



Lenguajes de Marcas





Tema 2. Utilización de lenguajes de marcas en entornos web. HTML

1. Introducción. Evolución y versiones de HTML

Cuando se usan los navegadores para consultar las distintas webs, se aprecia una serie de páginas que contienen elementos como enlaces, texto, imágenes, vídeos, etc. Si se accede al código fuente de dicha página, se aprecia que el HTML es el componente básico de la web, junto con otras tecnologías como pueden ser CSS, AJAX o JavaScript.

HTML es el lenguaje utilizado para crear la mayor parte de las páginas web. Es un estándar reconocido en todos los navegadores, por tanto, todos ellos visualizan una página HTML de forma muy similar independientemente del sistema operativo sobre el que se ejecutan.

HTML fue en su origen un sistema de hipertexto (Permite crear documentos interactivos que proporcionan información adicional cuando se solicita mediante "hiperenlaces", también llamados "enlace", que relacionan dos recursos. Dichos recursos pueden ser otras páginas web, imágenes, documentos o archivos) para compartir documentos electrónicos en 1980. La primera propuesta oficial para convertir HTML en un estándar fue en 1993, aunque ninguna propuesta consiguió convertirse en estándar oficial.

HTML 2.0 fue la primera versión oficial de HTML. Se publicó el estándar en septiembre de 1995. A partir del año siguiente ya fue el W3C el que se encargó de publicar los estándares de HTML.

HTML 3.2 se publicó en enero de 1997 por el W3C. Este ya incorpora los applets de Java (programa que puede incrustarse en un documento HTML, permitiendo conseguir una gran variedad de efectos en las páginas web) y texto alrededor de las imágenes.

HTML 4.0 se publicó en abril de 1998. Entre sus novedades se encuentran las hojas de estilo CSS o mejoras en los formularios.

Tras la publicación de la versión HTML 4.01 a finales de 1999, se detecta su incompatibilidad con herramientas basadas en XML. Para evitar estos problemas se crea el lenguaje XHTML que combina la sintaxis de HTML 4.0 con la de XML.

XML es un lenguaje mucho más sencillo que otros también derivados de SGML, que permite transmitir datos con una cierta estructura a través de la web. Los documentos XML no aportan información sobre cómo se van a tratar esos datos, ni cómo se van a presentar y esta separación constituye, en sí misma, una gran ventaja para su desarrollo y mantenimiento.

Por otro lado, XML es más estricto en su sintaxis, ya que exige, por ejemplo, escribir todos los elementos y atributos en minúsculas, los valores de atributos deben ir entre comillas, todo elemento debe tener su marca de cierre, el anidamiento de elementos debe ser correcto sin solapamiento, etc.

De todas formas, era imprescindible mantener la compatibilidad con HTML4, porque estaba ya muy extendido y los navegadores que no se habían adaptado al nuevo XHTML



debían ser capaces de interpretar estos documentos como si se tratase de HTML.

Por lo tanto, se puede decir que XHTML es un lenguaje formado por los mismos elementos que HTML pero que se ajusta a las normas sintácticas de XML.

Otra ventaja de XHTML es la posibilidad de incorporar elementos procedentes de otros espacios de nombres como pueden ser MathML o SVG (Gráficos vectoriales redimensionables). Esto permitirá construir documentos con contenido científico o gráficos vectoriales, de una forma mucho mas simple y eficaz.

Ventajas de XHTML sobre HTML

En un principio, según el W3C, consorcio que se encarga del estándar HTML así como el XHTML, las ventajas de usar XHTML en lugar de HTML son las siguientes:

Si sus documentos son XHTML 1.0 puro (sin incluir otros lenguajes de etiquetado) entonces las diferencias no serán muy significativas actualmente. Sin embargo, a medida que proliferan las herramientas XML, como XSLT para la transformación de documentos, las ventajas de usar XHTML serán más visibles. XForms por ejemplo, permitirá editar documentos XHTML (u otros tipos de documentos XML) de forma sencilla. Las aplicaciones de Web Semántica serán capaces de sacar provecho de los documentos XHTML.

