1. Introducción al Front-End. HTML y CSS

1.1. Introducción y objetivos

Todo proyecto de desarrollo web ha de separarse inevitablemente en dos partes muy bien diferenciadas: el *front-end* (o lado cliente) y el *back-end* (o lado servidor). Estas dos partes están separadas, pero se complementan enormemente.

* El *front-end* es la parte de una aplicación web con la que los usuarios interactúan directamente, por eso se la conoce también como el «lado del cliente»*.* A grandes rasgos, podríamos decir que se trata de todo aquello que podemos ver cuando interactuamos con la aplicación.
* El *back-end* es la parte de una aplicación web que se encarga principalmente de procesar, obtener, almacenar y asegurar todos aquellos recursos o datos necesarios para el correcto funcionamiento del negocio.

El *front-end* y el *back-end* deben trabajar juntos en perfecta sincronía para que una aplicación web funcione correctamente.

Este tema ilustra qué es el *front-end* y cuáles son las principales tecnologías que nos encontramos en este lado de las aplicaciones web. Además, se estudiará qué es el HTML y por qué es tan importante, junto con CSS.

Recuerda que tienes a tu disposición ejercicios de este tema en el repositorio asociado a tu PER en el GitHub de la asignatura.

<https://github.com/UnirCs>

1.2. Qué es el front-end

Todas las aplicaciones web se separan en dos partes: el *front-end* y el *back-end.* Los usuarios únicamente interactúan con el *front-end* o «lado del cliente» o *frontal,*ya que es en esta parte donde nos encontramos una gran cantidad de elementos visuales que le indican al usuario cómo usar la aplicación.

El *front-end* es la parte de una aplicación web con la que los usuarios interactúan directamente, por eso se la conoce también como el «lado del cliente».

El mundo del *front-end* agrupa una gran cantidad de tecnologías y de disciplinas. Al ser la parte del proyecto con la que el usuario interactúa, intervienen una gran cantidad de factores y no todos son necesariamente técnicos. Un desarrollador de *front-end*tendrá que tratar, con casi toda seguridad, con las siguientes tecnologías:

* HTML. Se trata de un lenguaje de marcas (y no de un lenguaje de programación) que permite al desarrollador definir la estructura de un documento que, posteriormente, será interpretado por un navegador web para presentar contenido al usuario.

* CSS. Es un lenguaje de diseño que sirve para añadir estilos a un documento HTML. Con este lenguaje podemos definir o crear el aspecto de un componente HTML.
* JavaScript. Se trata de un lenguaje de programación que aporta dinamismo a las aplicaciones web.

Estrechamente ligados al *front-end,* existen dos amplios conceptos muy importantes:

* Experiencia de usuario. Son todos aquellos factores que influyen y conforman la interacción del usuario con la aplicación.
* Usabilidad. Es la facilidad con la que los usuarios finales de la aplicación pueden hacer uso de ella y de todas las funcionalidades que ofrece.

1.3. Conceptos básicos de HTML

Introducción

El HTML (*hyper text markup language*) es el lenguaje utilizado para describir la estructura de una página web o de parte de ella. Un documento HTML está dividido en varios elementos, los cuales sirven para indicar al navegador web cómo debe mostrar el contenido.

Este lenguaje es regulado por el World Wide Web Consortium (en adelante, *W3C*) y a lo largo de los años han publicado diferentes versiones de este.

La más reciente, HTML 5, fue publicada el 28 de octubre de 2014. Esta última versión incluye nuevas etiquetas y atributos para poder tratar con la web semántica (o web 3.0), lo que facilita el procesamiento de la información de los sitios web por máquinas y describe su contenido.

Durante toda esta sección se mostrarán diferentes ejemplos de código HTML cuyo resultado final se puede ver a través de un navegador web como Google Chrome o Mozilla Firefox. En este punto, el código que se va a mostrar y desarrollar será sencillo y no es, por tanto, necesario ningún tipo de IDE especializado. Algunas de las aplicaciones que se pueden usar para crear nuestros primeros documentos HTML son:

* [Sublime Text 4](https://www.sublimetext.com/).
* [Atom](https://atom.io/).
* [Notepad ++](https://notepad-plus-plus.org/) (no disponible para MacOS).

En el siguiente vídeo, *Introducción a HTML,* se ilustran gran parte de los contenidos de la sección de «Conceptos básicos de HTML» de forma esquemática.

