplo, colores, tamaños, tipografías, etc.
Realmente son valores constantes que nos permitirán poder hacer cambios al principio del código y que afecten a todo el documento tal cómo haríamos en otro lenguaje de programación.
Para definirlas, en el fichero CSS lo primero que pondremos es :root y entre llaves y separadas por “;” los nombres que les asignemos con el prefijo “—” seguido de “:” y el valor.
Por ejemplo:

```
--nombre-variable: 26px;
```

Luego accedemos a ellas con la sintaxis `var(--nombre-variable);`

12. Uso de variables en CSS

Veamos un ejemplo de cómo definirlas y luego de cómo usarlas:

```
 /* Variables CSS: */
:root {
  --tam-fuente: 26px; /* Tamaños: */
  --negro: #000; /* Colores */
  --blanco: #FFF;
  --color-fondo-body: #f5f5dc;
  --color-fondo-header: #efc1a6;
  --color-fondo-nav: #eeefa6;
  --color-fondo-main: #bff4a0;
  --color-fondo-section: #a0f0f4;
  --color-fondo-aside: #b4b4e4;
  --color-fondo-footer: #eda6a6;
}
=============================================================
html {  font-size: var(--tam-fuente);}
#nav {  background-color: var(--color-fondo-nav);}
```

13. Validador CSS

Al igual que ocurre con html, disponemos de un validador para css que nos informará de los errores que tenemos en nuestros archivos de estilos. Este validador se encuentra en la url http://jigsaw.w3.org/css-validator/.

Es altamente recomendable pasar el validador antes de probar nuestro sitio web, ya que, muchas veces los estilos no se aplicarán como nosotros esperamos porque tenemos errores, y el validador nos puede ahorrar mucho tiempo a la hora de encontrar dónde está el problema.