Si sus documentos son algo más que XHTML 1.0, por ejemplo, si incluyen MathML, SMIL, o SVG, entonces las ventajas son inmediatas: ese tipo de combinaciones no son posibles con HTML.

HTML5

La última versión de HTML es la 5. Especifica dos variantes importantes para HTML, una clásica conocida como HTML5 y una variante XHTML conocida como XHTML5. Es la primera vez que HTML y XHTML se han desarrollado en paralelo. La versión definitiva se publicó en octubre de 2014. Sus etiquetas no son reconocidas por viejas versiones de navegadores.

En esta nueva versión, se desaconseja el uso de ciertos elementos e, incluso, algunos quedan directamente obsoletos. Además, esta versión aporta nuevos elementos enfocados a distintas tareas como la presentación, el layout de la página(diseño), facilitar la inclusión de audio y video, mejorar los formularios, almacenar datos de sesión evitando usar cookies, generar eventos “server-sent” desde el servidor, permitir la geolocalización del sitio web, construir una superficie de dibujo llamada canvas, arrastrar objetos de un lugar a otro de la página, etc.

Pero esta versión, además de incluir nuevos elementos, tiene el objetivo de convertirse en una plataforma de desarrollo, que integre todas las tecnologías web. Hoy en día, para desarrollar una web, se ha de hacer uso de la combinación de diferentes lenguajes (HTML, CSS, Javascript, ...) diferentes formatos de audio y video, animaciones, etc. Esto puede generar problemas de compatibilidad con algunos navegadores y diferentes resultados al visualizar una página según el navegador y la versión utilizada. Además, exige al usuario la descarga y actualización de los plug-in necesarios para la reproducción de animaciones y videos.

HTML5, mediante la incorporación de una API, pretende ser una solución a esos problemas que se exponen en el párrafo anterior. Es decir, aspira a que todos los navegadores se



comporten de igual forma teniendo una biblioteca de funciones lo suficientemente amplia.

Otra de las mejoras de HTML5 es la simplificación de atributos.

Por último, en cuanto a sintaxis, es muy flexible y pone fin a la bifurcación entre HTML y XHTML convirtiéndose en un estándar para los dos. Algunas características son las siguientes:

- Se permiten las mayúsculas en las etiquetas.
- La etiqueta de cierre es opcional en los elementos vacíos.
- Es opcional establecer un valor a los atributos.
- Las comillas son opcionales en los atributos.

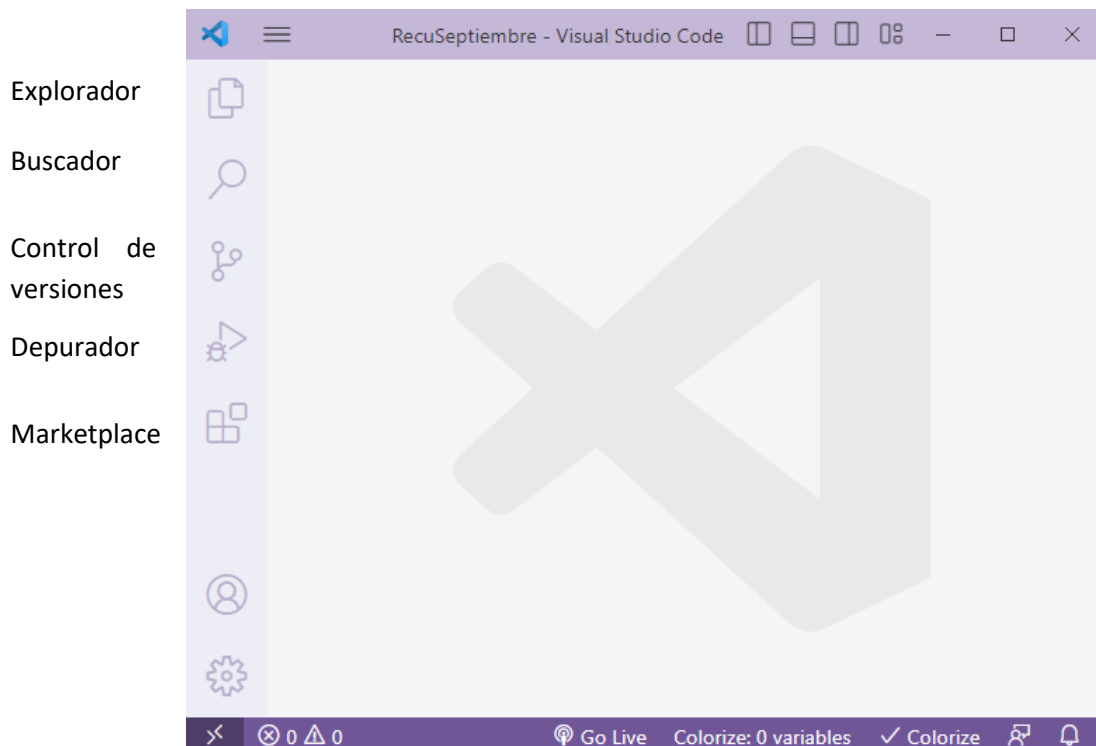
Dado que HTML5 es un lenguaje todavía en construcción, algunos de sus elementos o atributos no son soportados por todos los navegadores, lo que implica que algunos objetos, como por ejemplo los controles de los formularios, no se visualicen de igual forma.