Elementos y atributos del HTML

Un ejemplo sencillo de código HTML que posea diferentes elementos de este es el siguiente:

<!DOCTYPE html>

<html>

<head>

<title>UNIR - HTML</title>

</head>

<body>

<h1>Esto es un titulo de nivel 1</h1>

<br>

<p>Esto es un párrafo</p>

</body>

</html>

Un elemento HTML se indica por medio de un par de etiquetas de apertura y cierre (estas incluyen un «/» en el texto de la etiqueta) y entre ellas está su contenido.

Cada elemento aporta un significado concreto al documento y tiene una función determinada y diferenciada de la de otros. En el caso del código anterior, observamos los siguientes elementos HTML:

* <html> . Es el elemento raíz de cualquier documento HTML y siempre debe estar presente. Sirve para indicar al navegador que todo lo que esté dentro de él debe ser tratado como HTML.
* <head> . Este elemento ofrece metadatos acerca del documento HTML. Dentro de este podemos encontrar cosas como el título de la página o las hojas de estilo asociadas a ella.
* <title> . Indica el título de la página HTML. Este es el título que se verá en la pestaña del navegador web.
* <body> . Este elemento define el cuerpo del documento HTML. Se trata de un contenedor que incluye todo el contenido visible de la página (imágenes, encabezados, párrafos, tablas, etc.).
* <h1> . Se utiliza para crear *headers* o encabezados. Podemos encontrar variantes como h2, h3, h4, h5, h6, etc. Es útil para crear índices del contenido de la página.
* <p> . Este elemento define un párrafo de texto.

Encontramos también otro elemento, < br >, que no cumple con la definición antes dada, ya que presenta una etiqueta de apertura, pero no de cierre. Lo cierto es que algunos elementos HTML pueden no tener contenido, como es el caso de esta etiqueta, cuya funcionalidad es incluir un salto de línea, y son denominados elementos vacíos *(void elements*). Un elemento vacío no precisa de etiqueta de cierre.

Además, al principio del documento, encontramos una declaración de tipo  <!DOCTYPE html> . Esta debe estar presente al comienzo de cualquier documento HTML, ya que sirve para indicar al navegador web de qué tipo de documento se trata.

Existen algunas consideraciones sobre los elementos HTML que se deben conocer:

* Aunque algunos elementos HTML se muestren correctamente en el navegador sin etiquetas de cierre, es una buena práctica incluir siempre etiquetas de cierre cuando trabajemos con elementos no vacíos.
* El HTML no es *case sensitive* (no distingue entre mayúsculas y minúsculas), al menos en lo que a elementos del lenguaje se refiere. Es decir, es lo mismo escribir  <p>  que  <P> . Aquello que vaya dentro del párrafo (texto que mostrar), sí será *case sensitive.* El *W3C* recomienda el uso de minúsculas en el HTML, aunque no es obligatorio.

Como podrás imaginar, existe una gran variedad de elementos HTML que pueden ser usados para diferentes situaciones. Los elementos se agrupan en categorías, dependiendo de su uso.

A lo largo de la sección estudiaremos más elementos HTML de los que se han visto en este ejemplo, pero, si lo deseas, puedes obtener más información en la web del *W3C* con el recurso de *HTML Element Reference* desde la sección A fondo.

Los elementos HTML que hemos visto hasta ahora tienen una cosa en común: no contienen ningún atributo HTML. Los atributos sirven para indicar información adicional sobre el elemento HTML que estamos tratando.

Algunas consideraciones importantes sobre estos atributos son:

A screenshot of a computer

Description automatically generated

Tabla 1. Consideraciones sobre los atributos del HTML. Fuente: elaboración propia.

Atributo  src

El  src  se utiliza para indicar el *path* hacia un recurso externo al documento HTML. Es muy común ver este atributo junto al elemento HTML  img , el cual indica que se debe mostrar una imagen. Un ejemplo sería el siguiente:

<img src="https://www.unir.net/wp-content/uploads/2019/11/Unir\_2021\_logo.svg">

Nótese que, como  img  es un elemento vacío, no es estrictamente requerida una etiqueta de cierre. Dentro de  src  podemos encontrar dos tipos de URL diferentes.

A screenshot of a text box

Description automatically generated

Tabla 2. Tipos de URL. Fuente: elaboración propia.

Atributo  href

El  href  se utiliza para especificar la URL de un enlace. Es muy común ver este atributo junto al elemento HTML a, el cual define un enlace a otra página, ya sea externa o interna. Un ejemplo sería el siguiente:

<a href="https://www.unir.net/">Visita UNIR</a>

Atributos  width  y  height

El  width  y el  height  se utilizan para especificar la anchura y altura de un elemento HTML, respectivamente. Es muy común ver este atributo junto al elemento HTML img. Un ejemplo sería el siguiente:

<img src="https://www.unir.net/wp-content/uploads/2019/11/Unir\_2021\_logo.svg"

width="250" height="250">

Este código haría que el navegador mostrase la imagen modificando sus dimensiones, estableciendo un ancho y un alto de 250 píxeles para la misma.

