

Manual HTML

¿Qué es el HTML?

HTML (**Hyper Text Markup Language**) es el "lenguaje de programación" con el que están escritas las páginas Web. El lenguaje HTML concretamente define la **estructura** ("esqueleto") de una web. Dicho de otra forma sería como los pilares de un edificio.

Es un **lenguaje de hipertexto**, es decir, permite escribir texto de forma estructurada y está compuesto por **etiquetas** que marcan el inicio (<) y el fin (/>) de cada elemento de la página.

Un documento de hipertexto puede contener imágenes, sonidos, vídeos, etc., por lo que el resultado puede considerarse como un **documento multimedia**.

- HTML describe la **estructura de las páginas Web** usando el lenguaje de etiquetas. HTML formará parte de la estructura, CSS la apariencia, y Javascript la interactividad de la página, siempre desde el lado del cliente.



- Los navegadores se encargan de **interpretar el código HTML** de los documentos y mostrar a los usuarios las páginas web resultantes del código interpretado. Los navegadores NO muestran el texto de las etiquetas HTML sino que interpretan aquello que representan. Por ejemplo, la etiqueta '***Texto***' muestra el texto comprendido entre estas etiquetas en negrita pero en ningún caso mostrará el texto que contienen las etiquetas.

- Los documentos HTML deben utilizar la extensión **HTML** o **HTM** para que puedan ser visualizados en los navegadores. Hay otras extensiones que se utilizan frecuentemente y también representan páginas web como la extensión .php.
- El desarrollo de este lenguaje de marcado es regulado por el **Consortio W3C** (*World Web Wide Consortium*) que cada cierto tiempo publica actualizaciones sobre el lenguaje. Su función es establecer el estándar del lenguaje y todas las tecnologías asociadas.
- HTML5 es la tecnología que **engloba varios lenguajes de programación** (HTML, CSS, JS) aunque en muchas ocasiones solo se hace referencia de forma errónea a las nuevas etiquetas semánticas de HTML.

Evolución cronológica.

- En 1990 **Tim Berners-Lee** (CERN) desarrolla la primera versión HTML. Es conocido como el padre de la World Wide Web. Estableció la primera comunicación **cliente-servidor** utilizando el protocolo HTTP.
- El **primer estándar** en el año 1995 fue HTML 2.0 con **finalidad divulgativa**. Al poco tiempo, el comité encargado de establecer los estándares dentro de Internet (W3C) comenzó a trabajar en el borrador de una nueva versión de HTML, la versión HTML 3.0.
- En enero de 1997 **se aprobó el estándar HTML 3.2**. Este nuevo estándar incluía las mejoras proporcionadas por los navegadores Internet Explorer y Netscape Navegador. Ambos navegadores en la actualidad han dejado de utilizarse y recibir soporte.
- En diciembre de **1997 se aprobó el estándar HTML 4.0**. Estandarizaba hojas de estilo y los scripts. Esta evolución es considerada un gran avance en la construcción de páginas web, separar la estructura (HTML) del contenido.
- En septiembre de 2001 se aprobó el estándar HTML 4.01 que incorporaba actualizaciones de la versión anterior y eliminaba aspectos poco significativos.
- A principios del 2011 surge con fuerza **HTML5**. La W3C le dio notoriedad al nuevo estándar y en el año 2012 se estandarizó para los navegadores más utilizados. Fue la primera entrada fuerte del estándar HTML5 y marcó las directrices para la versión definitiva del estándar.
- El 28 octubre del 2014 se publica la **versión definitiva de HTML5**. Desde entonces se han ido incluyendo y eliminando diferentes etiquetas para optimizarlo a las nuevas necesidades.
- La W3C publicó el 21 de junio 2016 las nuevas recomendaciones para **HTML 5.1**. Se incluyeron etiquetas nuevas y mejoras para la estandarización del lenguaje.
- La W3C publicó el 14 de diciembre del 2017 las nuevas actualizaciones para de la versión HTML 5.1 bajo el nombre de **HTML 5.2**.
- **HTML6** será el estándar actualizado (no hay fecha salida) que priorizará la seguridad.

Sitios Web vs Aplicaciones Web

Para comenzar es importante conocer la **diferencia entre página web y aplicación web**. Aparentemente pueden parecer lo mismo, pero tienen sus particularidades.

1. Página web:

Los sitios web **son estáticos**, es decir, no se actualizan muy frecuentemente. Su objetivo es proporcionar información al usuario, por lo que **no hay interacción con él**. Un ejemplo clásico son los sitios web de empresa, donde seguramente se verá la descripción de productos, servicios ofrecidos, historia de la empresa y formas de contacto (correo electrónico, teléfono, etc.).

Se construyen **utilizando HTML, CSS**, y tal vez un poco de **JavaScript**. No se requiere lenguaje de programación, y mucho menos la utilización de una base de datos.

Los sitios web son una parte enorme de la web global y desempeñan papeles importantes como informar de un evento o producto nuevo que se publicará o mostrar los servicios de una empresa en particular, pero **la interacción con el usuario es mínima o nula**.

2. Aplicaciones Web:

Una aplicación web es una página web con más características. Contiene información sobre la que **se puede interactuar e incluso modificar**. La diferencia con las aplicaciones de escritorio es que las aplicaciones web no se instalan ni se ejecutan en el equipo, sino que lo hacen a través de un navegador. Ejemplos de aplicaciones web: Gmail, Hotmail, Google Docs, Vueling, Amazon ...

Las aplicaciones Web funcionan "igual" que las aplicaciones de escritorio (Word, Photoshop, Skype), son dinámicas y están evolucionando constantemente. Dependen de la interacción del usuario para lograr su objetivo, ya sea **contribuyendo con el contenido** (YouTube, Facebook, Twitter) o la **recopilación de datos** de otras fuentes y presentarlos por pantalla (Google Analytics, Klout).

Las aplicaciones web suelen **trabajar con bases de datos** donde se almacena la información que se muestra al usuario. Las aplicaciones web **se construyen con HTML, CSS y JavaScript** (lado cliente) y otros lenguajes de programación como **PHP, Ruby, SQL o Python** (lado servidor). **NodeJS** es la versión de Javascript que se ejecuta en el entorno del servidor.

La construcción de aplicaciones web demanda una **lógica mucho más compleja** porque entran en juego más lenguajes de programación, bases de datos y nuevas herramientas de desarrollo como frameworks o librerías. Las aplicaciones web tienen un coste mayor en la creación de la aplicación, en la implantación, y en el mantenimiento regular de las mismas. Todo y con eso, este sistema sigue siendo más óptimo económicamente respecto a las aplicaciones de escritorio.

Las aplicaciones o programas que están disponibles hoy en día en el mundo de la programación pueden clasificarse básicamente en tres tipos:

Aplicación nativa

La aplicación nativa está **desarrollada y optimizada específicamente para un sistema operativo** y dependiendo de la plataforma de desarrollo del fabricante (Android, iOS, etc.).

Este tipo de aplicaciones se adaptan al 100% a las funcionalidades y características del dispositivo obteniendo así un mejor rendimiento y una mejor experiencia de uso.

Sin embargo, el desarrollo de una aplicación nativa comporta un mayor coste, puesto que si se desea realizar una aplicación multiplataforma (diversos sistemas operativos) se debe desarrollar una nueva versión del producto para cada sistema operativo, **multiplicando considerablemente los costes de desarrollo**.

Ejemplos: *Word, Excel, Photoshop, Autocad, SAP, Visual Studio Code*, etc...

Aplicación web

La aplicación web es la opción más sencilla y económica de crear aplicaciones, puesto que al desarrollar una única aplicación se reducen al máximo los costes de desarrollo. Asimismo, en este tipo de aplicaciones, puede utilizarse el famoso "**Responsive Web Design**", creando así una única aplicación adaptada para todo tipo de dispositivos.

Por contra, la aplicación web, "por ahora", ofrece una **peor experiencia de uso**, puesto que ignora las características del dispositivo y una menor seguridad ya que depende de la seguridad que ofrezca el propio navegador.

Ejemplos: *Netflix, Spotify, Microsoft office Online, Wordpress, Vueling*, o cualquier aplicación a medida para una empresa.

Aplicación híbrida

Este tipo de aplicación, también denominadas **mashups** (mezcla), aprovecha al máximo la versatilidad de un desarrollo web y tiene la capacidad de adaptación al dispositivo como una app nativa. Permite utilizar los estándares de desarrollo web (HTML5) y **aprovechar las funcionalidades del dispositivo** tales como la cámara, el GPS o los contactos. Además, comporta un menor coste que una aplicación nativa y una mejor experiencia de uso que una aplicación web.

Sin embargo, tiene un **rendimiento ligeramente inferior** al de una aplicación nativa debido a que cada página debe ser renderizada desde el servidor y supone una mayor dificultad de desarrollo.

Ejemplos: Facebook, Uber, Amazon, Tesla, Adidas, Vogue, etc...

Nomenclatura básica

Las páginas web se visualizan en el PC a través del navegador, pero se **almacenan en servidores web** compartiendo o no espacio con otras páginas. Esta relación entre el PC y el ordenador remoto se denomina **Cliente – Servidor**.

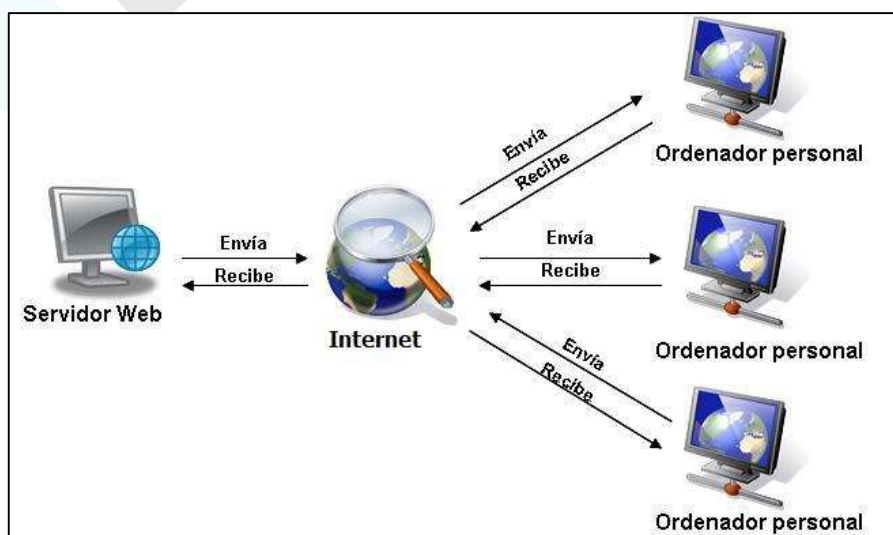
- **Cliente:** Es el equipo que solicita una página web. La herramienta para solicitar páginas web es el navegador. El cliente solicita una determinada página web y el servidor le **envía una copia de la misma con todo su contenido** (texto, imágenes, etc..) para visualizarla en el navegador.

Cada vez que se accede a una página nueva hay que realizar la petición al servidor para que nos envíe una **copia de la página web** y de todos sus elementos que la conforman.

La **memoria caché** del PC almacena las últimas páginas web visitadas y puede acelerar la carga de las mismas en el navegador sin tener que solicitarlas nuevamente al servidor. Esta característica, a priori beneficiosa para el usuario, puede estar mostrando una página que no corresponda con la versión más actualizada. Para actualizar contra el servidor **CTRL + F5**.

- **Servidor:** Son equipos activados 24h/7días que **almacenan las páginas web** y están preparados para atender las diferentes solicitudes por parte de los clientes. Un servidor puede atender a muchos clientes a la vez. La capacidad de respuesta dependerá del volumen de peticiones que pueda atender y esto siempre estará asociado al tipo de plan contratado en el hosting.

El servidor web más utilizado es **Apache** (Linux) aunque existen otros como Internet Information Server (Windows), LiteSpeed Web Server, Apache Tomcat (desarrollo Java), Nginx.



Para empezar a construir páginas web es necesario tener presente una serie de conceptos básicos sobre los servidores, como acceder a ellos y como realizar pequeñas configuraciones:

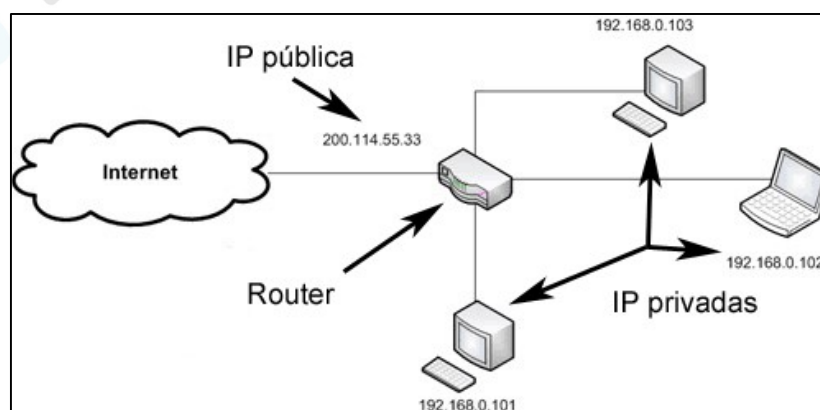
- **Número IP:** (*Internet Protocol*). Es la **identificación única de un servidor o equipo** dentro de Internet o una red. La dirección IP está compuesta por cuatro números enteros comprendidos entre 0 y 255 separados por puntos en el caso de la tecnología (IPv4 - 32bits) o por letras y números en hexadecimal en el caso de (Ipv6 - 128 bits). Por ejemplo :

a) **Ipv4:** 216 . 58 . 198 . 100 -> (es la dirección de Google).

b) **Ipv6:** 2607 : f0d0 : 4545 : 3 : 200 : f8ff : fe21 : 67cf -> (futuro nuevo estándar)

Todos los equipos informáticos, sean o no servidores, cuando se conectan a Internet disponen de su "DNI" que los identifica dentro de la red respecto al resto. Por este motivo hay que diferenciar los dos tipos de IP's que nos podemos encontrar.

1. **IP Pública:** Es la dirección IP con a que un equipo **sale a navegar por Internet**. Esta IP hace referencia al Router a través del cual se conecta. Esta dirección no es configurable y no es siempre la misma porque se va actualizando cada cierto tiempo. La IP pública la asigna el ISP (*Internet Provider Server*) a partir de las sus direcciones disponibles utilizando un servicio denominado DHCP (*Protocolo Configuración Dinámica de Host*).
2. **IP Privada.** Es la dirección IP que un equipo tiene **dentro de una red tipo LAN** (casa, empresa, oficina, etc..). Este tipo de IP's se pueden configurar y personalizar según las preferencias del administrador de red. De forma genérica las IP's de este tipo comienzan todas por la dirección 192.168.0.1, 192.168.0.2 y así sucesivamente.



- **URL:** (*Uniform Resource Locator*). Equivale a toda la dirección que se escribe en el navegador para acceder a un recurso web. Es bastante extensa y cada una de las partes de esta dirección tiene su utilidad descrita a continuación.



- **HTTP:** Es el protocolo creado para el envío y recepción de un recurso web. Utiliza el **puerto 80** de forma predeterminada en el navegador. Fue desarrollado por W3C y Internet Engineering Task Force. Es un protocolo que NO guarda el estado de sesiones anteriores, para esto se usan las cookies que almacenan información en el equipo del cliente.

Una actualización de este protocolo es **HTTPS que incluye conexión encriptada a través de los protocolos TLS y SSL** (recomendada). Actualmente la gran mayoría de páginas utilizan el protocolo `https://` entre otras cosas porque Google penaliza el posicionamiento de todas las páginas que no lo utilizan.

- **Dominio:** Determina la dirección de primer nivel que corresponde con la que se debe escribir si deseamos acceder a una página web. No se debe confundir con toda la URL para poder acceder a un determinado recurso en un site. Por ejemplo bcn.cat es un dominio de primer nivel.