2. Editor Visual Studio Code

Visual Studio Code (VSC) es un editor gratuito y de código abierto, desarrollado por Microsoft, que soporta diversos lenguajes de programación y es compatible para MAC, Windows y Linux. Además, es fácil de utilizar, simple y ligero. Cuenta, como la mayoría de entornos de desarrollo, con un marketplace que contiene miles de plugins que facilitarán el trabajo del desarrollador.

Una vez descargado e instalado, lo abrimos y lo primero que vemos es que hay una barra lateral (en principio situada a la izquierda) que provee de algunos iconos para cambiar de paneles. Serán paneles que usaremos bastante a lo largo de nuestros desarrollos.



En el menú Ver(View), podemos cambiar el tipo de display al que nos sea más cómodo para trabajar.

Con el comando *Ctrl+Shift+P* aparecerá una barra superior, el panel de comandos. Con esta opción podemos buscar cualquier cosa de manera rápida y sencilla, en lugar de tener que ir buscando lo que necesitamos por los distintos menús. Por ejemplo, si ponemos color, saldrán las opciones para poder cambiar el color a nuestro display.

Hemos dicho que VSC tiene miles de plugins. Para empezar, vamos a acceder al marketplace e instalaremos los siguientes plugins:

- *Live Server*: Permite una vista previa del archivo HTML en un navegador. Si tenemos activada esta funcionalidad, al guardar nuestros cambios éstos se verán reflejados en el navegador automáticamente.
 - *Auto Close Tag*: Permite cerrar automáticamente una etiqueta HTML.
 - *Auto Rename Tag*: Permite renombrar automáticamente etiquetas HTML. Al cambiar el nombre a una etiqueta, automáticamente renombra su par de cierre.
-



Si nuestro VSC se ha instalado en inglés, podemos buscar en el marketplace el paquete para español y cambiar el idioma en caso de que nos más fácil trabajar en castellano. Una vez instalado el paquete buscamos en el panel de comandos donde cambiar el idioma y realizamos el cambio.

En nuestro equipo vamos a crear un directorio principal donde iremos metiendo todos los proyectos que vamos a realizar. Creamos una carpeta que se llame LenguajesDeMarcas y dentro creamos una carpeta que contendrá nuestro primer proyecto.

Desde el VSC abrimos la carpeta del proyecto: File ->Open folder. Y una vez abierta creamos el archivo HTML con el que vamos a empezar: Si nos posicionamos sobre la carpeta que acabamos de abrir, nos saldrá un pequeño menú desde el que podremos crear un nuevo archivo, una nueva carpeta, actualizar el explorador o contraer las carpetas del explorador. Seleccionamos la opción *Nuevo Archivo* y le damos el nombre y la extensión que deseemos.