Atributo  alt

El  alt  se utiliza junto al elemento HTML  img . Este atributo especifica un texto alternativo para la imagen, que se mostrará si la imagen no puede ser cargada por cualquier motivo.

<img src="https://www.unir.net/wp-content/uploads/2019/11/Unir\_2021\_logooo.svg"

width="250" height="250" alt="Logo UNIR">

Dado que la URL absoluta indicada no es válida, el navegador no podrá cargar la imagen y veremos:

A close-up of a logo

Description automatically generated

Figura 1. Ejemplo de imagen inválida. Fuente: elaboración propia.

Atributos  class  y  id

El  class  se utiliza para indicar que un elemento HTML pertenece a una determinada clase. Múltiples elementos HTML (sean o no del mismo tipo) pueden pertenecer a una misma clase. No debemos pensar en el concepto de clase de otros lenguajes de programación orientados a objetos.

En este contexto, las clases son utilizadas, principalmente, para referirnos a un grupo de elementos desde una hoja de estilos CSS. Este atributo se puede usar en cualquier elemento HTML y el nombre de la clase que indiquemos sí será *case sensitive*. Además, un elemento HTML puede pertenecer a varias clases a la vez.

Un ejemplo sería el siguiente:

<h1 class="unir-text unir-header">Esto es un titulo de nivel 1</h1>

<br>

<p class="unir-text">Esto es un párrafo</p>

En el ejemplo podemos ver que tanto el elemento  <h1>  como el elemento <p> pertenecen a la clase unir-text , aun siendo diferentes elementos HTML entre sí. Esto nos indica que, probablemente, comparten algún tipo de estilo o propiedad. Por otro lado, únicamente el elemento  <h1>  pertenece a la clase  unir-header .

Por otra parte, el  atributo id  se utiliza para asignar un identificador único a un elemento HTML. Debido a esto, no se permite que dos elementos HTML tengan el mismo identificador. De nuevo, y al igual que en el caso anterior, es útil utilizar este identificador en una hoja de estilos de CSS, aunque también JavaScript puede hacer uso de él (y del atributo  class ).

<h1 class="unir-text unir-header" id="title1">Esto es un titulo de nivel 1</h1>

En las siguientes secciones estudiaremos algunos atributos más del HTML de los que se han visto, pero, si lo deseas, puedes obtener más información en la web del *W3C* con el recurso *HTML Attribute Reference* de la sección A fondo.

Por otro lado, existen dos elementos HTML que nos permiten organizar la información y maquetar el contenido de nuestro documento HTML: las tablas y las listas. Estos dos elementos suelen estar presentes en casi cualquier documento HTML, ya que permiten organizar la información y presentarla de una manera adecuada.

Tablas HTML

Las tablas en HTML, al igual que en cualquier otro ámbito, están formadas por filas y columnas, lo que forma celdas. Las celdas pueden contener a su vez cualquier contenido HTML. Para definir una tabla en un documento HTML tendremos que hacer uso de las siguientes etiquetas:

* <table> . Indica que comienza la descripción de una tabla en HTML.
* <tr> . (*Table row).* Señala que comienza una fila de la tabla.
* <th> . (*Table header*). Indica que una celda es *header* de una columna.
* <td> . (*Table data*). Señala que se trata de una celda.

Dado que todavía no estamos utilizando ningún tipo de estilo, las tablas que crearemos lucirán muy simples en un navegador web y utilizarán los estilos por defecto de este:

* Negrita y texto centrado para los nombres de las columnas.
* Texto alineado a la izquierda para el resto de las celdas.

Un ejemplo es el siguiente:

<table>

<tr>

<th>País</th>

<th>Nº de Alumnos</th>

</tr>

<tr>

<td>España</td>

<td>Ecuador</td>

</tr>

<tr>

<td>1000</td>

<td>1000</td>

</tr>

</table>

Este código se corresponde a una tabla que tiene dos columnas, dado que hay dos elementos  <th>  en la primera fila, que es la cabecera de la tabla. La primera columna contiene países en los que está presente UNIR y la segunda tiene la cantidad de alumnos en cada uno de esos países (estos datos son inventados). Al no haber aplicado estilos, la tabla se verá de la siguiente forma:

A close-up of a sign

Description automatically generated

Figura 2. Tabla sin estilos aplicados. Fuente: elaboración propia.