- **Subdominio:** A partir de un dominio principal se pueden crear subdominios. Los subdominios se encuentran **a la izquierda del dominio**. Los hostings siempre ofrecen la posibilidad de crear multitud de subdominios de nuestro dominio contratado utilizando el *CPanel* (Panel de Control).

Es habitual que todas las páginas web comiencen por 'www' que es un **subdominio de primer nivel**, pero no es obligatorio y cada vez se usan menos. Podemos tener nuestra página web de dos formas diferentes ('http://miweb.cat' o 'http://www.miweb.cat').

Ninguna es mejor que la otra, pero por convención y siguiendo la tendencia actual en la red es preferible que contenga las 'www' (subdominio). Los motores de búsqueda también lo tienen presente cuando indexan el contenido si empieza por 'www' y ayudan al posicionamiento SEO.

Ejemplo 1: *https://misvacaciones.com* (web con dominio y sin subdominios)

Ejemplo 2: *https://www.fotos.misvacaciones.com* (web con 2 subdominios)

Ejemplo 3: *https://fotos.misvacaciones.com* (web con 1 subdominio que no es 'www')

- **Ruta, path, camino:** Determina el camino o **ruta que debe seguir el navegador** para poder acceder a un recurso del servidor. Utiliza el mismo sistema que las estructuras de árbol para almacenar información en cualquier sistema operativo. Es importante no modificar el nombre de las carpetas porque si no pueden 'romperse' multitud de enlaces dando lugar a un error del tipo '*404 Página no encontrada*' que tan poco les gusta a los motores de búsqueda.

HTTP Error 404

404 Not Found

The Web server cannot find the file or script you asked for.
Please check the URL to ensure that the path is correct.

Please contact the server's administrator if this problem persists.

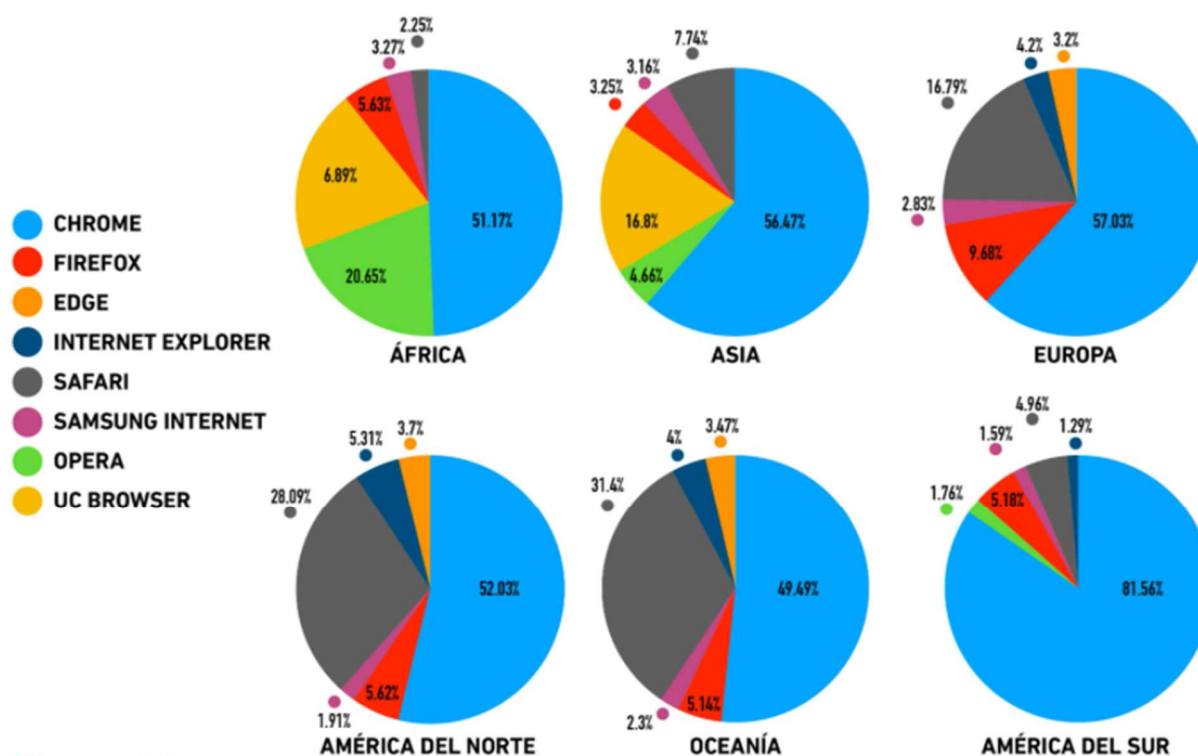
- **Recurso:** Es el **nombre del archivo o recurso que deseamos acceder**. La extensión del recurso puede variar, aunque lo más habitual es una extensión .html o php. Existen también otras extensiones que nos podemos encontrar como .jsp o .asp.

- **Extensión del dominio:** Determina **la región, país, o finalidad de la página web**. La tendencia actual es que cada vez haya más dominios que puedan ser utilizados en la red.
 - a) **Genéricos:** No son geolocalizables porque son a nivel global. Son los primeros dominios que surgieron. Los más conocidos son .gov (gubernamental), .info, .com, .net (network), .org, .mil, .edu (educación), .info, .biz (negocios), .museum, .travel (viajes), .tv.
 - b) **Territorial:** Determina la zona donde se encuentra ubicada la página web. Son plenamente geolocalizables. Por ejemplo, .es, .cat, .uk, .it, .fr, .ar, .mx, .us, .eu.
 - c) **Tercer nivel:** Son una mezcla de los dos tipos anteriores. De esta forma se puede especificar la tipología de la web y su ubicación. Por ejemplo, .gov.es, .org.cat, .edu.fr. También son denominados como subdominios.
- **FTP:** (*File transfer protocol*). Es un protocolo que se utiliza para el intercambio de archivos entre la máquina local y el servidor. Utiliza por defecto el **puerto 21**. Existen multitud de programas que realizan esta tarea como *Filezilla*, *Transmit*, *SmartFTP*, o incluso se puede configurar desde la aplicación de desarrollo web como Dreamweaver o Sublime Text (mediante plugin).
- **Directorio Web:** Es la ubicación dentro del servidor web donde se deben subir los diferentes archivos y directorios que conforman nuestra página web. El directorio puede variar de nombre, pero los más habituales son 'www', 'public', 'htdocs' o 'public_html'. El nombre del **archivo principal** al cargar la página y que buscará el servidor se denominará '*index.html*' o '*index.php*'.
- **DNS:** (*Domain Name Server*). Es un **sistema de nombres de dominio que permite asociar una dirección IP a un nombre** ya que es mucho más fácil de asimilar para las personas. Los DNS almacenan en una base de datos las direcciones IP y sus correspondientes nombres asociados. Si un servidor no dispusiera en su base de datos la correspondencia '*dirección IP-Nombre*' lo que hace es preguntar a otro servidor web para que se la indique. Existen DNS privados y públicos o gratuitos como Google (8.8.8.8). Los DNS se configuran en el equipo cliente.

Consejos antes de empezar a construir una página web

1. Los nombres de los archivos siempre **se escribirán en minúsculas y sin espacios** porque los servidores Linux, Javascript y PHP son sensibles a mayúsculas y minúsculas (case-sensitive).
2. **No utilizar caracteres especiales** (% , \$, # , i , * , @ , etc) para definir nombres de archivos. Lo más normal es que no deje guardar el archivo, pero si nos dejase es muy probable que acabe generando problemas de acceso en algún momento desde el navegador.
3. Si desarrollamos la página web en un entorno local (con un servidor propio en el equipo local) hay que tener presente que los hipervínculos con **URL absolutas o relativas** pueden funcionar en este entorno local pero pueden dejar de hacerlo en un servidor web remoto.
4. No repetir el nombre de las imágenes que se vayan utilizando en el proyecto y asignar un **nombre representativo a cada una de ellas** según el contenido de estas. Esta acción ayudará al posicionamiento de la web en los motores de búsqueda (Google, Yahoo, Bing, etc..) incluyendo a los metabuscadores (Zuula, Indeed, Duck Duck Go, Metacrawler, etc..).

USO DE LOS NAVEGADORES EN EL MERCADO POR ESPACIO GEOGRÁFICO



5. Las imágenes deben tener **extensiones cómodas para el navegador** como *.png*, *.jpg*, *.bmp*. los formatos *.webp* y *.jpeg2000* son los recomendados por los navegadores. Determinados formatos como el de Photoshop (*.psd*) o formatos profesionales (*.raw*) son formatos que pesan mucho y no son capaces de ser representados por la gran mayoría de los navegadores. El atributo *'alt'* de las imágenes deberá ser rellanado para facilitar el posicionamiento SEO.
6. **Recargar toda la página** (*'Ctrl + F5'*) cuando se suben nuevos archivos al servidor. Es posible que la web que estemos viendo se conserve en la memoria caché del equipo local y no sea la última actualización del servidor. No actualizar totalmente la web es un error muy frecuente cuando se trabaja subiendo archivos a un servidor remoto (hosting). El resultado es que se visualiza siempre la misma página, aunque se hayan realizado modificaciones.
7. **Organización de contenidos.** Si nuestro proyecto tiene pocas páginas y el volumen de información no es muy elevado se puede “improvisar” sobre la marcha la estructura de directorios del site. Como esto no es lo habitual se recomienda ser muy organizado con los diferentes directorios para poder absorber el posible crecimiento de la web.
8. Utilizar **nombres representativos en funciones, archivos y variables**. Cuando programamos es conveniente que todos los elementos tengan un nombre que sea fácil de entender y interpretar. En proyectos extensos donde se trabaja con muchos archivos y funciones puede provocar errores porque el programador no es capaz de seguir correctamente el hilo de ejecución.

En el siguiente ejemplo, para declarar el mismo procedimiento siempre será más recomendable la forma número dos que la forma uno por cuestiones de mantenimiento y escalabilidad.

- Forma 1: function ***calcular()***
- Forma 2: function ***calcularSuperficieVolumen()***

Navegadores y compatibilidad.

Los navegadores **son compatibles con la última versión de HTML** y deben saber interpretar el mayor número de etiquetas disponibles. Si un navegador no reconoce una etiqueta la ignora y el efecto que pretendía la etiqueta no queda reflejado o visualizado en la página. Es necesario **realizar actualizaciones de los navegadores para que puedan ser compatibles** para la última versión del lenguaje.

La versión HTML5 (la más reciente) es una **agrupación de diferentes tecnologías** (HTML - Estructura, CSS - Formato y Javascript - Interactividad). Es el estándar que más se utiliza y los navegadores se actualizan en función del estándar establecido por el **Consortio W3C**. Hace años este problema de incompatibilidad entre navegadores era muy frecuente. Hoy en día cualquier navegador de cierta notoriedad se actualiza frecuentemente y es capaz de interpretar el estándar HTML en cualquiera de sus versiones sin problemas.

Los navegadores actuales más reconocidos y utilizados por los usuarios son aquellos capaces de representar la última versión de HTML5 y sus tecnologías relacionadas (CSS, PHP, JS, ASP, etc...). Algunos navegadores cuando se actualizan pueden perder determinadas funcionalidades que estaban activadas. Por ejemplo, desde la versión 45 de Google Chrome no es posible ejecutar Applets de Java



Editores de lenguaje HTML

Un editor es un programa que nos permite **redactar documentos** (Word, Writer, Notepad...). Hoy en día existen un gran número de editores que permiten crear y diseñar páginas Web. Algunos de estos editores nos permiten crear páginas sin escribir prácticamente código como Dreamweaver (poco o nada recomendable). Estos editores disponen de un entorno visual WYSIWYG (lo que ves es lo que obtienes), crean automáticamente el código de las páginas y generan mucho código basura.

Algunos de los **editores profesionales de pago y gratuitos** de páginas Web más utilizados por los profesionales del diseño en el sector son:

De pago (profesionales)	Gratuitos
Sublime Text	Visual Studio Code
Dreamweaver (CS5, CS6, CC)	JetBrains Webstorm
Adobe PageMill.	Sublime Text (free version)
Komodo	Atom (personalizable)
CutePage.	Notepad ++

Para iniciarse en la creación de páginas o aplicaciones web es recomendable disponer de un buen editor, que sea personalizable para el programador, y sobre todo conocer de forma notable su funcionamiento y posibles opciones que disponga.

Escribir el código desde cero sobre las páginas web, sin ayudas ni autocompletados repercutirá de manera muy positiva en el conocimiento de las etiquetas.

Al guardar el archivo no se debe olvidar hacerlo con la extensión **htm** o **html**. Los programas como *Visual Studio Code*, *Atom*, *Notepad++* o *Sublime Text* son capaces de interpretar la gran mayoría de los lenguajes de programación existentes.

NOTA: La gran mayoría de programadores web NO son partidarios de la utilización de programas que generan código de forma automática porque se pierde cierto control sobre la página y el código que se genera se debe cargar ralentizando la página. La tendencia actual son editores sencillos de utilizar y muy personalizables (*Sublime Text*, *Atom*, *Visual Studio Code*).

Funcionamiento de las Etiquetas.

Las **etiquetas** (tags) son la base para construir una página web a nivel estructural (esqueleto). Delimitan cada uno de los elementos que componen un documento HTML. De forma genérica existen dos tipos de etiquetas, las de apertura (inicio) del elemento y las de cierre (fin) del elemento.

La **etiqueta de apertura** está delimitada por los caracteres mayor y menor **< b >**. Está compuesta por el identificador o nombre de la etiqueta, y puede contener una serie de **atributos** opcionales que permiten añadir y especificar ciertas propiedades.

Su sintaxis es: **<identificador atributo1="valor" atributo2="valor" >**

La **etiqueta de cierre** está delimitada por los caracteres mayor, menor y contra barra **< /b >**. Está compuesta por el identificador o nombre de la etiqueta, y no contiene atributos.

Su sintaxis es: **</identificador>**

El cierre de las etiquetas es obligatorio para la correcta visualización y interpretación de la página web por parte de los navegadores. La etiqueta se cierra utilizando el carácter de barra inclinada [/].

Cada uno de los elementos de la página web se encontrará entre una etiqueta de comienzo y su correspondiente etiqueta de cierre, a excepción de algunos elementos que no necesitan etiqueta de cierre, pero estos son una minoría como por ejemplo **
** (salto línea) o **<hr>** (línea horizontal).

Es posible **anidar etiquetas**, es decir, introducir etiquetas dentro de otras etiquetas. Es una práctica muy habitual que se debe dominar para construir páginas web de calidad. Al anidar etiquetas hay que seguir un orden lógico y nunca deben cruzarse etiquetas de apertura y de cierre. Por ejemplo, el siguiente código ** <u> Texto </u> ** es correcto mientras que ** <u> Texto </u>** podría ser mal interpretado por el navegador y no visualizarse correctamente.

Ej. Etiqueta sin atributos: ** Texto en negrita **

Ej. Etiqueta con atributos: **<p id="dni" class="num"> 4356723T </p>**

Ej. Etiqueta anidada con atributos: **<p id="dni" class="num"> 4356723T </p>**

Los atributos de las etiquetas son los elementos que permiten personalizar las características de las etiquetas. Hay etiquetas que NO tienen atributos, otras tienen muchos, y determinados atributos pueden ser compartidos por más de una etiqueta (*id*, *class*, etc..).

Además es posible que el usuario genere sus propios **atributos personalizados** para almacenar información dentro de la etiqueta que posteriormente será gestionada mediante algún lenguaje de programación (Javascript como normal general).

W3C recomienda utilizar etiquetas en minúsculas por estandarización, pero todos los navegadores aceptan las etiquetas en mayúsculas.

Estructura básica de una página

Las páginas Web tienen una estructura básica y común a todas ellas. El código se encuentra ubicado siempre entre las etiquetas **<html>** de apertura y cierre, y son las etiquetas **<head>** y **<body>**.