3. Conceptos básicos

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Primero doc. HTML</title>
  </head>
  <body>
    <h1>EJEMPLO</h1>
    <p>Esto es un documento HTML</p>
  </body>
</html>
```

- **Etiqueta:** En HTML5, una etiqueta es una serie de caracteres que se utiliza para marcar o identificar partes específicas del contenido en un documento HTML. Las etiquetas generalmente se escriben entre corchetes angulares < > y se utilizan para definir elementos y estructura en una página web. Puede haber etiquetas de apertura y/o cierre.
 - **Contenido:** El contenido se refiere a la información o el texto que se encuentra dentro de un elemento HTML. Puede incluir texto, imágenes, enlaces, otros elementos HTML y más. El contenido es lo que los usuarios ven en una página web y con lo que interactúan. Irá entre etiquetas de apertura y cierre, salvo en casos concretos.
 - **Elemento:** Es el conjunto de una etiqueta de apertura, un contenido y una etiqueta de cierre. En conjunto, forman una unidad que define un componente específico en la estructura de una página web.
 - **Atributo:** Definen características adicionales que se agregan a las etiquetas para proporcionar información adicional sobre un elemento. Los atributos se componen de un nombre y un valor, y se utilizan para configurar o modificar el comportamiento de un elemento. Por ejemplo, el atributo src se utiliza en la etiqueta para especificar la ubicación de una imagen.
 - **Valor del atributo:** Es la información específica que se asigna a un atributo en una etiqueta. Por ejemplo, en la etiqueta , "imagen.jpg" es el valor del atributo src. Los valores de los atributos suelen estar entre comillas (dobles o simples) para indicar que son datos del atributo.
-

4. Estructura de una página web en HTML

Los documentos HTML son archivos de texto plano formados por etiquetas de apertura y cierre (por ejemplo `<body></body>`) que permiten dar formato al contenido del documento. Cuando un navegador interpreta ese documento, el resultado es una página web. La estructura básica de una página HTML es la siguiente:

```
src > <> primerDoc.html > ...
1  <!DOCTYPE html>
2  <html lang="es">
3    <head>
4
5    </head>
6    <body>
7
8    </body>
9  </html>
10
11
12
```

Las etiquetas `<html>`, `<head>` y `<body>`, así como la declaración de tipo de documento que veremos más adelante, son opcionales, pero conviene incluirlas ya que de lo contrario el documento no será correcto ni legible.

La etiqueta `<html>` es la raíz del documento y dentro tenemos dos partes bien diferenciadas, primero tenemos la cabecera de la página, `<head>`, y después el cuerpo, `<body>`. Cabecera y cuerpo deben estar totalmente contenidas en la raíz y no pueden solaparse entre sí. Estos dos bloques pueden tener, a su vez, otros subbloques que tampoco deben solaparse.

Antes de abrir la etiqueta `<html>`, fuera de todo bloque, se puede incluir la declaración de tipo de documento. Esta declaración permite al navegador entender qué versión de HTML estamos utilizando. Esta información determinará la manera en la que el navegador procesará el documento, un DOCTYPE distinto podría implicar hasta una visualización diferente del sitio web dentro del mismo navegador.

En HTML5 la declaración es simple: `<!DOCTYPE html>`

5. Elementos de HTML

Los componentes básicos de HTML son los elementos. Cada elemento puede tener atributos y/o contenido. Si se quiere añadir atributos a un elemento, éstos se incluyen en la etiqueta de apertura del mismo. Puede haber elementos vacíos, que no tengan contenido. Los atributos, por norma general, son opcionales.

De forma genérica un elemento no vacío tendrá la siguiente sintaxis:

```
<nombre_elemento atributo1="valor1" atributo2="valor2" ...>
  CONTENIDO
</nombre_elemento>
```

Y para un elemento vacío será:

```
<nombre_elemento atributo1="valor1" atributo2="valor2" ...>
```

En html hay una serie de atributos que son los denominados globales. Estos atributos globales aplican a todos los elementos de html, aunque pueden no causar ningún efecto en algunos de ellos. Podemos acceder a todos ellos en el siguiente enlace:

https://www.w3schools.com/tags/ref_standardattributes.asp

5.1. <html>

Elemento raíz que delimita el contenido del documento. Su apertura y cierre son opcionales, si bien es cierto que de no incluirse el documento no será correcto.

Atributos que pueden incluirse:

- *xmlns*: Especifica el Espacio de nombres XML del documento. El valor por defecto es "http://www.w3.org/1999/xhtml". En XHTML es requerido y en HTML es opcional.
- *dir*: Indica el sentido de lectura de un texto cuando no se corresponde con el habitual (para el caso de estar trabajando en árabe): 'ltr' indica de izquierda a derecha y 'rtl' de derecha izquierda
- *lang*: indica el idioma por defecto del documento: 'de', 'en', 'es', etc.
- otros como *version* están en desuso.

5.2. <head>

Este elemento delimita la cabecera del documento. Al igual que pasa con el elemento raíz, estas etiquetas son opcionales, pero conviene incluirlas.

Este elemento acepta los atributos globales.

La cabecera contendrá uno o varios elementos que detallaremos más adelante.

El navegador con que abramos el documento html, analizará en primer lugar los datos recogidos en la cabecera, ya que se trata de información general de todo el documento.

5.3. <body>

Delimita el cuerpo de un documento html. Al igual que los anteriores, su apertura y cierre son opcionales pero recomendables de usar.

Repite también con los atributos *dir* y *lang* de las etiquetas que hemos visto previamente, además de los atributos globales. Vamos a ver alguno de ellos:

- *id*: sirve para identificar de manera unívoca al elemento. Este valor lo utilizaremos para identificar elementos en hojas de estilo y en scripts.
- *class*: Con este atributo asignamos un nombre de clase a un elemento, pudiendo utilizar el mismo nombre para varios elementos. Se usa para mejorar el rendimiento de las hojas de estilo.
- *title*: utilizamos este atributo para asociar un comentario a un elemento. Si añadimos un comentario de este tipo, al situar el ratón sobre el elemento se mostrará el comentario en una ventana emergente.
- *style*: sirve para aplicar información de estilo a un elemento.
- Atributos para captura de eventos con *onload*: Se utilizan para asociar una acción a un evento, en este caso se trata del evento de finalización de la carga del documento por parte del navegador.

Hay un conjunto de atributos, para dar estilo a los elementos, que están totalmente desaconsejados, ya que lo correcto es hacer uso de CSS. Se trata de los atributos background, text, link, alink, vlink, bgcolor.

5.4. Comentarios

Se utilizan para incluir líneas con información que no serán interpretadas por el navegador. No se pueden anidar. Un comentario se definirá de la siguiente manera:

```
<!-- Esto es un comentario en HTML5 -->
```

5.5. Conjuntos de atributos

En un elemento html podrá incluirse cualquier atributo que tenga definido dicho elemento, y además podrán incluirse eventos. Los eventos se incluyen de la misma forma que cualquier otro atributo.

```
<body onLoad="alert('Carga completa');">  
  <h1> Encabezado 1 </h1>  
</body>
```

Ya hemos visto atributos definidos para la cabecera y el cuerpo del documento. Algunos eventos que pueden ser de utilidad son: *onclick*, *onchange*, *onkeypress*, etc.

Para que el tratamiento de estos eventos sea de utilidad hay que conocer algún lenguaje script con el que podamos ejecutar una determinada acción cada vez que se produzca el evento.



Ejercicio 1

- Crea tu primer proyecto, dentro de la carpeta raíz LenguajesDeMarcas, para ello:
 - Crea una carpeta, que será el directorio principal del proyecto.
 - Dentro crea un documento html llamado primerosPasos.html.
 - Escribe en ese documento las siguientes líneas:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>Mi primera página</title>
5  </head>
6  <body>
7    <h1>Hola Mundo</h1>
8  </body>
9  </html>
10
```

- Añade atributos al elemento raíz, indicando que la página está en español y que el sentido del texto será de derecha a izquierda.
- Añade color de fondo con el atributo style.
- Por último, en el cuerpo del documento añadir el control de un evento. Cuando se cargue la página se mostrará una alerta que indique que se ha finalizado la carga:
 - Busca primero que tipo de evento debes capturar.
 - Después, busca un script que permita mostrar la alerta al usuario.