Para poder diferenciar de una forma más estructural el contenido de la tabla, existen otras tres etiquetas, cuya función es agrupar contenido para el *header*, el *body* y el *footer.* Son  <thead> ,  <tbody>  y <tfooter> .

<table>

<thead>

<tr>

<th>País</th>

<th>Nº de Alumnos</th>

</tr>

</thead>

<tbody>

<tr>

<td>España</td>

<td>Ecuador</td>

</tr>

<tr>

<td>1000</td>

<td>1000</td>

</tr>

</tbody>

<tfooter>

<tr>

<td>Total</td>

<td>2000</td>

</tr>

</tfooter>

</table>

No aportan ninguna diferencia al *layout* de la tabla, salvo que mediante CSS se decida modificar sus estilos.

Listas HTML

Las listas HTML permiten agrupar una serie de términos relacionados dentro una lista. Las listas pueden ser de dos tipos.

A screenshot of a computer

Description automatically generated

Tabla 3. Tipos de lista en HTML. Fuente: elaboración propia.

Los elementos que están incluidos en la lista, ya sea ordenada o desordenada, se especifican con la etiqueta  <li>  (*list* ítem).

<p>Lenguajes de Front-End</p>

<ul>

<li>HTML</li>

<li>CSS</li>

<li>JavaScript</li>

</ul>

<p>Contenidos de la asignatura</p>

<ol>

<li>Tema 1</li>

<li>Tema 2</li>

<li>Tema 3</li>

</ol>

A close-up of a paper

Description automatically generated

Figura 3. Vista de las listas en HTML. Fuente: elaboración propia.

Las listas ordenadas disponen de un atributo HTML llamado  type  que permite indicar el tipo de numeración de la lista. Los posibles valores de este atributo son:

* type=“1” . Valor por defecto. Numeración decimal (1, 2, 3, etc.).
* type=“A” . Numeración con letras mayúsculas (A, B, C, etc.).
* type=“a” . Numeración con letras minúsculas (a, b, c, etc.).
* type=“I” . Numeración romana en mayúsculas (I, II, III, etc.).
* type=“i” . Numeración romana en minúsculas (i, ii, iii, etc.).

En cuanto a las listas desordenadas, también es posible modificar mediante CSS el carácter que se usa para indicar cada elemento de esta.

Formularios

Por último, estudiaremos uno de los elementos HTML más utilizados: los formularios. Los formularios nos permiten indicarle al usuario qué información necesitamos para poder llevar a cabo una acción determinada. En la mayoría de los casos, la información proporcionada será enviada a un servidor web para su procesamiento. En este nivel no añadiremos ninguna funcionalidad a los formularios, ya que nos dedicaremos únicamente a su diseño estructural.

El elemento HTML que debemos utilizar para trabajar con los formularios es  <form> , el cual se utiliza para crear el formulario. Este estará compuesto por todos los elementos HTML incluidos dentro de este, ya que es un contenedor.

Un formulario contendrá múltiples elementos HTML de formulario. Los más utilizados son:

* <input> . Es el elemento HTML de formularios más utilizado y sirve para almacenar la información que introduce el usuario. Dispone de un atributo  name  que siempre debe tener valor y se recomienda que coincida con el valor del atributo id. Puede ser de varios tipos, según lo que se indique en su atributo type . Existen varios valores, aunque los más usados son:
  + text . Cuadro de texto de una única línea.
  + radio . Botón de tipo radio (seleccionar una opción de varias).
  + checkbox . Botón de tipo *checkbox* (seleccionar una o ninguna opción de varias).
  + submit . Botón para completar el formulario.
  + password . Cuadro de texto de una única línea con formato de contraseña (no se ve el texto en claro).

* <label> . Define una etiqueta visual para una parte del formulario. Tiene un atributo llamado  for , cuyo valor debe ser idéntico al del atributo id del elemento HTML  <input>  al que se quiere enlazar.

El aspecto visual de los formularios, al igual que el de cualquier otro elemento HTML, puede ser modificado mediante hojas de estilo CSS. A continuación, veremos un ejemplo de formulario que ilustra lo expuesto.