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Bienvenido
    </title>
  </head>
  <body>
    <p></p>
  </body>
</html>
```

DOCTYPE establece la versión HTML que se utiliza para la creación de la página web. Determina qué conjunto de caracteres serán utilizados para realizar la página. Esto facilita al navegador como interpretarla. La declaración más frecuente es para la versión en HTML5.

Entre las etiquetas **<html>** de apertura-cierre se encuentra el código que forma parte de la página Web. Esta etiqueta puede contener el atributo **lang** que especifica el lenguaje de la web, cada país tiene su código.

La cabecera **<head>** se utiliza para agrupar información genérica sobre la página Web y para exportar los archivos necesarios. También puede contener en su interior otras etiquetas como **<link>**, **<style>**, **<script>**, **<meta>**, **<title>**. La información de esta sección no se visualiza en la web.

La información de la etiqueta **<title>** se mostrará en la parte superior de la ventana del navegador. Ayuda a la indexación dentro de los motores de búsqueda y se utiliza para categorizar la página web.

Google, por ejemplo, le da más importancia a la etiqueta **<title>** que a la etiqueta **META Name="keywords"**.

El cuerpo **<body>** del documento contiene la información propia del documento, es decir, lo que queremos que se visualice, el texto de la página, las imágenes, los formularios, etc.

Lenguaje html

```
<html>
  <head>
    Información técnica
    para el navegador
  </head>
  <body>
    Contenido que aparecerá
    en la página web
  </body>
</html>
```

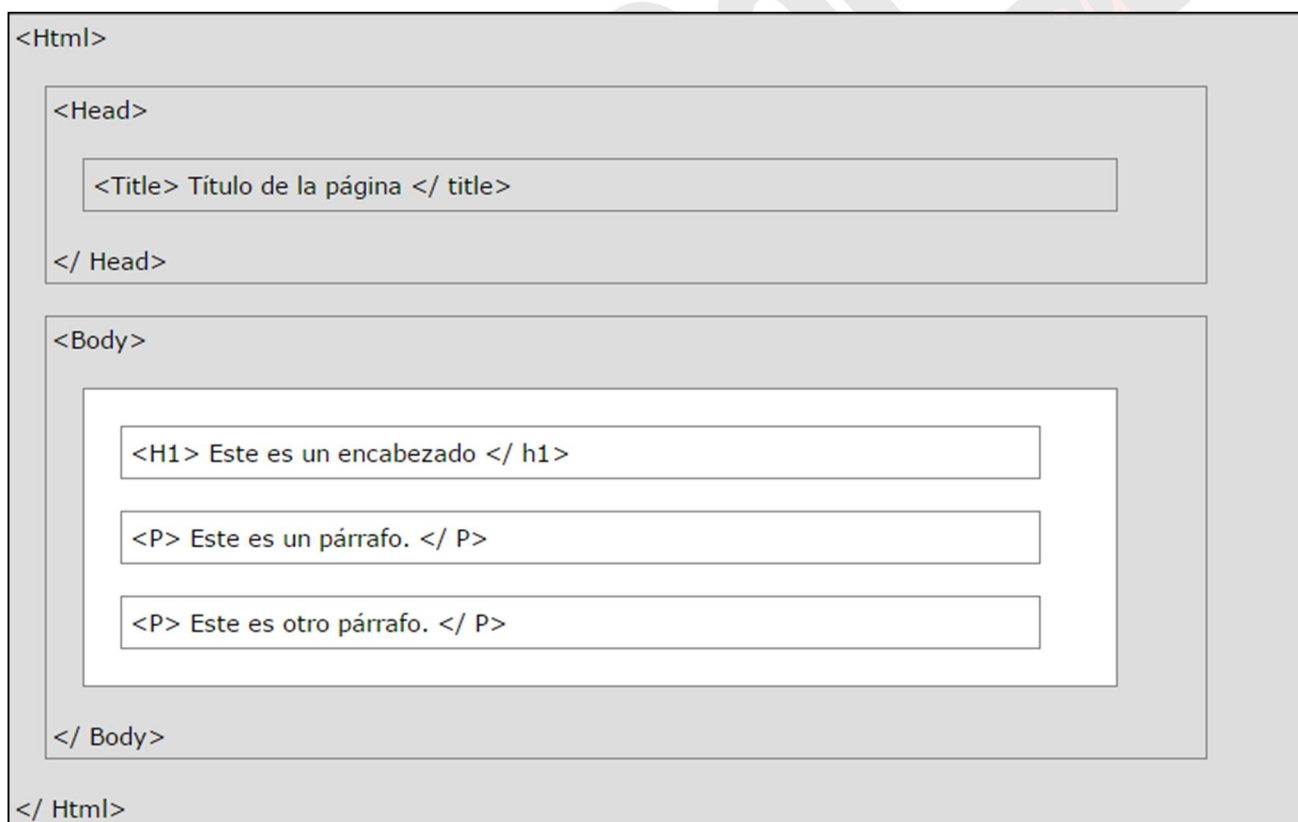
En ocasiones es necesario introducir **comentarios** dentro del código de la página Web. La finalidad es facilitar la comprensión del código introducido, tanto para nosotros como para otras personas que tengan que interpretar el código que hayamos creado.

La etiqueta que se utiliza para realizar comentarios es `<!-- Texto de comentario -->`

Ejemplo: ` Texto en negrita ` `<!-- Texto en negrita -->`

En la construcción de páginas web es habitual repetir estructuras de etiquetas en diferentes zonas de la página. A continuación, se muestra un esquema clásico de una página web.

Estructura básica de una página Web

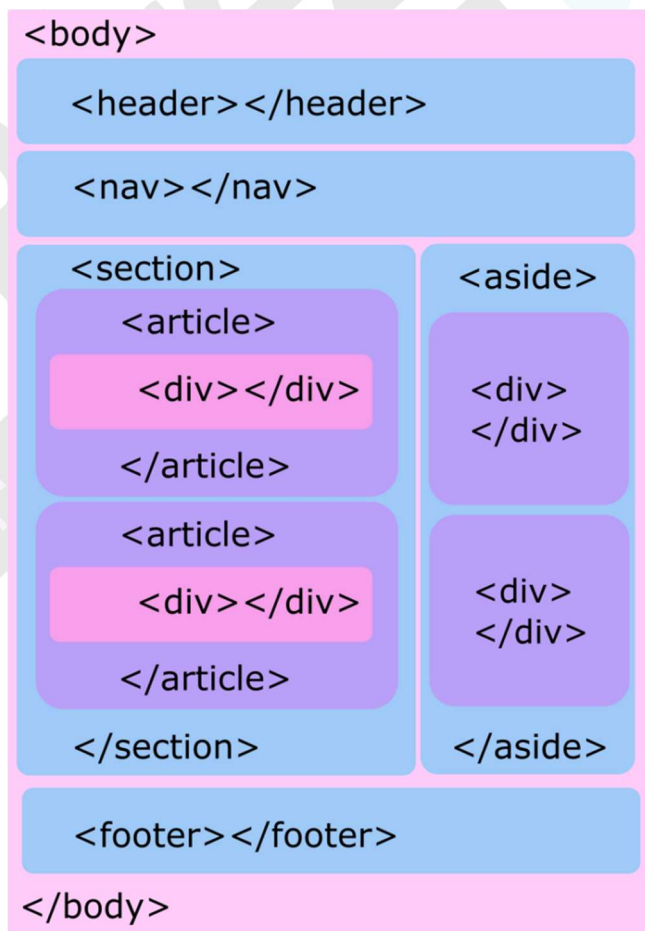
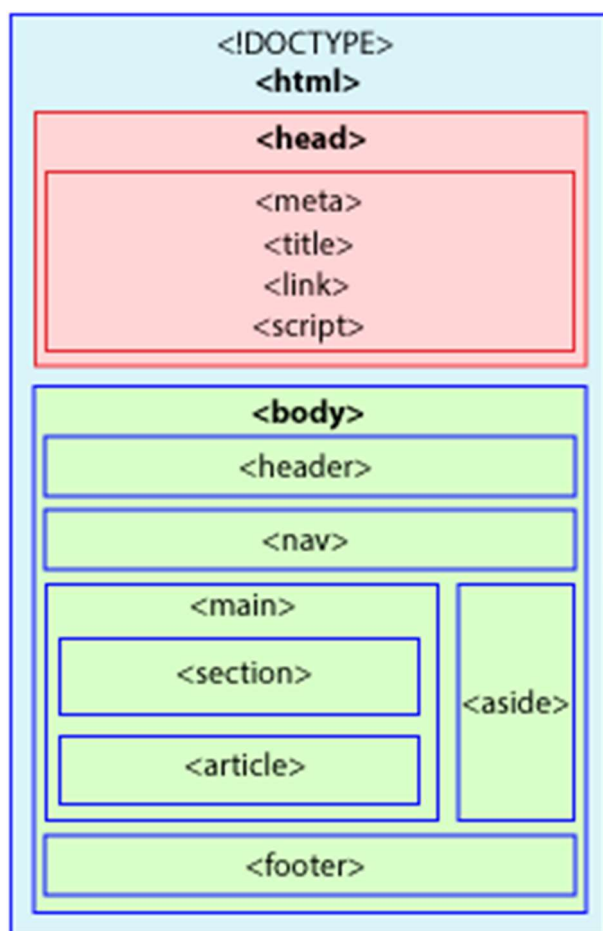


Al guardar un documento en formato web hay que asegurarse que la extensión sea HTML o HTM, y la codificación de caracteres (el alfabeto utilizado para la creación de la página) sea **UTF-8**. Esta codificación es la más estandarizada a nivel mundial y es la que **recomienda el Consorcio W3C** para la construcción de páginas web.

Las siguientes imágenes muestran un **wireframe** (borrador) clásico con la distribución de los elementos en una página web. Estas estructuras pueden cambiar su formato y en ningún caso deben seguir siempre el mismo esquema.

La imagen de la izquierda muestra la estructura de toda la web mientras que la imagen de la derecha muestra solo la estructura del cuerpo **<body>** de la página.

El nombre y la función de las etiquetas se explicarán en otros apartados pero hay que tener presente que una etiqueta es un **contenedor de información** y que en su interior puede contener texto, imágenes o más etiquetas anidadas.



HTML etiqueta <head>

La etiqueta <head> es un contenedor de metadatos (datos genéricos referentes a la web) y se ubica entre las etiquetas <html> y <body>.

Características de la etiqueta <head>:

- Almacena todos los metadatos de la página web. Los Metadatos **no se visualiza en la web** y se utilizan para **ofrecer información a los motores de búsqueda** sobre el contenido de la página. Existen un número considerable de etiquetas META que son configurables para la web. La utilización de estas etiquetas ayuda al posicionamiento de la web a través de SEO.
- Se puede definir el título de la página <title> y el conjunto de caracteres que afecta a todo el documento (UTF-8 como más habitual). Ejemplo: <meta charset="UTF-8"/> .
- Enlaza la página con las hojas de estilo CSS (formato) y con los archivos de JavaScript mediante las etiquetas <link> y <script> respectivamente.
- **Enlaza con los CDN** (*Content Delivery Networks*) que son repositorios en Internet de donde podemos obtener contenido como, por ejemplo, Font Awesome (obtención de iconos) o Google Fonts (obtención de cualquier tipo de fuente).
- A nivel más avanzado, esta etiqueta también contiene el código necesario para poder realizar la **analítica de datos** dentro de nuestra página web con Google Analytics.

Las etiquetas que se pueden utilizar como metadatos con la etiqueta <head> son:

- Elemento <title>: Define el título del documento, el título en la pestaña del navegador, el título de la pestaña cuando se añade a favoritos, y el título de la página para los motores de búsqueda.

```
/* Declaración de un título para la web dentro del <head> */  
<head>  
  <title> Bicicletas eléctricas </title>  
</head>
```

- **Elemento <style>**: Define la información que hace referencia al estilo de los elementos de la página actual. Como normal general, los estilos se encuentran ubicados en archivos externos y se accede a ellos mediante un enlace. Como recomienda W3C hay que tener separadas correctamente las tres capas (Estructura - Estilo – Interactividad) pero todavía es posible definir los estilos dentro de la misma página.

```
/* Declaración de estilo dentro del <head> */
```

```
<head>
```

```
<style>
```

```
body {background-color: blue;}
```

```
h1 {color: red;}
```

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

```
</style>
```

```
</head>
```

Esta declaración de estilo define las propiedades para las etiquetas **<body>**, **<h1>** (encabezado), y **<p>** (párrafo) exclusivamente para esta página.

- **Elemento <link>**: Se utiliza como normal general para **enlazar con hojas de estilo externas** o con los CDN's de datos. Esta etiqueta se puede utilizar tantas veces como se desee para enlazar diferentes hojas de estilo. El atributo 'rel' determina el tipo de archivo que se desea enlazar, y el atributo 'href' especifica el nombre del archivo. El uso de múltiples archivos externos en una misma página es una práctica muy común. Además, ayuda a tener modulado y separado el contenido del contenedor.

```
/* Enlaza la web con un archivo externo CSS y con un CDN */
```

```
<head>
```

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

```
<link rel="stylesheet" href="https://fonts.googleapis.com">
```

```
</head>
```

La etiqueta **<link>** se utiliza también para establecer el icono de la pestaña para la página web denominado **favicon** (*Favorite Icon*). Los navegadores actuales adaptan la imagen seleccionada por el usuario escalándola como favicon pero no siempre da buenos resultados. Las características de un favicon son:

- La extensión del archivo más recomendable es '.ico', pero existen otros posibles formatos como .png, .jpg, .bmp, gif, .icns (Apple) que son aceptados.
- Debe ser un cuadrado y sus medidas son 16x16 o 32x32. Algunas extensiones de imagen aceptan más tipos de resoluciones 48x48, 36x36, 64x64.
- Se pueden utilizar píxeles transparentes de fondo (cuadrícula gris-blanca).
- Es posible introducir un favicon dinámico en nuestro site pero este deberá ser animado a través de código de programación (Javascript).
- Existen paginas online que permiten crearlas como www.faviconer.com



/* Introduce un favicon en la página web y el navegador seleccionará la imagen más conveniente de las 3 opciones */

<head>

```
<link rel="icon" href="favicon.ico" type="image/x-icon" sizes="16x16">
```

```
<link rel="icon" href="favicon.png" type="image/png" sizes="64x64">
```

```
<link rel="icon" href="favicon.gif" type="image/gif" sizes="32x32">
```

</head>

- Elemento `<script>`: Se utiliza para definir o **enlazar con archivos JavaScript del lado cliente**. Las rutinas Javascript también se suelen agrupar en archivos externos (biblioteca de métodos) y las páginas web acceden a ellos mediante la etiqueta `<link>`.

```
// Enlaza la web con un archive externo JS de funciones
```

```
<script src="/js/myScript1.js"></script>
```

```
// Declara la función (miFuncion) dentro del <head>
```

```
// La función captura el identificador de un párrafo y sustituye su  
contenido mediante la orden 'innerHTML'
```

```
<head>
```

```
  <script>
```

```
    function miFuncion {
```

```
      document.getElementById("demo").innerHTML="Hola JavaScript!";
```

```
    }
```

```
  </script>
```

```
</head>
```

- Elemento `<base>`: Especifica la URL de destino por defecto para todas las URL's relativas en una página. La etiqueta solo puede aparecer una única vez dentro del documento web y debe situarse a continuación de la etiqueta de encabezado de documento `<head>`, siendo así la primera etiqueta del encabezado. No tiene etiqueta de cierre.