6. Contenido de la cabecera

La cabecera de un documento html contendrá metadatos acerca del documento. Podrá contener los siguientes elementos: <title>, <base>, <meta>, <link>, <style>, <script>.

6.1. <title>

Define el título del documento que se mostrará en la pestaña de título del navegador. También será el nombre propuesto si se quiere añadir a marcadores la página en cuestión. Se trata de un elemento no vacío.

6.2. <base>

Aquí podemos especificar la dirección URL base que se utilizará para todas las URL relativas que incluyamos en el documento. Indica la dirección raíz del sitio web. El elemento <base> es único para todo el documento. Se trata de un elemento vacío.

Por ejemplo:

```
<base href="http://www.imagenesDeGatos.com/imagenes/gatitos">
```

Y en el cuerpo podremos referenciar a imágenes de ese directorio sin ponerlo entero.

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

Podrá contener también el atributo target que indicará cómo deben ser abiertos los vínculos del documento.

Uso de rutas relativas

Es recomendable usar rutas relativas en lugar de absolutas, ya que una ruta absoluta incluye el directorio raíz y depende de la máquina en la que se encuentre el documento. Una ruta relativa, en cambio, lo es respecto del documento actual sin depender de la máquina en absoluto:

- imagenes/imagen.jpg(./imagenes/imagen.jpg): buscará imagen.jpg en una carpeta llamada imagenes que se encuentre en la misma carpeta que nuestro documento html. (en href del elemento base pondríamos imagenes/)
 - ../imagen.jpg: buscará la imagen en la carpeta raíz que también contiene una carpeta donde se aloja nuestro documento html. Con los .. volvemos a la carpeta padre. (en href del elemento base pondríamos ../)
 - ../imagenes/imagen.jpg: La imagen se encuentra en una carpeta imagenes que es hermana de la carpeta que contiene nuestro documento html. (en href del elemento base pondríamos ../imagenes/)
-



6.3. <meta>

Se trata de un elemento que provee metadatos que no pueden ser incluidos en ningún otro elemento de la cabecera. Este elemento se puede incluir de tres maneras diferentes:

- Con el atributo *charset*. En este caso solo tendrá este atributo y su valor indicará el conjunto de caracteres utilizado por el documento. Solo habrá uno por documento.
`<meta charset="utf-8">`
- Con el atributo *name*. Este atributo provee metadatos a nivel de documento organizados en pares nombre-valor. El valor se incluirá en el atributo *content*:
`<meta name="Nombre del elemento" content="Contenido asignado"/>`
- Con el atributo *http-equiv*. Este atributo aporta información adicional relacionada con el protocolo HTTP. Sus valores pueden ser, entre otros, cache-control, content-type, set-cookie, refresh, content-style-type, etc.. Los valores para su interpretación son definidos en el atributo *content*.

Vamos a detallar un poco más las principales meta etiquetas en función de su utilidad:

meta name

Información general:

- *abstract*: Proporciona un resumen muy corto de la etiqueta *description*.
- *author*: Nombre del autor del documento.
- *copyright*: Para indicar que hay información bajo derechos de autor.
- *date*: Para incluir la fecha de creación de la página.
- *generator*: Para incluir la herramienta utilizada para construir el sitio web.