Se trata de un formulario de registro en una aplicación interna de UNIR. Para ello es necesario aportar cierta información básica:

<form>

<label for="nombre">Nombre</label><br>

<input type="text" name="nombre" id="nombre"><br>

<label for="apellidos">Apellidos</label><br>

<input type="text" name="apellidos" id="apellidos"><br>

<label for="password">Contraseña</label><br>

<input type="password" name="password" id="password"><br>

<label for="passwordConf">Confirmar contraseña</label><br>

<input type="password" name="passwordConf" id="passwordConf"><br>

<p> Seleccione los estudios de mayor nivel que ha completado </p>

<input type="radio" name="estudios" id="grado" value="grado">

<label for="grado">Grado Universitario</label><br>

<input type="radio" name="estudios" id="master" value="master">

<label for="master">Máster Universitario</label><br>

<input type="radio" name="estudios" id="phd" value="phd">

<label for="phd">Doctorado</label><br>

<p> Indique con qué tecnologías tiene experiencia </p>

<input type="checkbox" name="java" id="java" value="java">

<label for="java">Java</label><br>

<input type="checkbox" name="html" id="html" value="html">

<label for="html">HTML</label><br>

<input type="checkbox" name="aws" id="aws" value="aws">

<label for="aws">AWS</label><br>

<br>

<input type="submit" value="Confirmar">

</form>

El ejemplo anterior (con datos ya incluidos en el formulario) se muestra así en un navegador web:

A screenshot of a computer

Description automatically generated

Figura 4. Formulario HTML visto desde el navegador. Fuente: elaboración propia.

Algunas observaciones:

* Varios elementos HTML incluyen un atributo  value . El valor de este atributo es el que sería enviado al servidor. Por ejemplo, si se seleccionase el botón de tipo  radio  «Máster Universitario», lo que se enviaría al servidor sería «master».
* Para que un grupo de botones  radio  sea considerado como tal, todos los botones que se quiera que formen parte del grupo deben tener el mismo valor en su atributo  name . En el caso del ejemplo, sería «estudios».

Si lo deseas, puedes obtener más información sobre otros tipos de elementos HTML en formularios y sobre los diferentes tipos de <input> con *HTML Form Elements* y *HTML input types*, dos recursosde la sección de A fondo.

*Display inline* y *block*

Todos los elementos HTML disponen de un modo de visión o *display* por defecto, el cual dependerá del tipo de elemento que sea. Consideramos dos tipos: *block* e *inline.*

*Display block*

Los elementos HTML con este tipo de *display* siempre comenzarán en una nueva línea y abarcarán, por defecto, el ancho máximo disponible de la pantalla. Estos elementos dispondrán siempre de márgenes superiores e inferiores, algo de lo que los elementos *inline* carecen.

Un ejemplo de elemento HTML con *display block*es  <div> , aunque hay muchos más como  <form> , <table> ,  <p>  o  <h1>-<h6> . Los ejemplos de código HTML mostrados anteriormente ilustran lo anterior.

*Display inline*

Los elementos HTML con este tipo de *display* nunca comenzarán en una nueva línea, sino que estarán presentes en la misma línea en la que son incluidos, dentro de otros elementos que podrán ser de tipo *block* o también *inline*. Abarcarán el mínimo ancho de pantalla necesario.

Un ejemplo de elemento HTML con *display inline* es <span>, aunque hay muchos más como  <a> ,  <b> , <br>  o  <label> .

< p > Este elemento HTML se muestra con *display block* y  < b > esto es un elemento HTML inline  </ b></ p > .

1.4. Conceptos básicos de CSS

Introducción

CSS (*Cascading Style Sheets*) es el lenguaje de estilos (o de diseño gráfico) que utilizan los desarrolladores para modificar la presentación de los elementos HTML en pantalla. Al igual que ocurre con el HTML, este lenguaje está regulado por el *W3C* y su última versión oficial es CSS3.

Pero ¿por qué surge CSS? Porque el HTML nunca fue desarrollado con la intención de ser un lenguaje para dar formato a un sitio web, sino que fue creado para describir su estructura y contenido.

Hasta ahora no hemos incluido ningún tipo de estilo en los ejemplos de HTML que se han mostrado, a pesar de ser posible. Podemos indicar de forma *inline* el estilo de un elemento HTML, pero esto no está recomendado por las siguientes razones, entre otras:

* Separación de funcionalidades. El HTML es un lenguaje que describe la estructura de una página web, no su visualización. CSS está específicamente pensado para separar la parte estructural de la parte visual.
* Mantenibilidad. Los estilos *inline* en el HTML aplican únicamente al elemento en el que se definen. Si utilizamos los mismos estilos *inline* en varios elementos y luego queremos cambiarlos, tendremos que hacerlo uno a uno. Sin embargo, si usamos CSS y el atributo  class , podemos hacerlo de una forma mucho más limpia.
* Múltiples temas. Podemos tener diferentes archivos CSS que representen los diferentes temas o modos de la aplicación. Un claro ejemplo son los típicos modos oscuro y luminoso de la mayoría de las aplicaciones modernas.