```
<head>
```

```
  <base href="http://www.mypage.com/images/" target="_blank">
```

```
</head>
```

NOTA: Omitir la etiqueta `<head>` ha sido una práctica generalizada en la construcción de páginas web. El consorcio W3C recomienda su uso para evitar problemas con los navegadores y con los lenguajes de marcas más estrictos con el código como XHTML. Además, es recomendable su uso por cuestiones de posicionamiento web.

La etiqueta `<meta>` o meta-tags

Se utiliza para añadir información sobre la página. La etiqueta `<head>` es un contenedor de metadatos que no se visualizan en la web pero que afectan a toda la página. Esta información adicional será utilizada por los buscadores que consultan la información contenida en la etiqueta `<meta>`.

La etiqueta `<meta>` no necesita etiqueta de cierre. Para cada etiqueta `<meta>` solo es posible indicar un tipo de información y su valor, pero es posible insertar varias etiquetas `<meta>` en un mismo documento.

La etiqueta `<meta>` debe estar entre las etiquetas `<head>` y `</head>`. Esta etiqueta nos ayuda al posicionamiento mediante SEO de nuestra web.

SEO: Es toda práctica o acción realizada sobre nuestra página web que nos ayudará a estar bien posicionados en los motores de búsqueda (Google y otros). Realizar SEO en la web **es gratuito**, en cambio si queremos salir en las primeras páginas de Google al escribir determinadas palabras en el buscador es SEM. El sistema de posicionamiento SEM es de pago a través de Google Adwords u otras formas de publicidad.

Ejemplo 1:

```
<meta name="author" content="Victor"/>
```

Autor de la página.

```
<meta name="description" content="Bicicletas alquiler"/>
```

Descripción del contenido de la página. La “description” se mostrará en los resultados de búsqueda de Google sino será Google quien decida el texto que saldrá en el buscador. Google coge el primer párrafo que considere representativo según sus criterios. Entre 50 y 70 palabras.

```
<meta name="keywords" content="bicicleta bici btt ciclismo montaña "/>
```

Son aproximadamente 20 palabras máximo. Se ha hecho un abuso desmesurado de este atributo de palabras clave (*keyword stuffing*) y los buscadores no le prestan atención.

```
<meta name="language" content="spanish" />
```

Se utiliza para determinar el idioma en el que se ha escrito el contenido de la web. Esta etiqueta está obsoleta en la actualidad y se recomienda en HTML5 el uso del atributo '**lang**'. Si non se introduce este atributo, por defecto el navegador intentará o solicitará traducir todas las páginas que no coincidan con el idioma del navegador.

```
<html lang="es-ES">
```

```
<meta name="Copyright" content="El periódico"/>
```

Empresa o organismo que tiene la propiedad intelectual del contenido de la página.

```
<meta name="robots" content=" index | follow" />
```

El valor “**content**” puede tener varias combinaciones

Index, Follow – Permite la indexación y rastreo de la página y es el valor por defecto.

Prescindir de la etiqueta meta robots es lo mismo que utilizarla con esta configuración.

NoIndex, Follow – Evita la indexación, pero permite el rastreo. Es la configuración ideal cuando no quieres que una página aparezca en los resultados del buscador.

Index, NoFollow – Permite la indexación, pero evita el rastreo.

NoIndex, NoFollow – Evita la indexación y el rastreo.

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

El **viewport** es el dispositivo de visualización. Esta Meta se utiliza en los sitios que cuentan una versión para móviles del tipo adaptable o responsive design.

width=device-width – El tamaño de la página debe ser como el del dispositivo en que se muestra la página.

initial-scale=1.0 – La página debe mostrarse inicialmente tan grande como permita la pantalla del dispositivo.


```
<meta name="revisit-after" content="14 days"/>
```

Se utiliza para indicar a los buscadores el promedio con el que modificamos la página para que vuelva a visitarla e indexar los cambios. El contenido o valor del atributo en este caso debe estar en inglés (days, month, year, week).

```
<meta name="geo.region" content="ES-VC" />
```

```
<meta name="geo.placename" content="Valencia" />
```

```
<meta name="geo.position" content="39.470239;-0.376805" />
```

Esta etiqueta permite geolocalizar la página web. Google no le presta demasiada atención a esta etiqueta, en cambio Bing sí. Cada motor de búsqueda tiene sus criterios de selección.

La etiqueta **<meta>** también se utiliza para indicarle al navegador alguna información o alguna acción que debe realizar. En este caso se utiliza el atributo **http-equiv**, en lugar del atributo 'name'. Una de las prácticas más frecuentes en este caso consiste en refrescar la pantalla cada cierto tiempo. El atributo 'content', en este caso, especificará el tiempo de actualización del site.

Ejemplo 2:

```
<meta http-equiv="Refresh" content="30"; URL="http://www.mipagina.es"/>
```

Por ejemplo, las páginas de apuestas deportivas deben estar actualizándose constantemente para mostrar los resultados en tiempo real, otras en cambio no son necesario.

Otras etiquetas META que siguen la misma estructura y que realizan otro tipo de acciones son:

- Codificación del idioma europeo occidental. Nos permite visualizar correctamente los caracteres especiales de la web (ñ, ny, ç, etc..).

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

```
<meta http-equiv="Content-Type" content="text/html" charset="iso-8859-1"/>
```

- Idioma de la página. En este caso el valor '**es-es**' indica español de España

```
<meta http-equiv="Content-Language" content="es-es"/>
```

- El valor '**Revisit**' indica al robot que vuelva a pasar a visitar la web pasados 'X' días.

```
<meta name="Revisit" content="X days"/> siendo X el número de días.
```

- El valor '**Rating**' nos permite determinar el público al que se dirige nuestra página. Los posibles valores de rating son:

General | Mature | Save for kids | Restricted | 14 years.

```
<meta name="rating" content="general"/>
```

```
<meta name="rating" content="safe for kids"/>
```

Etiquetas en HTML

La etiqueta **<body>** contiene todos los elementos de nuestra web. Dispone de una serie de atributos heredados de los inicios del html que actualmente están en desuso (bgcolor, link, text, etc..) entre otras cosas porque muchos navegadores no los soportan. Actualmente cualquier configuración de estilo se realiza a través de las hojas de estilo CSS.

HTML permite que **las etiquetas puedan estar anidadas**, es decir, unas dentro de otras. Esto es una práctica habitual en la creación de páginas Web. Anidar etiquetas obliga a mantener un orden en el posicionamiento de las etiquetas que engloban a un mismo elemento.

Ejemplo: **<i> ** Texto en negrita y cursiva ** </i>**

Este código y sus etiquetas tienen el orden correcto en la construcción de la página web. Si intercambiamos las etiquetas, los navegadores lo podrían interpretar de forma errónea y el documento puede visualizarse incorrectamente. Los lenguajes de etiquetas más estrictos como XHTML no lo permiten en la estructuración de las páginas.

En el estándar HTML5 no es necesario poner las dobles comillas (") a los valores de los atributos. W3C recomienda utilizar comillas dobles porque no todos los lenguajes de etiquetas soportan la mejora establecida por HTML5.

Es recomendable utilizar minúsculas en la creación de código y etiquetas. Igual que en el caso anterior algunos lenguajes pueden no soportar estructuras o etiquetas escritas en mayúsculas y minúsculas.

Existen una serie de caracteres especiales que pueden ser mal interpretados por los navegadores y no nos permitirán visualizar la información de la web. Para evitar este problema, el estándar de HTML prevé una serie de representaciones para poder ver a través de la Web caracteres especiales y/o algunos exclusivos del código como por ejemplo los caracteres de apertura y cierre de etiqueta **< >**, los acentos en las palabras, o la letra ñ.

Todos los caracteres, incluso los que son símbolos o formas excepcionales, tienen su representación mediante el sistema de codificación internacional. El sistema está dividido en dos tipos de codificaciones (son los mismo pero diferente representación): **ISO 8859-1 Caracteres** y **ISO 8859-1 Símbolos**.

Los **códigos especiales** (Entidades) en codificación universal se pueden escribir de dos formas (con número HTML o con nombre HTML). En la medida de lo posible se debe intentar evitar introducir este tipo de códigos dentro de la página o aplicación web aunque no siempre es posible.

- **Número HTML**: Esta codificación utiliza el número ASCII precedido del ampersand (&) y la almohadilla (#). Esta codificación tiene más caracteres implementados que mediante la codificación mediante Nombre HTML. Un ejemplo: **€** (signo del euro).
- **Nombre HTML**: Es una codificación similar a la anterior. Ejemplo: ** ** (espacio).

La siguiente tabla muestra un listado con los caracteres especiales (*no están definidos todos*) más utilizados teniendo en cuenta las dos codificaciones ISO.

Resultado	Descripción	Por Nombre	Por Número
	Espacio en blanco	 	
<	Menor que	<	<
>	Mayor que	>	>
&	Y	&	&
"	Comillas	"	"
¡	Apertura signo de exclamación	¡	¡
¿	Apertura signo de interrogación	¿	¿
®	Marca registrada	®	®
©	Derecho de autor	©	©
€	Euro	€	€
á	a minúscula con acento	á	á
é	e minúscula con acento	é	é
í	i minúscula con acento	í	í
ó	o minúscula con acento	ó	ó
ú	u minúscula con acento	ú	ú

ñ	ñ minúscula	ñ	ñ
ü	u minúscula con diéresis	ü	ü
Á	A mayúscula con acento	Á	Á
É	E mayúscula con acento	É	É
Í	I mayúscula con acento	Í	Í
Ó	O mayúscula con acento	Ó	Ó
Ú	U mayúscula con acento	Ú	Ú
Ñ	Ñ mayúscula	Ñ	Ñ
Ü	U mayúscula con diéresis	Ü	Ü
°	Signo de grado.	°	°
±	Signo de más-menos.	±	±
²	Superíndice dos.	²	²
³	Superíndice tres.	³	³
´	Acento agudo.	´	´
μ	Signo de micro.	µ	µ
¶	Signo de calderón.	¶	¶
·	Punto centrado.	·	·
¸	Cedilla.	¸	¸
¹	Superíndice 1.	¹	¹
º	Indicador ordinal masculino.	º	º
»	Signo de comillas francesas de cierre.	»	»
¼	Fracción vulgar de un cuarto.	¼	¼
½	Fracción vulgar de un medio.	½	½
¾	Fracción vulgar de tres cuartos.	¾	¾
¿	Signo de interrogación abierta.	¿	¿
×	Signo de multiplicación.	×	×
÷	Signo de división.	÷	÷

En las diferentes evoluciones de HTML el conjunto de caracteres utilizados ha ido variando.

Cronológicamente la evolución de los diferentes conjuntos de caracteres ha sido.

- **ASCII** fue el primer **estándar de codificación de caracteres** (también llamado juego de caracteres). ASCII define **127 caracteres alfanuméricos** diferentes que podrían ser utilizados en Internet o en otras áreas dentro de la programación web como los núm. (0-9), letras inglesas (A-Z), y algunos caracteres especiales como! \$ + - () @ <>.
- **ANSI** (Windows-1252) fue el conjunto original de caracteres de Microsoft Windows, con soporte para **256 códigos de caracteres** diferentes.
- **ISO-8859-1** fue el juego de caracteres predeterminado para HTML 4. Este conjunto de caracteres también apoyó 256 códigos de caracteres diferentes.
- Debido ANSI e ISO-8859-1 eran tan limitada, la codificación de caracteres por defecto fue cambiado a **UTF-8** en HTML5.
- **UTF-8 (Unicode)** cubre casi todos los caracteres y símbolos en el mundo, **hasta llegar a 10.000**. La versión de HTML4 también soporta la codificación UTF-8.

Para visualizar la página correctamente un navegador debe conocer el **conjunto de caracteres con el que se ha desarrollado la página**. Esto se especifica con la etiqueta **<meta>**.

Para HTML 4.01:

```
<meta http-equiv="Content-Type" content="text/html" charset="ISO-8859-1">
```

Para HTML 5:

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


Las etiquetas HTML se utilizan para crear la estructura de la página web (esqueleto). Las etiquetas además de realizar pequeñas acciones también ayudan a identificar el contenido que albergan. Por ejemplo, si queremos añadir un párrafo utilizaremos la etiqueta **<p>**, en cambio si nuestra intención es crear un título lo realizaremos con una etiqueta de encabezado **<h>**.

Las etiquetas son **contenedores de información** y aunque de forma particular puedan realizar alguna acción o modificar el estilo del elemento que contiene, la característica mas significativa es que son "cajas" que almacenan datos u otras cajas en su interior. Es el concepto que más adelante se hará referencia como 'Model Box'.

Es conveniente utilizar la etiqueta adecuada para según el tipo de contenido que deseamos mostrar. No todas las etiquetas son válidas para todos los contenidos. Por ejemplo, si deseamos insertar una imagen en la web se realizará siempre con la etiqueta **** o **<picture>**.

Entre las etiquetas mas significativas para empezar (no están todas) podemos destacar:

- **<p> Párrafo**: Esta etiqueta se utiliza para delimitar donde comienza y donde finaliza cada uno de los párrafos. En una misma página se puede utilizar la etiqueta tantas veces como sea necesario. Es una de las etiquetas de mayor uso en la construcción de páginas web. El estilo que se asignará al texto interior se realizará siempre a través de hojas de estilo en CSS.

Esta etiqueta lleva incorporada una acción automática que deja un **espacio adicional** respecto al párrafo anterior y posterior.

En el ejemplo1 se crea un párrafo y se asigna al texto el estilo directamente dentro de la etiqueta. Es una opción válida que inicialmente se utiliza para entender el funcionamiento de las hojas de estilo pero que en un nivel más avanzado se asignaría de otra forma.

En el ejemplo 2 se construye un párrafo con un atributo **'id'** (muy utilizado para identificar elementos HTML) al cual se le asigna un estilo (**'alineación'**) que previamente ha sido creado en una hoja de estilos CSS.

Ej 1: **<p style="color:blue">** Texto en color azul **</p>**

Ej 2: **<p id="alineacion">** Párrafo centrado **</p>**

- **<div>** Contenedor: Esta etiqueta es un contenedor que se utiliza para estructurar la página web en diferentes secciones. En su interior suele contener otras etiquetas y no realiza ninguna acción de forma automática cuando se inserta en la página. Se utiliza básicamente para posicionar otros elementos en un área específica. De la misma forma que la etiqueta **<p>** el estilo se asigna a través de hojas de estilo CSS.

Ej.: `<div id='caja_parrafos'>`

`<p> Texto 1 </p>`

`<p> Texto 2 </p>`

`</div>`

- ****: Negrita: **Texto en negrita**.

` Texto en negrita ` // HTML para negrita

`b { font-weight: bold; }` // CSS para negrita

- **<i>**: Cursiva: *Texto en cursiva*.

`<i> Texto en cursiva </i>` // HTML para cursiva

`i { font-style: italic; }` // CSS para cursiva

- **<u>**: Subrayado: Texto en subrayado.

`<u> Texto en negrita </u>` // HTML para subrayado

`u { text-decoration: underline; }` // CSS para subrayado

- ****: Tachado: ~~Texto tachado~~. Provoca un efecto de tachado en el texto. En CSS tiene dos atributos 'cite' y 'datetime' que determinan el momento que se suprimió.

` Tachado ` // HTML para tachado

`del { text-decoration: line-through; }` // CSS para tachado

- **<ins>: Insertado:** El efecto provocado es el mismo que el subrayado. Su ubicación normalmente es a continuación de la etiqueta .

```
<del> Tachado </del> // HTML para insertar
```

```
del { background-color:yellow; } // CSS para resaltado
```

Mi color favorito es ~~rojo~~ **amarillo**.

- **:** Es una **etiqueta comodín** que se utiliza como contenedor dentro de la web. Por si sola no realiza ninguna acción y la diferencia respecto a una etiqueta <div> es la forma de interactuar dentro de la web. Esta etiqueta se caracteriza por ser un elemento en línea mientras que la etiqueta <div> trabaja como un elemento en bloque. (más adelante se explicarán detenidamente estos dos conceptos).

```
<p> Coche <span style="color:blue"> azul </span> </p>
```

- **<mark>:** Define un texto como marcado o destacado (resaltado). **Texto destacado**. Como en la mayoría de las etiquetas el estilo será aplicado a través de CSS.