Información específica para buscadores:

- *description*: Para explicar el contenido y objetivos del sitio web. Se trata de información utilizada por los motores de búsqueda para indexar la página.
- *keywords*: lista de palabras clave para que nuestro sitio sea localizable. Si se incluyen palabras no relacionadas con el tema del sitio web, se considera spam y esto es penalizado por los motores de búsqueda.
- *robots*: Sirve para controlar los robots de los motores de búsqueda. Algunos de sus posibles valores son: index, follow, noindex, nofollow, .noarchive

```
<head>
  <meta name="author" content="Sonia Gutiérrez" />
  <meta name="description" content="tienda online de componentes electrónicos" />
  <title>Mi primera página</title>
</head>
```



meta http-equiv

Tipo de contenido:

- *content-type*: Especifica el conjunto de caracteres utilizado por el documento. Es una alternativa a la declaración meta con el atributo charset. El UTF-8 contiene todos los caracteres posibles y cada uno tiene un código diferente. Para los caracteres específicos como los acentos o la ñ debemos utilizar sus códigos correspondientes como ´ o ñ
- *default-style*: Para especificar el nombre de la hoja de estilos predeterminada. Su valor debe coincidir con el atributo title de algún elemento link que enlace a una hoja de estilos, o con el atributo title de un elemento style. Esta directiva es útil cuando hay más de una hoja de estilos en un solo documento.
- *refresh*: Para indicar el tiempo de espera en segundos hasta una actualización o redirección automática de la página.

```
<meta http-equiv="refresh" content="60; URL=http://www.google.com/">
```

6.4. <link>

Define un vínculo a otro documento. Se trata también de un elemento vacío. Algunos de sus atributos más usados son:

- *href*: especifica la URL del recurso enlazado. La URL debe ser absoluta o relativa.
- *hreflang*: Este atributo indica el idioma del recurso enlazado. Es meramente informativo. Se recomienda usar este atributo solamente si href está presente.
- *type*: indica el tipo de contenido del documento vinculado, para que el navegador no lo abra en caso de no tener soporte. El valor del atributo debe ser un tipo MIME, como text/html, text/css. El uso mas habitual es para definir el tipo de hoja de estilos enlazada, y el valor más común es text/css.
- *rel*: Indica la relación del documento enlazado con el actual.
 - Con valor alternate indica que hay una versión en otro idioma:

```
<link rel="alternate" hreflang="en-gb" href="https://www.ejemplo.com/uk/" />  
<link rel="alternate" hreflang="fr" href="https://www.ejemplo.com/fr/" />
```

- Para especificar la página principal cuando hay duplicación en diferentes idiomas:

```
<link rel="canonical" href="http://www.ejemplo.es/" />
```

- Para indicar que hay un archivo rss:

```
<link rel="alternate" type="application/rss+xml" title="RSS"  
href="/actualizaciones.rdf" />
```

- Insertar un icono(favicon) en la barra de título o pestaña del navegador:

```
<link rel="icon" type="image/png" href="/imagenes/mifavicon.png" />
```



En el siguiente enlace hay un ejemplo de cómo utilizar este elemento y sus atributos:

<https://www.w3.org/QA/Tips/use-links.html.es>

6.5. <style>

Con este elemento se incluye la declaración de estilos en el propio documento en lugar de en un documento aparte. Es decir, inserta una hoja de estilos interna. Por ejemplo, con la siguiente declaración todo título h1 tendrá un borde alrededor y estará centrado:

```
<style>
  h1 {
    border-width: 1px;
    border: solid;
    border-color: red;
    text-align: center;
  }
</style>
```

Se puede dar formato a cualquier elemento que más adelante vaya a incluirse, a todos los elementos de un mismo tipo o a elementos concretos indicando su clase o su id.

Algunas propiedades CSS para empezar son:

- *color*: color del texto
- *font-size*: tamaño de letra
- *text-align*: alineación horizontal
- *border*: borde para el contenido y su relleno
- *background-color*: color de fondo

Se pueden meter tantas opciones como se quieran para el estilo de un elemento concreto o grupo de elementos. Para especificar a que elemento, o elementos, se va a asignar cierto estilo, se usan los llamados selectores. La estructura general será la siguiente:

```
selector {
  opcion1: valor 1;
  opcion2: valor2;
  ...
  opcionN: valorN;
}
```

Donde selector será:

- * : Los estilos asignados aplicarán a todo el documento.