En el siguiente vídeo, *Introducción a CSS,* se ilustran gran parte de los contenidos de la sección 1.4 de forma esquemática.

Hojas de estilo

Las hojas de estilo CSS están compuestas por varias declaraciones y tienen una sintaxis muy sencilla:

A close-up of pink text

Description automatically generated

Figura 5. Sintaxis de una declaración en una hoja de estilos CSS. Fuente: elaboración propia.

Un ejemplo de lo anterior sería:

p {

color: green;

text-align: center;

}

Esta declaración en la imagen quiere decir que todos los elementos HTML  <p>  tendrán color verde y estarán alineados al centro. Podemos observar que:

* El selector es p. Todas las sentencias incluidas dentro del bloque indicado entre corchetes aplicarán a todos los elementos HTML  <p>  de nuestro documento. Este selector es de tipo simple.
* Dos propiedades de CSS están siendo modificadas. En primer lugar, color, que sirve para indicar el color del elemento HTML, y, por otro lado,  text-align , que sirve para indicar qué tipo de alineamiento queremos para el texto. Los valores para estas propiedades son  green  y  center , respectivamente.

Un dominio amplio de CSS pasa por conocer necesariamente todos los tipos de selectores disponibles y una gran variedad de propiedades de CSS. Esto último es algo que se obtiene con la experiencia, ya que *W3Schools* lista en su página web más de 200 propiedades distintas de CSS. En esta sección se cubrirán todos los tipos de selectores disponibles y algunas de las propiedades más importantes de CSS.

En el contexto de CSS, un selector es el mecanismo que se utiliza para encontrar (o seleccionar) los elementos HTML que se verán afectados por la sentencia. Existen varios tipos:

* Simples. Seleccionan elementos usando su nombre, identificador o clase.
* Combinados. Seleccionan elementos usando una relación entre ellos.
* De *pseudoclase*. Seleccionan elementos usando el estado de una clase.
* De atributos. Seleccionan elementos usando un atributo o su valor.

Selectores simples

Se basan en el nombre, identificador o clase de los elementos para encontrar los elementos HTML que van a ser afectados por la sentencia.

El siguiente ejemplo muestra cómo usar estos selectores:

#id {

color: red;

text-align: center;

}

.className {

color: red;

text-align: center;

}

p.center {

color: red;

text-align: center;

}

\* {

color: red;

text-align: center;

}

h2, h4, p {

color: red;

text-align: center;

}

Este código nos muestra los cinco tipos de selectores simples que hay, a los que habría que sumar el que se ha usado en el ejemplo del código de CSS con estilo. Cada uno de ellos obtiene el/los elementos HTML a estilizar de una forma diferente:

* #id . Este selector recoge un único elemento HTML, ya que busca aquel elemento cuyo atributo id sea idéntico al indicado.
* .className . Este selector puede recoger varios elementos HTML, ya que busca todos aquellos cuyo atributo  class  sea idéntico al indicado.
* p.center . Este selector puede recoger varios elementos HTML, ya que busca todos aquellos elementos  <p>  cuyo atributo  class  sea (o contenga) center.
* \* . Se trata del selector universal. Recoge todos los elementos HTML.
* h2 ,  h4 ,  p . Se trata de una agrupación de selectores simples. Es útil utilizar este formato si queremos que  <h2> ,  <h4>  y  <p>  compartan una serie de estilos independientemente de su clase.

Selectores combinados

Un selector combinado indica una relación que debe existir entre dos selectores. En CSS se distinguen cuatro tipos diferentes de selectores combinados.

form label {

background-color: black;

}

form > label {

background-color: black;

}

h1 + p {

background-color: black;

}

form ~ label {

background-color: black;

}

El fragmento de CSS muestra los cuatro tipos:

* form label . *Descendant selector.* Sirve para seleccionar todos los elementos HTML descendientes de otro. En este caso, seleccionaríamos todos los  <label>  dentro de cualquier  <form> .
* form > label . *Child selector.* Sirve para seleccionar todos los elementos HTML hijos de otro. En este caso, seleccionaríamos todos los  <label>  hijos de cualquier  <form> . La diferencia con respecto al anterior selector es que solo se tendrán en cuenta los elementos HTML cuyo padre sea el especificado en la parte izquierda del  > .
* h1 + p . *Adjacent sibling selector.* Sirve para seleccionar los elementos HTML que están directamente después de otro. En este caso, seleccionaríamos todos los  <p>  que se encuentren estrictamente después de un  <h1> .
* h1 ~ p . *General sibling selector.* Sirve para seleccionar los elementos HTML que están después de otro. En este caso, seleccionaríamos todos los  <p>  que se encuentren después de un  <h1> . La diferencia respecto al anterior es que no es necesario que el segundo elemento esté estrictamente después del primero.