```
<Mark> Texto destacado <Mark> // HTML para color de fondo
```

```
mark { background-color:yellow; } // CSS para color de fondo
```

- **
: Salto de línea:** Esta etiqueta permite realizar un salto de línea. A diferencia de otras etiquetas esta no lleva la etiqueta de cierre. En la última especificación de HTML5 se recomienda también cerrar estas etiquetas para mantener la coherencia estructural de la web. Hoy en día con el tipo de estructuras web que se realizan es una **etiqueta poco utilizada** ya que es difícil integrarla en los modelos actuales.

Ej.: Creación de páginas Web **
** Código HTML. **
** (opcional)

- **<pre>: Texto preformateado:** Se utiliza para asegurarnos que el texto aparezca en el navegador tal cual ha sido insertado dentro del código. Es algo restrictiva porque no permite introducir otras etiquetas anidadas en su interior como **** (para incluir imágenes), **<object>** (para incluir objetos como animaciones), **<big>**, **<small>**, **<sub>** ni **<sup>**. Es una etiqueta que por sus propiedades no se utiliza demasiado.

Ej.: **<pre>** Este texto se mostrará en la Web
tal y como lo estamos escribiendo
en este momento escalándolo
hacia la derecha del documento
de forma progresiva. **</pre>**

- **<hr>: Separador de sección:** El elemento que suele utilizarse para separar secciones dentro de una misma página es la regla horizontal (actualmente se realiza de otra forma). Para insertar una regla horizontal hay que insertar esta etiqueta que se caracteriza por no precisar etiqueta de cierre. Esta etiqueta dispone de algunos atributos propios para su configuración.

- **Align:** Alineación de la regla dentro de la página (left, right, center).
- **Width:** Ancho de la regla. Este número es un porcentaje.
- **Size:** Alto de la regla.
- **Noshade:** Elimina el sombreado de la regla.

Ej.: Texto por encima de la línea .

```
<hr align="left" width="90%" size="5" noshade >
```

Texto por debajo línea.

- **: Destacado:** El efecto es el mismo que la negrita, pero el significado es "importante".

```
<strong> Texto destacado </strong> // HTML para destacado.
```

```
strong { font-weight: bold; } // CSS para negrita
```

- **<blockquote>** Sangrado de texto: Deja un margen a la izquierda de la página Web. El efecto cada vez que se introduce esta etiqueta es la misma que pulsar la tecla Tabulador. El atributo '*cite*' especifica la fuente de una cita y no produce ningún cambio visible. El sangrado o espaciado también se puede realizar a través de reglas CSS.

Ej.: `<blockquote cite="http://www.worldwildlife.org ">`

Texto tabulado.

`</blockquote>`

- ****: Énfasis: El texto se muestra en cursiva y marcado como "importante" para buscadores.

`Texto enfatizado ` // HTML para texto enfatizado

`em { font-style: itálica; }` // CSS para texto importante

- **<sub>**: Subíndice: Sitúa el texto como ^{subíndice} del texto, por encima.

`_{Texto destacado}` // HTML para subíndice.

`sub { vertical-align: sub; }` // CSS para posicionar zona baja línea

`font-size: smaller; }` // CSS para reducir el tamaño

- **<sup>**: Superíndice: Sitúa el texto como ^{superíndice} del texto, por debajo.

`^{Texto destacado}` // HTML para superíndice.

`sup { vertical-align: sub; }` // CSS para posicionar zona alta línea

`font-size: smaller; }` // CSS para reducir el tamaño

- **<small>**: Reduce el tamaño de la fuente, define el texto mas pequeño.

`<small> Texto destacado </small>` // HTML para reducir tamaño.

`small { font-size: smaller; }` // CSS para reducir tamaño texto

- **<dfn>**: Aplica y marca un texto como definición. El efecto es el mismo que la cursiva. Dispone de un atributo 'title' que muestra un tooltip (globo informativo).

```
<dfn title="HyperText Markup Language"> HTML </dfn>

dfn { font-style: oblique; } // CSS para cursiva
```

- **<abbr>**: Esta etiqueta visualmente no muestra nada diferente en la web, pero a nivel de código queda marcado como acrónimo para ofrecer información a los motores de búsqueda. Son básicamente etiquetas que informan del contenido de la web para una mejor indexación del contenido en los buscadores. La etiqueta se visualiza con un subrayado punteado.

```
<abbr title="HyperText Markup Language"> HTML </abbr>

abbr { font-style: underline dotted black; } // CSS para cursiva
```

- **<address>**: Define la información de contacto o autor para un documento web o artículo. Referencia direcciones postales y se utiliza sobre todo a nivel semántico. La información se muestra en cursiva y los navegadores añaden un salto de línea antes y después del elemento.

```
<address>
  Written by John Doe. <br>
  Visit us at: <br>
  Example.com <br>
  Box 564, Disneyland <br>
  USA
</address>
```

- **<cite>**: Define la información de cita para un elemento del documento. La información se muestra en cursiva (pero sin las dobles comillas como la etiqueta <q>) y información adicional para los motores de búsqueda.

```
<p> <cite> El grito </cite> por Edward Munch 1893 </p>
```


- **<q>**: Esta etiqueta define una cita breve sobre un texto. Visualmente el texto se muestra entre comillas dobles en la web y a nivel de código queda marcado como texto “importante” para el posicionamiento en buscadores.

`<p> La ONU es: <q> Organización Naciones Unidas </q> </p>`

- **<bdo>**: Se utiliza para anular la dirección del texto actual y escribirlo en orden inverso si el idioma lo requiere. Para modificar la dirección del texto dispone del atributo 'dir' que permite especificar la dirección (rtl -> right to left).

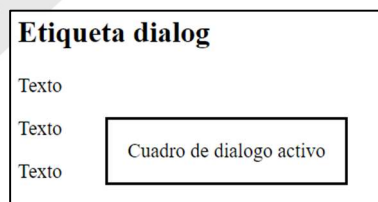
`<bdo dir="rtl"> Dirección escriptura invertida </bdo>`

- **<time>**: Esta etiqueta está pensada para incluir fechas con un formato legible para las máquinas. En su interior dispone del atributo 'datetime' que podrá adoptar diferentes formatos.

`<time datetime="01/08/2013"> 1 de Agosto del 2013 </time>`

- **<dialog>**: Esta etiqueta define un cuadro de diálogo o una subventana. Facilita la creación de cuadros de diálogo emergentes y modales en una página web. Dispone del atributo 'open' que determinará si el elemento de diálogo estará activo o no.

`<dialog open> Cuadro de diálogo </ dialog>`



- **<base>** Especifica la URL base para todas las URL relativas en un documento. Es preciso indicar los atributos 'href' (enlace) y 'target' (destino). Solo puede haber un único elemento base por documento y este debe estar dentro de las etiquetas **<head>**.

`<base href="https://miweb.com" target="_blank">`

- **<noscript>**: Esta etiqueta se utiliza para informar que el navegador no es capaz de interpretar el código Javascript de la página mostrando un mensaje informativo.

```
<noscript> El navegador no soporta Javascript </noscript>
```

- **<template>**: Se utiliza para ocultar contenido HTML al usuario cuando se carga la página. Es útil cuando se quiere mostrar el mismo código HTML una y otra vez. Para mostrar el contenido se deberá realizar a través de Javascript. Existen también otras formas de mostrar u ocultar las diferentes etiquetas en un documento HTML sin tener que utilizar obligatoriamente esta.

```
<template>  
    <h1> Contenido </h1>  
</template>
```

- **<canvas>**: Es una etiqueta que se utiliza como contenedor para dibujar gráficos a través de código Javascript. Por si sola no realiza ningún tipo de acción.

```
<canvas href="midibujo"> </canvas>
```

- **<title>**: Esta etiqueta se utiliza para especificar el título de la página web. Será visible en la barra superior del navegador. Siempre se debe introducir dentro de la etiqueta <head> y su inclusión es muy importante para los motores de búsqueda en temas de posicionamiento.

```
<title href="midibujo"> </title>
```

En determinadas ocasiones es necesario escribir un texto como si fuese en “código computadora”, es decir, siguiendo la misma tipografía de letra. HTML5 ha establecido tres etiquetas para “simular” código máquina de entrada de teclado, de salida de computadora, código de ordenador, y definición de variables. Es útil cuando deseamos categorizar contenido para que lo tengan presente los motores de búsqueda.

- **<code>**: Es para definir código de programación dentro de una página web. Se usa en conjunto con la etiqueta <pre> para presentar código de programación en formato original.

```
<code> document.getElementById("demo").innerHTML = x; </code>
```

- **<kbd>** Es para definir código de entrada mediante teclado.

```
<kbd> Archivo | Abrir </kbd>
```

```
kbd { font-style: monospace; } // CSS para cursiva
```

- **<samp>** Es para definir código salida de la computadora.

```
<samp> demo.exam.com login: Apr 12 09:10:17 </samp>
```

- **<var>** Se utiliza para definir una variable en una expresión matemática o en el contexto de los lenguajes de programación. Una página "normal" no tendría porque utilizar esta etiqueta.

```
<var> let x = 50; </var>
```

Muchas etiquetas HTML han evolucionado, otras han desaparecido, y algunas se conservan por compatibilidad con versiones anteriores del lenguaje pero se encuentran en proceso de desaparición y se consideran obsoletas. Debido a que muchas páginas aun conservan estas etiquetas se comentan a continuación las más relevantes para saber que nos la podemos encontrar:

- **** (*-- No soportada en HTML5 --*). Formateo de texto: Las propiedades del texto se pueden modificar mediante esta etiqueta con los atributos que incorpora. W3C no recomienda utilizar esta etiqueta puesto que cualquier modificación sobre el formato del texto se realizará con las hojas de estilo CSS. La etiqueta sigue vigente aunque el estándar la considera obsoleta.
- **<basefont>**: (*-- No soportada en HTML5 --*). Fuente base: Esta etiqueta es muy similar a la etiqueta . A diferencia de ésta se utiliza para definir una fuente para todo el documento. Se debe insertar tras la etiqueta <body>. En caso de coincidir con la etiqueta el navegador por orden de lectura hará prevalecer siempre la última etiqueta que haya leído.
- **<big>**: (*-- No soportada en HTML5 --*). Aumenta el tamaño de la fuente.
- **<strike>**: (*-- No soportada en HTML5 --*). Realiza un tachado en la fuente.
- **<center>**: (*--No soportada en HTML5--*). Centrado: Esta etiqueta permite centrar el texto o párrafo dentro de la página Web. Actualmente esta acción se realiza mediante CSS. Los navegadores aun reconocen y interpretan correctamente la etiqueta.
- **<acronym>**: (*--No soportada en HTML5--*). Realiza la misma función que la etiqueta **<abbr>** y básicamente es una actualización de nombre.
- **<frame>**: (*--No soportada en HTML5--*). Utilizada para dividir la página en diferentes áreas de trabajo (1 por frame). Esta técnica de distribuir las páginas por frames ha sido sustituida por las nuevas formas de maquetar.
- **<menu>**: (*--No soportada en HTML5--*). Se utilizaba para crear listas de elementos. Actualmente esta acción se realiza a través de las etiquetas **** y ****.
- **<applet>**: (*--No soportada en HTML5--*). Utilizada para introducir applets de Java (pequeñas aplicaciones). Actualmente la inserción de determinados elementos externos se puede realizar, entre otras, con las etiquetas **<embed>** y **<object>**.

Atributos globales

Los atributos son aquellas propiedades que se asignan a las etiquetas para que procedan de una forma específica. Por ejemplo, el atributo **'src'** determina el origen de un recurso. Por tanto, los atributos modifican ligeramente el comportamiento de la etiqueta que lo contiene.

Algunos atributos pueden ser utilizados por muchas etiquetas (**'id'** o **'class'**) y otros son exclusivos de unas determinadas etiquetas (**'href'** -> etiqueta **<a>**). A continuación se muestra un listado de los atributos de uso mas frecuente (no están todos).

Atributo	Descripción
href	Establece un enlace en un texto.
name	Establece un ancla para un enlace.
alt	Texto alternativo de una imagen si no se puede visualizar.
title	Como atributo es un mensaje de ayuda al ponerse sobre el texto.
target	Establece el destino del vínculo: _blank _self _top _parent .
link	Modifica el color del vínculo antes de ser activado o pulsado.
size	Tamaño de la fuente.
accesskey	Combinación de teclas para activar el elemento.
id	Especifica un identificador de la etiqueta que la hace única (DNI).
class	Asigna uno o varios estilos personalizados.
width	Modifica la anchura del objeto.
height	Modifica la altura del objeto.
translate	Determina si el texto de la etiqueta se puede traducir
contenteditable	Permite modificar el contenido de un elemento.
src	Establece que imagen se va a insertar, el archivo de origen.

Colores en HTML

Los colores en HTML se representan normalmente mediante un número hexadecimal. En la actualidad cualquier navegador es capaz de representar ±16 millones de colores (24 bits – color verdadero porque la policromía se acerca al ojo humano) pero aun existen dispositivos actuales que pueden basarse en la paleta de colores seguros (216 colores).

El número **decimal** utiliza valores del 0 al 9 (diez combinaciones disponibles) mientras que un número **hexadecimal** utiliza del 0 al 9 y de la letra A a la letra F (dieciséis valores distintos). Se utilizan números hexadecimales porque el número de combinaciones y posibilidades es mayor que utilizando valores decimales.

	Equivalencias entre valores Decimales y Hexadecimales															
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Los colores en HTML se pueden especificar de 6 formas diferentes. Aunque el sistema hexadecimal es el más utilizado cualquiera de los otros sistemas es valido para indicar el color. Cuando trabajamos con el editor es posible ir cambiando la representación del color en diferentes formatos.

1. Mediante sistema RGB: Es uno de los sistemas más utilizados debido a su facilidad de componer uno número muy elevado de colores. Se basa en los 3 colores (red, green, blue) especificando un valor para cada uno de ellos que siempre estará comprendido entre 0 y 255.

```
<h2 style="background-color: rgb( 240, 230, 55 )">
```

2. Mediante valor Hexadecimal: Se utilizan valores hexadecimales para representar los valores que en RGB se pasan como decimales.

```
<h2 style="background-color: #FF002A">
```


3. Mediante nombre directo: Son 140 nombres que representan los colores mas populares. Es un sistema muy limitado y siempre hay la posibilidad de no escribir correctamente el color.

```
<h2 style="background-color: purple">
```

4. Sistema HSL: Sistema inglés que representa los colores a partir del matiz, saturación y luminosidad. El matiz es un valor de 0 a 360, y la saturación y luminosidad un valor expresado en porcentaje entre 0% y 100%.



```
<h2 style="background-color: hsl(180, 100%, 7%);">
```

5. Mediante sistema RGBA: Es igual que el sistema RGB pero en este caso se añade el valor **Alpha** (transparencia) que será un valor siempre comprendido entre 0 transparente y 1 opaco.

```
<h2 style="background-color: rgba( 240, 230, 55, 0.4) ">
```

6. Sistema HSLA: Sistema inglés que representa los colores a partir del matiz, saturación y luminosidad. En este caso se añade el valor **Alpha** (transparencia) que será un valor comprendido entre 0 transparente y 1 opaco. Esta forma de representar el color es a partir de la especificación CSS3.

```
<h2 style="background-color: hsla(180, 100%, 7%, 0.3);">
```

En internet hay multitud de aplicaciones para poder capturar colores y convertirlos en valores hexadecimales para que el usuario pueda utilizarlos en su diseño (ColorPic, Eye Dropper, ColorPix). Además los navegadores pueden contener plugins para determinar el color de una zona de una página para poder capturarlo.

Encabezados

Los elementos de **encabezado** implementan seis niveles de encabezado del documento, **<h1>** es el más importante, y **<h6>**, el menos importante. Un elemento de encabezado describe brevemente el tema de la sección que presenta. Los encabezados se utilizan en la actualidad para establecer las diferentes áreas dentro de la web. Los motores de búsqueda utilizan los encabezados para **indexar la estructura y el contenido de sus páginas web** en los resultados de búsqueda.

Los encabezados, de mayor a menor importancia, van de **<h1>** hasta **<h6>**, y disponen del atributo **Align** (*--No soportado en HTML5--*) como complemento a esta etiqueta. Los navegadores otorgan estilos por defecto a los elementos **<h>** que incluyen márgenes y diferentes tamaños de letra dependiendo de la jerarquía. Se recomienda por cuestiones de SEO que **solo haya una etiqueta <h1>** por página.

Cada sección de nuestra página (etiqueta **<section>**) debería contener en su interior un encabezado tipo **<h1>** (según HTML5) y de forma correlativa el resto de encabezados si fueran necesarios. No se deben omitir niveles de encabezados, es decir, no debería haber una etiqueta **<h3>** si previamente no hay una etiqueta **<h2>**.