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



- **nombreEtiqueta:** los estilos afectarán a todos los elementos de ese tipo. Se puede aplicar a más de una etiqueta, separando con comas todas ellas:

```
h1 {  
    color: red;  
}  
h1, h2, h3 {  
    color: #8a8e27;  
    font-weight: normal;  
    font-family: Arial, Helvetica, sans-serif;  
}
```

- **#idElemento:** Para aplicar nuestra definición de estilo a un elemento concreto que tenga su atributo id con el valor idElemento.
- **.clase:** El estilo definido afectará solo a los elementos cuyo atributo class sea igual a clase:

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

Si dentro de los elementos de una clase queremos aplicar estilos solo a un tipo concreto de ellos, se puede hacer como sigue:

```
p.clase {  
    color: red;  
}
```

- **nombreEtiqueta nombreEtiquetaDescendiente:** Se trata de aplicar el estilo a todos los elementos de un tipo que sean descendientes de otro especificado:

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

En este caso se aplica a todos los elementos span que se encuentren dentro de un elemento párrafo. Puede ser hijo directo o que haya más elementos de por medio.

Se pueden indicar tantos subconjuntos como se desee, siempre separados.

- **nombreEtiqueta * nombreEtiquetaDescendiente:** En este caso, el estilo se aplicará a todos los elementos de tipo nombreEtiquetaDescendiente que se encuentren contenidos en cualquier elemento que sea descendiente directo de nombreEtiqueta:

```
p * a {  
    color: red;  
}  
...  
<p><a href="#">Enlace</a></p>  
<p><span><a href="#">Enlace</a></span></p>
```



Ejercicio 2

- En el Aula Virtual encontrarás un proyecto llamado “Propiedades CSS”. Descárgalo y ábrelo en el Visual Studio Code.
- Una vez hayas entendido los estilos que incluye, prueba a cambiar los valores de las propiedades CSS incluidas.
- Pasa las declaraciones de estilos a un fichero diferente con la extensión .css. Incluye en el documento html la sentencia necesaria para referenciar a ese archivo de estilos.

6.6. <script>

Con este elemento es posible insertar scripts dentro del documento. Puede aparecer tanto en la cabecera como en el cuerpo.

Podrá definir los siguientes atributos:

- type: para indicar el lenguaje en el que está escrito el script. En HTML5 no es necesario incluirlo.

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

- src: indica la URI donde está alojado el script externo.
- defer: booleano que indica que los scripts deberían ser ejecutados una vez que el documento ha sido completamente cargado
- async: al contrario que el anterior, este booleano indica al navegador que los scripts deberían ser ejecutados tan pronto como le sea posible, pero sin bloquear el proceso de carga del documento

Puede haber scripts internos o scripts externos, y se declararán de la siguiente manera:

- Internos: En este ejemplo estamos creando una función que pondrá el foco sobre el elemento cuando sea llamada:

```
<script >
    function getfocus() {
        document.getElementById('idElement1').focus();
    }
</script>
```

- Externos: El script está incluido en un documento diferente, que tendrá extensión .js si se trata de JavaScript.
-



Ejercicio 3:

Siguiendo con el proyecto del ejercicio 1:

- Añade una imagen al cuerpo del documento, indicando la ruta en la cabecera. Comprueba que se carga correctamente.
 - Añade un conjunto de elementos meta que indiquen:
 - Autor del documento.
 - Descripción del documento.
 - Una redirección al buscador de google.
 - Añade estilos para los siguientes elementos:
 - Color para los encabezados de tipo h1 y h2.
 - Tamaño de fuente
 - Cambiar el tipo de fuente del estándar para los títulos h2.
 - Crea en el body del documento un encabezado de tipo h1 y dos de tipo h2.
 - Incluye, en la declaración de estilos, un color de fondo solo para el segundo h2.
 - Busca o crea una imagen de tamaño 80x80px, guárdala en una carpeta del proyecto y llámala logo.ico. Inserta esta imagen para que aparezca junto al título en la pestaña del navegador.
 - Añade un script para la siguiente funcionalidad:
 - Busca un código javascript para visualizar la fecha y hora del sistema.
 - Inserta dicho código en nuestro documento.
 - Visualizar el resultado de la forma FECHA: HORA:
-