Selectores de pseudoclase

Una pseudoclase en CSS se utiliza para indicar un estado especial para un elemento HTML. Es decir, son estilos condicionales que solo se muestran cuando la condición se cumple y el elemento HTML pasa a estar en el estado deseado. Por ejemplo, cuando pasamos el ratón sobre una parte de la página.

p:hover {

background-color: yellow;

}

Este código muestra una sentencia en la que se aplicará un fondo de color amarillo a los elementos  <p> cuando el ratón se pase por encima de ellos.

Selectores de atributos

Un selector de atributos se utiliza para obtener uno o varios elementos HTML a través del valor de un atributo.

p[textType="code"] {

background-color: darkgray;

}

El fragmento de CSS muestra una sentencia en la que se aplicará un color de fondo gris oscuro a todos aquellos elementos  <p>  que contengan un atributo  textType  cuyo valor sea  code .

CSS y HTML

A continuación, se muestra el código HTML que se usará en este apartado:

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="hojaDeEstilos.css">

<title>UNIR - HTML y CSS</title>

</head>

<body>

<h1 class="unir-text unir-header" id="title1">Galería artística</h1>

<br>

<p class="unir-text">Ejemplo para ilustrar HTML con CSS</p>

<br>

<h3 class="unir-text unir-header" id="title1">Obras de arte</h3>

<table>

<thead>

<tr>

<th>Impresionismo</th><th>Cubismo</th><th>Renacentismo</th>

</tr>

</thead>

<tbody>

<tr>

<td estilo="impresionista"><img

src="https://www.publicdomainpictures.net/pictures/420000/nahled/impressionsism-flowers.jpg"></td>

<td estilo="cubista"><img

src="https://www.publicdomainpictures.net/pictures/270000/nahled/cubisme-2018-1.jpg"></td>

<td estilo="abstracto"><img

src="https://www.publicdomainpictures.net/pictures/270000/nahled/art-abstrait-5.jpg"></td>

</tr>

<tr>

<td estilo="impresionista"><img

src="https://cdn.pixabay.com/photo/2016/09/28/19/48/art-1701280\_1280.jpg"></td>

<td estilo="cubista"><img

src="https://www.publicdomainpictures.net/pictures/270000/nahled/red-abstract-cubism-background.jpg"></td>

<td estilo="abstracto"><img

src="https://www.iordanoff.com/images/arte-abstracto-iordanoff-0356.jpg"></td>

</tr>

</tbody>

</table>

</body>

</html>

Para indicar el uso de un archivo CSS externo a un documento HTML es necesario incluir un  <link> dentro del elemento  <head>  del archivo HTML, tal como vemos en el fragmento de código HTML anterior. La hoja de estilos que usaremos es la siguiente:

.unir-text {

font-family: Arial, Helvetica, sans-serif;

}

.unir-header {

font-style: italic;

}

table {

width: 100%;

}

td[estilo="impresionista"]:hover {

background-color: grey;

}

td[estilo="cubista"]:hover {

background-color: seagreen;

}

td[estilo="abstracto"]:hover {

background-color: brown;

}

table, th, td {

border: 1px solid black;

text-align: center;

}

td > img {

max-width: 250px;

max-height: 250px;

}

td[estilo="impresionista"] > img {

border: 5px solid darkblue;

}

td[estilo="cubista"] > img {

border: 5px solid darkmagenta;

}

td[estilo="abstracto"] > img {

border: 5px solid darkorange;

}

Si dejamos el archivo HTML y el archivo CSS en una misma carpeta en el ordenador y abrimos el archivo HTML, podremos ver el resultado final de este ejemplo.

Dado que estamos utilizando URL absolutas para referenciar las imágenes que se incluyen en el documento HTML, es posible que, si su contenido cambia o sencillamente se elimina, nuestra página se vea de forma diferente. En este caso se trata de un ejemplo, pero, en nuestros desarrollos, trataremos de descargar las imágenes e incluirlas en nuestros proyectos para poder referenciarlas mediante URL relativas y así evitar estos inconvenientes.