```
</section>
  <h1> El pato </h1>
  <section>
    <h2> Introducción </h2>
    <p> Esta sección amplía el concepto del pato. <p>
  </section>
  <section>
    <h2> Hábitat </h2>
    <p> El pato vive en zonas tranquilas. </p>
  </section>
</section>
```

De todas formas, para formatear el texto de los encabezados se realizará **a través de las hojas de estilo (CSS)** que nos permiten asignar el formato que deseemos a la etiqueta encabezado como el tipo de letra, color, alineación, etc...

El siguiente ejemplo muestra como en una etiqueta del tipo encabezado **<h1>** se modifica el tamaño y el color de la fuente mediante la declaración de un estilo CSS. Una práctica bastante extendida sobre etiquetas que incorporan formatos preestablecidos (márgenes, tamaños etc...) consiste en resetear todos los valores (se explicará mas adelante).

```
<h1 style="font-size:60px; color:blue;"> Cabecera principal </h1>
```

Imágenes

Es posible introducir imágenes en una Web para mejorar su apariencia. La etiqueta que se utiliza es ``. Esta etiqueta NO necesita cierre. El nombre de la imagen se especifica mediante el atributo 'src'. Existe la etiqueta `<picture>` que sería una actualización de esta etiqueta con mas posibilidades de configuración.

El formato de imágenes más utilizado y recomendable por su reducido tamaño que ocupan en memoria son los formatos **JPG**, **GIF** (imágenes animadas), y sobre todo **PNG**. Existen otros formatos como **BMP** o **TIFF** que ocupan mucho más espacio y no todos los navegadores son capaces de visualizarlos. Actualmente hay dos formatos que se están utilizando de forma habitual y que son recomendados por sus propiedades por los navegadores mas importantes **WEBP** y **JPEG2000**.

Ejemplo 1: ``

Ejemplo 2: (carpeta externa) ``

Ejemplo 3: ``

Atributos adicionales

La etiqueta `` dispone de una serie de atributos que la complementan como:

- **'alt'**: Muestra un texto alternativo en lugar de la imagen si no se pudiera mostrar. Con la etiqueta `<title>` se visualiza un pequeño Tooltip, pero no en todos los navegadores. Este atributo es recomendable asignarlo por cuestiones de posicionamiento ya que permite indexar las diferentes imágenes de nuestro site.
- **'height'**: Permite modificar el alto de una imagen. Si se modifica solo este atributo la imagen se escala de forma proporcional manteniendo la escala de ancho-alto. Actualmente las dimensiones de las imágenes se realizan a través de hojas de estilo CSS.
- **'width'**: Permite modificar la anchura de la imagen. Si se modifica solo este atributo la imagen se escala de forma proporcional manteniendo la escala de ancho-alto. Actualmente las dimensiones de las imágenes se realizan a través de hojas de estilo CSS.

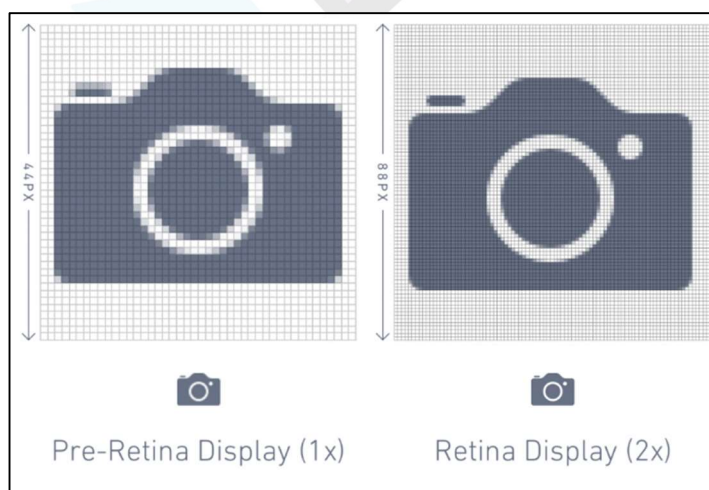
- **'srcset':** (*-- Nuevo en HTML5 --*) Este atributo permite definir **diferentes versiones de la imagen** para mostrar una u otra según el tipo de dispositivo (Viewport). Este atributo prevalecerá respecto al atributo **'src'** si existe conflicto. Este atributo se utiliza normalmente con la etiqueta **<picture>** otorgándole mayor potencia.

Para utilizar imágenes en pantallas con diferente densidad de píxeles se realiza con el atributo **'srcset'** y la ruta de la imagen, seguido de un espacio y un valor que determina la densidad de la imagen (de 0,5X a 4X).

Densidad de píxel: Es el número de píxeles encajados en un determinado espacio (se toma como referencia la pulgada – 2,54cm). Las famosas pantallas de retina (Apple fue la pionera) son de alta densidad de píxel y se consideran de este tipo porque **superan los 326 píxeles** por pulgada. El ojo humano solo es capaz de llegar a distinguir hasta 320 píxeles por pulgada, de ahí su nombre.

Con el valor 480w determina que la imagen solo se cargará en **pantallas de ancho igual o inferior** a 480px, pero sin tener en cuenta la densidad de píxeles.

```
 /* Foto doble densidad
```



	RESOLUCIÓN	NÚMERO DE PÍXELES
HD	1280x720	921.600
FULL HD	1920x1080	2.073.600
UHD (4K)	3840x2160	8.294.400

0.5x 1x 1.5x 2x 3x 4x

- **'ismap':** (– Nuevo en HTML5 –) Permite enviar las coordenadas de la posición donde se ha realizado clic dentro de la imagen. El texto es **tratado desde el lado del servidor** y el formato es un número que corresponde con el eje X y otro con el eje Y, por ejemplo, (56, 40).

```
/* Al pulsar sobre la imagen envía las coordenadas del clic a  
la página 'action_page.php'*/
```

```
<a href="/action_page.php">  
    
</a>
```

Los atributos **align**, **border**, **hspace**, **vspace** NO son soportados por HTML5, pero la mayoría de navegadores los reconoce y los muestra correctamente por pantalla.

Hipervínculo en imágenes

El funcionamiento es igual que si se tratase de un texto, pero en este caso incluyendo una imagen entre las dos etiquetas de hipervínculo.

Ejemplo: ` `

Mapas de imágenes

Los mapas de imágenes se utilizan para asignar varios enlaces a una misma imagen, es decir, una misma imagen con varias **áreas predefinidas** que enlazan a otras páginas. Se pueden construir tantas áreas dentro de una imagen como se desee y hay que intentar que **no se solapen entre sí**.

Para crearlas la etiqueta `` debe tener el atributo `'usemap'` con un identificador de la imagen, y la etiqueta `<map>` haciendo referencia a este identificador determinará las áreas sobre la imagen.

El siguiente código muestra una imagen con 3 áreas (1 rectangular y 2 circulares)

```

```

↕

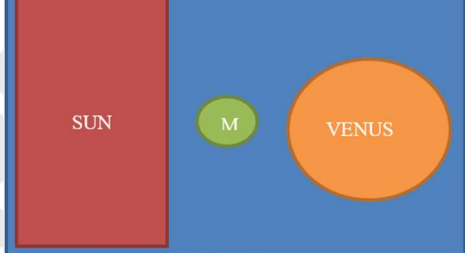
```
<map name="planetmap">

  <area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm" >

  <area shape="circle" coords="90,58,3" alt="Mercury" href="mercur.htm" >

  <area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm">

</map>
```



El atributo `'shape'` tiene el valor `'poly'` que permite crear formas poligonales a partir de las coordenadas de los vertices. Si no coinciden las coordenadas iniciales con las finales el polígono se cierra de forma automática. La sintaxis de las coordenadas es `x1,y1, x2, y2, x3, y3, etc...`

```
<area shape="poly" coords="90,25,162,26,163,96,89,25" href="moon.htm">
```

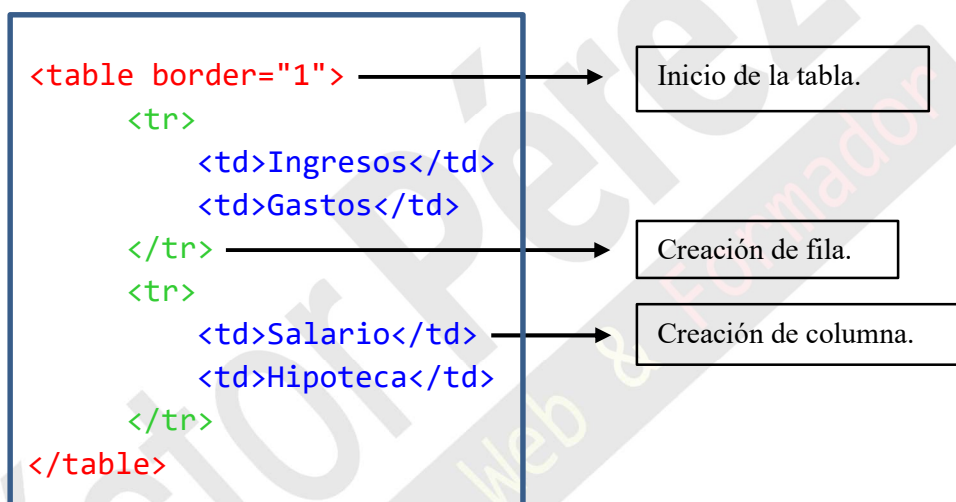
Existen aplicaciones on-line y de escritorio que permiten visualizar las coordenadas de las imágenes para realizar mapeos de forma sencilla y rápida como *'Easy Image-Map'* o *'Image Map Generator'*.

El inconveniente de utilizar los mapeos en imágenes se produce cuando la imagen cambia de dimensiones porque utilizamos un dispositivo diferente al que previsto (movil, tablet) o la ventana del navegador se reduce. En estos casos el área de mapeo al cambiar el tamaño de la imagen no coincide. Además no es posible trabajar con unidades relativas como porcentajes.

Tablas (uso poco frecuente)

Para crear una tabla hay que insertar las etiquetas `<table>` y `</table>`. Entre dichas etiquetas habrá que especificar las filas y columnas que formarán la tabla. En una celda de la tabla puede ir tanto texto, como imágenes o cualquier tipo de archivo multimedia de audio y video.

Para crear filas se hace mediante la etiqueta `<tr>` y `</tr>`. Siempre se tendrán que crear las filas antes que las columnas. Para crear columnas se hace mediante la etiqueta `<td>` y `</td>`. Las columnas se crean siempre después de haber creado la fila.



Al trabajar con tablas es posible modificar el formato de la tabla, el formato de la fila, o el formato de cada una de las celdas si es preciso.

Atributos para modificar el formato de la tabla o celda. (*--No soportados desde HTML5--*).

align	Alinea horizontalmente la tabla respecto a su entorno (left, center, right).
background	Permite colocar un fondo para la tabla a partir de un enlace a una imagen.
bgcolor	Establece color de fondo a la tabla.
border	Define el número de píxeles del borde principal (grosor).
bordercolor	Define el color del borde de la tabla o celda.
cellpadding	Espacio en píxeles entre los bordes de la celda y el contenido de la misma.

cellspacing	Define el espacio entre los bordes (en pixeles).
height	Define la altura de la tabla en pixeles o porcentaje.
width	Define la anchura de la tabla en pixeles o porcentaje.

Ejemplo 1: `<table width="50%" border="2" align="center" cellspacing="0" bordercolor="#000000" bgcolor="#FFCC99"> Contenido tabla </table>`

Ejemplo 2: `<table width="50%" border="2" align="center" valign="middle" cellspacing="0" bordercolor="#000000" bgcolor="#FFCC99"> Tabla </table>`

Existe el atributo '**nowrap**' que provoca que el contenido de la celda NO se ajuste de manera automática al ancho de la columna, sino que el ancho de la celda se adapta al ancho del contenido para que el contenido ocupe una sola fila. Este atributo no toma ningún valor, simplemente se añade o no a las etiquetas `<td>`.

El atributo **frame="void"** permite eliminar el borde de la tabla y el atributo **rules="none"** permite eliminar el borde de las celdas.

Existe la etiqueta `<th>` que actúa de forma similar a la etiqueta `<td>`. La etiqueta `<th>` es posible especificar los mismos atributos que para la etiqueta `<td>`, pero esta nueva etiqueta hace que el texto de la celda aparezca centrado y en negrita, por lo que se utiliza para definir los encabezados o títulos de las columnas. Es una etiqueta similar a `<tr>` pero al ser diferente del resto de la tabla se le puede asignar un formato sin que afecte al resto de las filas de la tabla.

La etiqueta `<caption>` permite asignarle un nombre a la tabla generada. El nombre de la tabla puede ser interesante en relación con el posicionamiento de la web ya que los motores de búsqueda se fijan mucho en las descripciones de los elementos.

```

<table border="7" align="center">
  <caption align="top"> Titulo </caption>
  <tr>
    <th> Ingresos </th>
    <th> Gastos </th>
  </tr>
  <tr>
    <td> Salario </td>
    <td> Hipoteca </td>
  </tr>
</table>

```

→ Título de la tabla.

→ La etiqueta **<th>** toma la fila como **encabezado** y aplica un efecto de negrita dejando el texto centrado en su interior.

La etiqueta **<td>** es igual que la etiqueta **<th>** pero sin la connotación que es una cabecera.

Combinar celdas

Las etiquetas **<td>** y **<th>** tienen los atributos **colspan** y **rowspan**, que se utilizan para combinar celdas.

A través del atributo **colspan** se especifica el número de columnas por las que se extenderá la celda (número de columnas que se combinan), y a través del atributo **rowspan** se especifica el número de filas por las que se extenderá la celda (número de filas que se combinan).

La primera fila de la tabla determina el número máximo de columnas que tendrá la tabla.

El resto de filas deberán tener exactamente el mismo número de columnas, ya sea la suma de las combinadas y las que no lo están. En el siguiente ejemplo se realizan varias combinaciones.

A continuación se muestra una tabla con celdas combinadas. Para generar esta estructura es necesario combinar filas y columnas respetando las posibles celdas ocupadas.

Tabla de salarios

Ingresos medios empresa				
Ingresos Enero		Ingresos Febrero		Ingresos Abril
Salario A	Salario B	Salario C	Salario D	

```
<table border="5" align="center" bgcolor="#CCAAB2" >
```

Propiedades tabla

```
<caption> Tabla de salarios </caption>
```

Nombre de la tabla.

```
<tr>
    <th colspan="9"> Ingresos medios </th>
</tr>
```

La 1ª fila de la tabla determina el número máx. de columnas que contendrá.

```
<tr>
    <td colspan="3"> Ingresos Enero </td>
    <td colspan="3"> Ingresos Febrero </td>
    <td colspan="2"> Ingresos Marzo </td>
    <td rowspan="2"> Ingresos Abril </td>
</tr>
```

En las siguientes filas tienen que coincidir la suma de columnas con la primera fila. (3+3+3=9).

```
<tr>
    <td colspan="2"> Salario A </td>
    <td colspan="2"> Salario B </td>
    <td colspan="2"> Salario C </td>
    <td colspan="2"> Salario D </td>
</tr>
```

Las Colspan se contabilizan como la suma de los números que contienen cada una de ellas (3+5). En cambio, las Rowspan se contabilizan solo como 1 columna, aunque el número sea mayor. En este caso 3+5+1=9.

```
</table>
```

Desde la introducción de HTML5 se han introducido tres etiquetas nuevas para tablas:

- **<thead>**: Permite agrupar el contenido de la cabecera de una tabla.
- **<tbody>**: Permite agrupar el contenido del cuerpo de una tabla.
- **<tfoot>**: Permite agrupar el contenido del pie de una tabla.

<table>

```
<thead>  (azul)
  <tr>
    <th>Meses</th>
    <th>Salario</th>
  </tr>
</thead>
```

Con la utilización de las etiquetas de agrupamiento se pueden especificar estilos CSS para cada una de las zonas agrupadas son necesidad de asignar fila por fila cada estilo.

```
<tfoot>  (verde)
  <tr>
    <td>Total</td>
    <td>300€</td>
  </tr>
</tfoot>
```

El orden de las etiquetas de agrupamiento dentro de una tabla NO afecta al resultado final de la tabla. La etiqueta **<thead>** irá siempre en la zona superior, la etiqueta **<tbody>** en el centro, y la etiqueta **<tfoot>** al final de la tabla.

```
<tbody>  (rojo)
  <tr>
    <td>Enero</td>
    <td>100€</td>
  </tr>
  <tr>
    <td>February</td>
    <td>200€</td>
  </tr>
</tbody>
```

Meses	Salarios
Enero	100€
Febrero	200€
Total	300#

</table>

NOTA: En la actualidad el uso de las tablas se centra esencialmente en mostrar datos tabulados (en líneas unas debajo de otras) extraídas de una consulta de una base de datos. Para temas de diseño en la estructura de las páginas se utilizan las capas mediante la etiqueta **<div>** o similares con significado semántico desde HTML5.

Enlaces – Hipervínculos - Links

Un hiperenlace es un elemento esencial para la creación de páginas Web. Se puede denominar de varias formas, pero todas significan lo mismo: hiperenlace, vínculo, link, hipervínculo, enlace y otras.

El hiperenlace al ser pulsado nos redirige hacia una página o un archivo. El texto no es el único elemento que puede contener esta característica dentro de una web, sino que las imágenes y otros elementos que la conforman también pueden contener un vínculo creado en su interior.

La etiqueta que se utiliza para introducir hipervínculos es `<a>` y ``. Mediante el atributo **href** se especifica la página de enlace y que página se visualizará cuando el usuario haga clic. Los hiperenlaces de texto se visualizan "como norma general" en color azul aunque es modificable desde CSS.

- Ejemplo: Este link redirige a la Web de la `` Generalitat ``
- Resultado en la web: Este link redirige a la página de la [Generalitat](http://www.gencat.cat).

Tipos de referencias o enlaces

Existen diferentes formas de expresar una referencia a una página a través del atributo **href**.

1. Referencia absoluta: Conduce a una **ubicación externa** al sitio en el que se encuentra el documento. Si no se escribe el nombre de la página se cargará la página de inicio asociada al dominio. Es preciso escribir el texto **http://** que hace referencia al protocolo.

Ejemplo: `` Enlace externo ``

2. Referencia relativa al mismo sitio: Conduce a un documento situado **dentro del mismo sitio** o directorio que el documento actual. Si todas las páginas se encuentran dentro de la misma carpeta se puede escribir directamente el nombre de la página sin necesidad de escribir toda la ruta.

Ejemplo: `` Enlace en misma página ``

3. Referencia relativa a un documento: Conduce a un documento o recurso situado dentro de un subdirectorio de la carpeta actual (ejemplo 1). Si por el contrario la referencia relativa está por encima del directorio actual el código utilizado será el `../` para acceder al directorio superior (ejemplo 2).

Ejemplo 1: `` Relativa mismo directorio ``

Ejemplo 2: `` Relativa desde nivel superior ``

4. Punto de fijación, marcador (ancla): Conduce a un punto en concreto dentro de un documento, ya sea dentro del actual o de otro diferente. Un ancla puede realizarse sobre una misma página o sobre una página diferente del Website. La estructura, en función del tipo de salto hacia el ancla, puede ser:

- Ancla a página diferente: `"página#nombre_de_punto_fijación"`.
- Ancla en la misma página: `"#nombre_de_punto_fijación"`.

Para crear un ancla se realiza con el atributo `'href'` (el texto que enlaza) y con el atributo `'id'` (en texto enlazado). El atributo `'name'` que se utilizaba para crear anclas no está aceptado desde el estándar HTML5.

Los pasos para crear el marcador (ancla) y el enlace en la misma página son los siguientes:

- a. Crear el enlace en cualquier punto de la web hacia el marcador que se ha creado. Aquí si se introduce el texto que llevará el link.

Ejemplo 1: (misma página)

`` Enlace a 'ancla_ejemplo' que se encuentra dentro de la misma página. ``

Ejemplo 2: (diferente página)

`` Enlace al 'ancla_ejemplo' que se encuentra ubicado en otra página. ``

Código ancla con estándar HTML5

```
<body>

    <h2 id="top"> Zona superior de la página </h2>

    ----- Contenido página -----

    <a href="#top"> Ir a zona superior </a>

</body>
```

Otros tipos de referencias o enlaces

Además de los enlaces explicados en el punto anterior existen los siguientes tipos de enlaces:

1. Correo electrónico: Es un enlace que abre Outlook Express. El atributo utilizado es **mailto**: Existen otros atributos que acompañan al correo. La sintaxis cambia ligeramente de lo utilizado hasta el momento en lenguaje HTML. En el siguiente ejemplo se muestran las opciones más utilizadas.

// Envía un mensaje de correo a la dirección especificada

Ejemplo: ` Dirección de correo `

2. Teléfono: (--- Desde HTML5 ---) El atributo 'href' acepta el valor 'tel' para especificar un número de teléfono al cual se llamará al ser pulsado el texto del enlace.

// Hace una llamada al número de teléfono de la etiqueta

Ejemplo: ` Pulsa para llamar `

3. Ficheros de descarga: Es posible también realizar un enlace a un archivo ya sea para abrirlo o para descargarlo en la máquina local. Los navegadores reconocen la mayoría de las extensiones de los archivos. Cuando se descarga el archivo se muestra una ventana que nos da la opción de Abrirlo o Guardarlo en el ordenador. Si se incluye el atributo 'download' se descargará de forma automática.

// Abre un archivo de Excel, se puede abrir o guardar

Ejemplo: `` Descarga fichero Excel ``

// Especifica el idioma del documento linkado

Ejemplo: `` Idioma documento ``

// Especifica la relación entre el doc actual y el linkado

Ejemplo: `` Tipo relación ``

// Especifica el tipo de documento media vinculado

Ejemplo: `` Tipo relación ``

4. Vínculo vacío: Al pulsar sobre un enlace vacío no se abre ninguna página ni archivo, pero el formato es el mismo que el de cualquier otro enlace. El vínculo debe ser el símbolo almohadilla "#". Este tipo de enlace resulta útil para trabajar con comportamientos Javascript.

// Vínculo vacío para lanzar comportamientos Javascript

Ejemplo: `` Enlace a vínculo vacío ``

Destino del enlace

El destino del enlace determina en qué ventana va a ser abierta la página vinculada, se especifica a través del atributo **target** al que se le puede asignar los siguientes valores:

1. **_blank**: Abre el enlace en una nueva ventana del navegador. Los navegadores actuales realizan la apertura de la nueva página en una pestaña del mismo. Es la opción que normalmente se utilizará si se decide utilizar el atributo '**target**'.

Ejemplo: ``
Enlace externo abriéndose una nueva ventana ``

2. **_self**: Es la opción predeterminada y abre el enlace en la misma ventana.

Ejemplo: `` Enlace
externo abriéndose una nueva ventana ``

3. **_parent**: (*-- En desuso --*). Abre el enlace en la ventana del marco que contiene el vínculo. Esta opción se utiliza cuando se trabaja con marcos, pero desde la última actualización en HTML5 y las nuevas técnicas de maquetación han entrado en desuso.
4. **_top**: (*-- En desuso --*). Abre el documento vinculado en la ventana completa del navegador. Esta opción también es utilizada en el trabajo de marcos.


NOTA: Actualmente los dos tipos de target más utilizados son '*_blank*' y '*_self*'.

Listas de elementos

Las listas permiten introducir un listado de elementos dentro de una estructura organizada. Las etiquetas que permiten hacer esta acción son `` y ``. Las listas también se pueden utilizar para crear menús y barras de navegación.

Ejemplo:

```
<li> Barcelona </li>
<li> Roma </li>
<li> Paris </li>
```



- Barcelona
- Roma
- Paris

Listas desordenadas

Las listas desordenadas son muy similares a las listas básicas. La diferencia radica en el tipo de dibujo que precede cada elemento de la lista que puede ser un **círculo (circle)**, **cuadrado (square)**, o **disco (disc)**. La etiqueta `` se sitúa al inicio y al final de la lista. En el siguiente ejemplo se especifica el tipo de dibujo mediante estilo CSS y la propiedad `'list-style-type'`.

Ejemplo:

```
<ul style="list-style-type:circle;">
  <li> Barcelona </li>
  <li> Roma </li>
  <li> Paris </li>
</ul>
```



- Barcelona
- Roma
- Paris

Listas ordenadas

Las listas ordenadas, a diferencia de las desordenadas, permiten enumerar (de diversas formas) el listado de elementos que conforman la lista. En el atributo `'type'` puede adoptar los siguientes valores:

- **1** (números),
- **a** (letras minúsculas).
- **A** (letras mayúsculas).
- **i** (números romanos en minúsculas).
- **I** (números romanos en mayúsculas).

Ejemplo: `<ol type = "A" >`

` Barcelona `

` Roma `

` Paris `

``



- A. Barcelona
- B. Roma
- C. Paris

Las listas permiten estar **anidadas**, es decir, crear listas dentro de otras listas. La estructura de la lista anidada es igual que una simple. Solo es preciso tener claro cuando empieza y acaba.

``

` Café `

`Té`

``

` Té negro `

` Té verde `

``

``

` Leche `

``



- Café
- Té
 - Té negro
 - Té verde
- Leche

Listas de definición

Es una lista de términos, con una descripción de cada término. Trabaja conjuntamente con las etiquetas `<dt>` y `<dd>` para definir los ítems de la lista. La configuración de las listas de definición es muy similar a las listas ordenadas o desordenadas. Estas últimas siempre tendrán los elementos hijos anidados mientras que las listas de definición no.

- `<dl>`: Define una lista de descripción, inicio y fin de lista.
- `<dt>`: Define los términos de los objetos de lista.
- `<dd>`: Descripción de cada elemento de la lista.

`<aside>`

`<dl>`

`<dt> Chocolate </dt>`

`<dd> Cacao </dd>`

`<dd> Azúcar </dd>`

`<dd> Leche </dd>`

`<dt> Caramelo </dt>`

`<dd> Azúcar </dd>`

`<dd> Colorantes </dd>`

`</dl>`

`</aside>`



Chocolate
Cacao
Azúcar
Leche
Caramelo
Azúcar
Colorantes

Etiqueta <iframe>

La etiqueta <iframe> se utiliza para crear un espacio dentro de la página web donde se puede incrustar otra web o elemento. Es un recuadro cuyas dimensiones y propiedades a nivel de formato se deben especificar con hojas de estilo CSS. Un caso muy habitual consiste en incrustar vídeos de Youtube en nuestro site.

Esta etiqueta es muy útil si nuestros videos están en un servidor de redes sociales y desde nuestra página generamos el link. De esta forma obtenemos dos beneficios:

1. Los videos no estarán alojados en nuestro servidor provocando un consumo mayor de datos.
2. El vídeo se cargará de forma más rápida.
3. Generaremos tráfico desde redes sociales mejorando el SEO.

Estos serían los atributos disponibles para la configuración de la etiqueta <iframe>:

- **src:** Indica la página web que se mostrará en el espacio del frame. Hace referencia al elemento que será cargado en la página.
- **width:** Define la anchura del recuadro del iframe.
- **height:** Define la altura del iframe.
- **name:** Especifica el nombre del iframe que deseamos generar. utilizar luego para referenciarlo a él mediante Javascript.
- **id:** Para indicar el identificador del iframe, y poder referirnos a él desde Javascript. Los 'id' son una forma muy utilizada para asignar estilo a los elementos de la página.
- **name:** Se utiliza para asignar un nombre al frame. Esto puede ser necesario para cargar enlaces en un determinado espacio a través del atributo 'target'.

```
<iframe name="Iframe_A" src="página.htm"></iframe>
```

```
<a href="https://miweb.com" target="Iframe_A"></a>
```

- **style y class:** Los atributos para definir el aspecto del iframe se realiza por medio de hojas de estilo CSS. Es posible utilizar el atributo 'style' con todas las posibilidades de configuración de las propiedades (left, top, width, etc...) o utilizar los atributos 'id' o 'class' para asignarle un estilo CSS.
- **srcdoc:** (-- Nueva HTML5 --). Especifica el contenido de la página que se visualizará en el iframe. Realiza la misma tarea que el atributo 'Alt' para las imágenes'.

```
<iframe srcdoc="Video musical" src="página.htm"></iframe>
```

- **sandbox:** (-- Nueva HTML5 --) Habilita un conjunto adicional de restricciones para el contenido de un iframe. Los posibles valores pueden ser:

Valor	Descripción
(no value)	Aplica todas las restricciones.
allow-forms	Habilita la presentación de formulario.
allow-pointer-lock	Habilita las API's.
allow-popups	Habilita popups (ventana emergente) Asociado al enlace 'target blank'.
allow-same-origin	Permite que el contenido del iframe sea tratado como siendo del mismo origen
allow-scripts	Habilita scripts
allow-top-navigation	Permite que el contenido de iframe navegue por su contexto de navegación de nivel superior

Ejemplo 1: El atributo 'sandbox' permite visualizar el contenido de los iframes que contengan formularios y scripts. Es posible asignar más de un valor a 'sandbox'.

El siguiente código habilita la presentación de formularios y de scripts dentro del iframe.

```
<iframe src="demo_iframe_sandbox.htm" sandbox="allow-forms allow-scripts"> </iframe>
```

Esta línea determina el ancho y alto del iframe, la página que abrirá por defecto al cargar la página antes de pulsar el enlace, y el nombre que tendrá el iframe para hacer referencia a él.

```
<iframe height="300px" width="100%" src="demo_iframe.htm" name="iframe_a"> </iframe>
```

Esta línea es un enlace que abre la página de Google, en la ventana (target) referencia con el nombre 'iframe_a', referida al código de la primera línea del ejemplo.

```
<a href="http://www.google.com" target="iframe_a"> Página de Google </a>
```

Capas, Cajas, Contenedores <div>

Para trabajar con las capas y todas sus funcionalidades hay que comprender la diferencia entre las etiquetas que se utilizan y que trabajan como '*Elementos en bloque*' o como '*Elementos en línea*'.

- **Elementos en bloque:** (BLOCK) Son las etiquetas HTML que al insertarse en una página **ocupan todo el ancho disponible de la línea** donde se encuentran ubicados de izquierda a derecha. Estos elementos tienen un tamaño personalizado y generan saltos de línea. Por ejemplo, etiquetas que tienen estas propiedades son <section>, <header>, <nav>, <footer>, <h1>, <div>, <p>, , etc..

<div> Caja que ocupa el ancho al 100% </div>

La etiqueta <div> es un elemento de bloque ocupa todo el ancho de la página (100%).

- **Elementos en línea:** (INLINE) Son las etiquetas HTML que **ocupan exclusivamente el ancho necesario para su colocación** dentro de la página. NO respetan los anchos, los altos, ni los márgenes verticales de los elementos. Se colocan uno al lado de otro en la misma línea, a menos que no haya suficiente espacio horizontal. Su tamaño depende del contenido y no generan saltos de línea. Por ejemplo, las etiquetas <a>, , , etc... disponen de esta propiedad.

 Caja que ocupa el tamaño necesario

La etiqueta es un elemento de línea, solo ocupa lo necesario

- **Elementos en bloque ajustado:** (INLINE-BLOCK) Son las etiquetas HTML que pueden tener un comportamiento inicial en Bloque o en Línea indistintamente, pero que a través de las hojas de estilo se adaptan a un ancho específico. NO respetan los anchos, los altos, ni los márgenes verticales de los elementos. Cualquier etiqueta HTML puede ajustarse su tamaño.

<div style="display:inline-block" > Caja que ocupa un ancho específico </div>

La etiqueta <div> con ancho ajustado (80%)

Las capas o contenedores **<div>** son unos recuadros que de forma predeterminada **trabajan como un elemento en bloque**, aunque posteriormente esta característica puede modificarse mediante las hojas de estilo CSS.

Las cajas las podemos ubicar en cualquier zona de la página como si fuesen un cuadro de texto. El posicionamiento también se realizará mediante CSS. En este aspecto hay que tener presente que todas las cajas se posicionan tomando como referencia las cajas previamente insertadas.

Las cajas pueden ser **modificadas completamente mediante hojas de estilo CSS**, con los atributos y los posibles valores que puedan adoptar.

En el siguiente ejemplo se muestran los atributos y sus valores más comunes que podemos utilizar en una etiqueta **<div>**. En este caso todos los estilos se incluyen "incrustados" en la misma etiqueta para una mayor comprensión, pero esta acción se realiza desde las hojas de estilo CSS. La explicación del mismo se realiza a continuación.

Ejemplo 1: Estilo integrado dentro de la misma etiqueta.