Las sentencias CSS que se han usado se traducen, de arriba abajo, en lo siguiente:

* Tipo de fuente Arial, en su defecto Helvética o si no *sans-serif* para todos los elementos HTML de la clase  unir-text .
* Estilo  italic  (cursiva) para los elementos HTML de la clase  unir-header .
* Todos los elementos  <table>  tendrán un ancho del 100 % de la pantalla.
* Todas las celdas cuyo atributo estilo sea impresionista tendrán un fondo gris al pasar el ratón sobre ellas. Ocurre lo mismo con las dos siguientes sentencias.
* Todos los elementos  <table> ,  <th>  y  <td>  tendrán un borde sólido de 1 píxel de color negro y el texto contenido dentro de ellos estará alineado en el centro.
* Todos los elementos  <img>  dentro de un elemento  <td>  tendrán un ancho y un alto de 250 píxeles.
* Todos los elementos  <img>  dentro de un elemento  <td>  cuyo atributo estilo sea impresionista tendrán un borde sólido de 5 píxeles de color azul oscuro. Ocurre lo mismo con las dos últimas sentencias.

CSS cuenta con muchas más propiedades de las que se han expuesto aquí. Puedes obtener más información en la sección A fondo con los recursos *CSS Reference* y *CSS Properties.*

1.5. BEM

A la hora de trabajar con HTML y CSS de forma conjunta, es habitual que se definan los estándares para nombrar los estilos CSS que cada elemento HTML tendrá. Si bien dentro de cada empresa u organización podríamos encontrar diferentes estrategias para conseguir esto, existe una metodología de referencia usada que debemos conocer: BEM (bloque, elemento y modificador).

La metodología BEM persigue de una forma concreta el uso de los selectores escritos al atender la utilización que se les va a dar a los elementos HTML asociados. La mejor forma de entender esta metodología es mediante un ejemplo. Observa el siguiente código HTML que muestra varias tarjetas de las asignaturas del máster:

<section class="subjects">

<div class="subject active">

<h3>Desarrollo Web: Full Stack</h3>

<p>Conceptos principales de Front-End y Back-End</p>

<a href="#">Moodle DWFS</a>

</div>

<div class="subject">

<h3>Ciberseguridad Web</h3>

<p>Conceptos principales de Front-End y Back-End</p>

<a href="#">Moodle CIW</a>

</div>

<div class="subject">

<h3>Plataformas de Desarrollo de Software</h3>

<p>Herramientas no-code y low-code</p>

<a href="#">Moodle PDS</a>

</div>

Como se puede observar, en algunos elementos HTML se asocian los nombres de las clases CSS que se están usando en una hoja de estilos externa.

Ahora, observa este código donde se ha aplicado la metodología BEM:

<section class="subjects">

<div class="subjects\_\_subject--active">

<h3 class="subject\_\_title">Desarrollo Web: Full Stack</h3>

<p class="subject\_\_description">Conceptos principales de Front-End y Back-End</p>

<a href="#" class="subject\_\_url">Moodle DWFS</a>

</div>

<div class="subjects\_\_subject">

<h3 class="subject\_\_title">Ciberseguridad Web</h3>

<p class="subject\_\_description">Conceptos principales de Front-End y Back-End</p>

<a href="#" class="subject\_\_url">Moodle CIW</a>

</div>

<div class="subjects\_\_subject">

<h3 class="subject\_\_title">Plataformas de Desarrollo de Software</h3>

<p class="subject\_\_description">Herramientas no-code y low-code</p>

<a href="#" class="subject\_\_url">Moodle PDS</a>

</div>

Podemos observar lo siguiente:

* La clase  subjects  es un bloque, ya que se trata de una sección claramente diferenciada de la página que está haciendo referencia a una serie de asignaturas.
* El nombre de las clases de cada  subject  sigue la nomenclatura  bloque\_\_elemento  o elemento\_\_subelemento  (siempre con guiones bajos). Es decir, cada  subject  individual es un elemento que forma parte del bloque  subjects  y, con ello, indica las propiedades de cada  subject.
* El modificador, en este caso, es la clase que incluye el  --active  (siempre con guiones normales) y define un modificador al que se le aplica un elemento al que va asociado (en este ejemplo,  subject ). Sin embargo, no tiene por qué estar asociado a todos los elementos, por tanto, será una modificación sobre el elemento original.

De esta forma, podemos utilizar únicamente el nombre de la clase para realizar reglas CSS, por lo que no será necesario usar las etiquetas HTML. No obstante, podremos seguir haciéndolo siempre que queramos, aunque, gracias a esta metodología, será mucho más fácil entender y hacer entender nuestro código.