```
<div id="capa" style="left:100px; top:40px; position:absolute;
width:200px; height:115px; z-index:3; visibility: visible;
background-color: #0099CC; background-color: #0099CC; background-
image: url(fondocapa.jpg); background-image: url(fondocapa.jpg);">
Texto que se encontrará dentro de la capa.
</div>
```

Explicación Ejemplo 1:

- **id:** Determina el nombre que tendrá la capa en la página. Es un identificador único para referenciar al elemento (DNI, matrícula vehículo, ISBN, etc...). Los identificadores se utilizan para poder acceder y actuar sobre un determinado elemento normalmente desde Javascript.
- **style:** Este atributo se utiliza cuando deseamos incrustar los estilos del elemento dentro de la misma etiqueta. Puntualmente podemos utilizar este sistema pero lo correcto es hacerlo a través de hojas de estilo para liberar al documento HTML de código.

- **left , top:** Se establece la posición de la capa respecto a los márgenes izquierdo y superior de la página. Pueden tomar un número como valor, acompañado de **px** cuando haga referencia a píxeles, y acompañado de **%** cuando haga referencia a un porcentaje.

```
.caja { left: 80px; }
```

```
.caja { top: 20%; }
```

- **position:** Con el valor 'absolute' la capa se mostraría pegada al margen izquierdo superior de su caja madre. Si hubiera algún elemento más estos quedarían solapados. Por este motivo el atributo '**position**' precisa de los atributos '**top**' y '**left**' para mover el elemento.
- **width y height:** Se establece el tamaño de la capa. Pueden tomar un número como valor, acompañado de **px** cuando haga referencia a píxeles, acompañado de **%** cuando haga referencia a un porcentaje o cualquiera de los sistemas disponibles.
- **z-index:** Puede establecerse el índice de la capa dentro de la página. Una capa podrá estar solapada por aquellas capas cuyo índice sea mayor, es decir, a mayor número de esta propiedad la capa estará por encima de las otras capas.
- **visibility:** Puede establecerse la visibilidad de la capa. Puede tomar los valores:
 - Inherit: (heredada). Se muestra la capa mientras la capa padre a la que pertenece también se esté mostrando.
 - Visible: Se muestra la capa, aunque la capa a la que pertenece no se esté viendo.
 - Hidden: La capa está oculta, aunque la caja madre esté visible.
 - Collapse: Aplicado solo en tablas. No permite que el contenido se muestre fuera de la celda.

```
.caja { visibility: hidden;}
```

- **background-image:** Se puede establecer una imagen de fondo para la capa. La ruta y el nombre de la imagen han de aparecer entre paréntesis, después de la palabra **url**.

```
body { background-image: url("paper.gif"); }
```

- **background-color:** Se puede establecer un color de fondo para la capa. Puede ser un número hexadecimal o cualquier nomenclatura aceptada por W3C.

```
.caja { background-color: blue; }
```

- **overflow:** Puede establecerse si se mostrará o no el contenido de la capa cuando no pueda ser visualizado en su totalidad, por ser la capa demasiado pequeña. Puede tomar los valores:
 - **visible:** Se muestra todo el contenido de la capa, aunque esto implique hacer que la capa sea más grande.
 - **hidden:** No es posible visualizar el contenido de la capa que sobresale en ella.
 - **scroll:** Se muestra una barra de desplazamiento, aunque el contenido de la capa pueda ser visualizado totalmente.
 - **auto:** Se muestra la barra de desplazamiento cuando sea necesario.
- **clip:** Puede establecerse el área de la capa que podrá ser visualizado. Lo que hace es recortar la capa, haciendo que partes de ella no sean visibles. Ha de especificarse la distancia de los márgenes de la capa entre paréntesis, después de la palabra **url**. Sistema para uso en casos muy específicos.

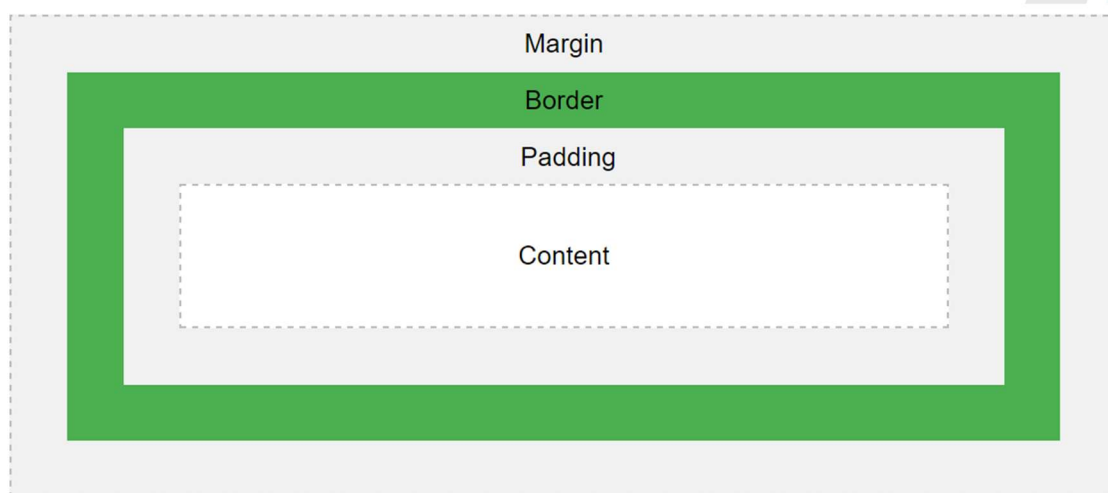
Es de uso muy extendido entre diseñadores web trabajar anidando capas o cajas unas dentro de otras en varios niveles de profundidad utilizando la etiqueta **<div>**. En HTML5 la etiqueta **<div>** puede ser sustituida por otras que adquieren mayor valor semántico actuando de la misma forma como **<aside>**, **<article>**, **<header>**, **<footer>**, etc.. pero sin repetir tanto la etiqueta **<div>**.

En el ejemplo anterior el formato de la capa se ha establecido mediante el atributo **style**, pero lo más adecuado es asignarle el estilo mediante una hoja de estilos enlazada a la página web. La tendencia actual es separar el contenido del contenedor, dicho de otra forma, separar el texto de su formato. La finalidad de todo esto es trabajar con mayor velocidad.

Modelo de cajas en CSS

Todos los elementos en HTML son cajas o contenedores de información. Las cajas se componen de márgenes, bordes, relleno y contenido real. Todas las cajas interactúan con las cajas que tienen a su alrededor para conformar la estructura de la página.

La siguiente imagen ilustra el modelo de caja con estas propiedades.



- **'Content'** – Es el contenido de la caja, donde aparecen textos, imágenes, etc... El contenido puede interactuar con la caja con estilos básicos como por ejemplo, la alineación, el tamaño de la letra, el color, el sombreado, el color de fondo, etc...

```
p { text-align: center; background-color: blue; }
```

- **'Padding'** – (Relleno). Representa el margen interior del elemento.

```
p { padding: 2cm 4cm 3cm 4cm; } // superior-derecha-inferior-izq
```

- **'Border'** – Representa la línea perimetral que está alrededor del elemento. El borde se especifica obligatoriamente con los valores de '*grosor*', '*estilo línea*' y '*color*' (en formato W3C). Un aspecto muy importante a tener en cuenta es que al aumentar el borde de la caja las dimensiones de la misma también aumentan por defecto.

```
p { border: 5px solid red; }
```

- **'Margin'** - Borra una zona fuera de la frontera. El margen es transparente.

```
p { margin: 5vw; } // Los 4 márgenes con la misma distancia
```

Para contabilizar el tamaño de una caja hay que tener presente la suma de todos sus elementos y no solo del contenido del mismo. El en siguiente código se muestra el tamaño real del elemento que es la suma de todas las propiedades (visibles y no visibles).

```
div {  
  width: 320px;  
  padding: 10px;  
  border: 5px solid gray;  
  margin: 0;  
}
```



Este elemento tiene la anchura de 320px + 10px por cada lado + el borde de 5px de izquierda y derecha.

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

Las cajas de una página **se crean automáticamente**. Cada vez que se inserta una etiqueta HTML, se crea una nueva caja rectangular que encierra los contenidos de ese elemento. Las cajas de las páginas no son visibles a simple vista porque inicialmente no muestran ningún color de fondo ni ningún borde. Cada una de las cajas está formada por seis partes.

Nuevas etiquetas HTML5

En este apartado realiza una breve explicación sobre las nuevas funcionalidades del estándar HTML 5 creadas por Ian Hickson (editor de HTML5). Básicamente se centra en las nuevas etiquetas semánticas que se han incorporado en el estándar de uso recomendado.

HTML 5 es el nuevo estándar para desarrollo web que incluye las siguientes tecnologías:

- HTML: Estructura de las páginas web (esqueleto).
- CSS: Estilo o formato del documento (apariencia).
- Javascript: Interactividad y animación que pueden disponer las páginas (dinamismo).

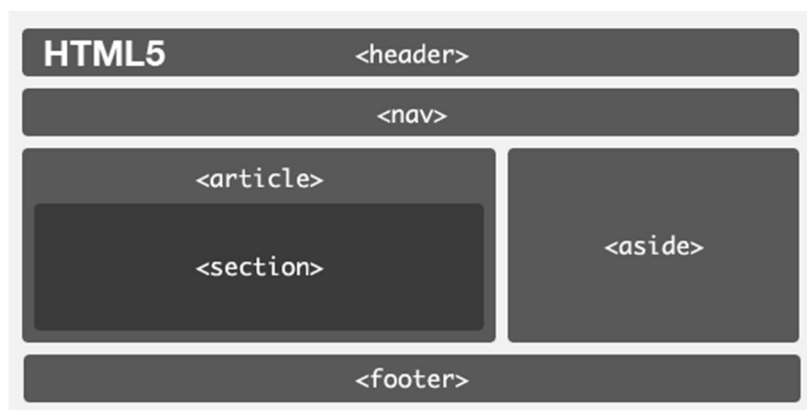
La disposición de la ubicación de los elementos en una página web muestra el contenido en múltiples columnas (como una revista o un periódico). HTML5 ofrece **nuevos elementos semánticos** que definen las diferentes partes de una página web.

El nuevo estándar pretende mejorar las características del anterior HTML4 y XHTML. HTML4 y HTML5 son 100% compatibles entre sí. Implementa etiquetas semánticas, es decir, que por sí mismas ya tienen un significado y evita que los diseñadores abusen tanto de la etiqueta **<div>**.

Las principales etiquetas HTML5 nuevas no tienen una representación especial en pantalla. Todas se comportan como un **<div>** o un ****, pero cada una tiene un significado semántico superior a un simple **<div>** o ****. En las siguientes imágenes se muestran dos estructuras HTML5 válidas. La primera seguiría el sistema clásico y la segunda el sistema recomendado.



En HTML 'antiguo' cada zona de la estructura de la web debía tener su correspondiente identificador para que mediante CSS se le pudiese dar formato (tamaño, posicionamiento, etc...)



En HTML5 las nuevas etiquetas ya tienen sentido semántico propio. Mediante CSS se les puede dar el formato que se desee, pero para la indexación en los motores de búsqueda es mucho mejor utilizar esta nueva nomenclatura.

La distribución de los elementos y las etiquetas en la creación de una web pueden ser muy diversos, es decir, no siempre se encontrarán ubicados en la misma zona, aunque semánticamente el contenido quede reflejado mediante la etiqueta.

A grandes rasgos, las aportaciones del nuevo estándar HTML5 son:

1. Elementos semánticos con significado propio, `<header>`, `<footer>`, `<aside>`, `<section>`, etc... Estos nuevos elementos trabajan como elementos de bloque aunque siempre podrán adaptarse según las necesidades del proyecto.
2. Nuevos atributos para los **elementos de formulario**, fecha, hora, número, calendario, etc...
3. Nuevos elementos multimedia `<audio>` y `<video>`.
4. Nuevos elementos gráficos `<svg>` (elementos vectoriales) y `<canvas>` (dibujo Javascript).

La nueva API para HTML5 permite realizar las siguientes acciones a nivel de programación:

- Geolocalización HTML.
- Drag and Drop en HTML (arrastrar y soltar).
- HTML Local Storage.
- Caché en HTML.
- Web Workers en HTML (rutinas en segundo plano).
- HTML con SSE (actualizaciones automáticas desde el servidor).

Para empezar, el nuevo estándar dispone de una nueva declaración de documento o Doctype. El 'doctype' determina el conjunto de caracteres utilizados y que serán interpretados por el navegador para la construcción de la página.

```
<!doctype html> // Declaración para HTML5
```

Algunas recomendaciones de estilo sobre el nuevo estándar HTML5 sugieren:

- **Utilizar siempre minúsculas** en el código HTML5, aunque las mayúsculas se aceptan.
- **Eliminar espacios innecesarios** entre atributos y valores.
- **Cerrar todas las etiquetas**, aunque estas no tengan específicamente una etiqueta para ello.
- Poner **siempre dobles comillas** (") a los valores de los atributos, aunque permita no hacerlo.
- Introducir **el atributo 'alt'** en las imágenes para evitar problemas de parpadeo en la carga.
- No introducir líneas de código que superen 80 caracteres.
- **Utilizar comentarios** para la posterior lectura del código.
- **Tabular el código** de programación para hacer más fácil la lectura y compresión del mismo.
- **Ajustar la ventana gráfica** al dispositivo de salida (movil, tablet, etc...).
- Las etiquetas **<head>**, **<body>**, y **<html>** se podrían omitir, pero no es recomendable.
- Javascript es muy **sensible a mayúsculas y minúsculas**, así que todo en minúsculas.
- Utilizar la sintaxis adecuada para las hojas de estilos si son reglas cortas o reglas largas.

```
// Regla en formato corto
```

```
p.intro { font-family: Verdana; font-size: 16em; }
```

```
// Regla en formato largo
```

```
body {
```

```
    background-color: lightgrey;
```

```
    font-family: "Arial Black", Helvetica, sans-serif;
```

```
    font-size: 16em;
```

```
    color: black; }
```

Las etiquetas nuevas en HTML5 para la distribución de elementos son:

- **<header>**: Se utiliza para añadir la cabecera del sitio web. Todo lo que se introduzca en esta etiqueta se mostrará en la parte superior de la web. No confundir con etiqueta <head>. Hacer cosas como `<div id="header">` es innecesario porque la gran mayoría de proyectos web incluyen una cabecera en la zona superior.

La etiqueta **<header>** está diseñada para reemplazar la necesidad de crear divs sin significado semántico. Esta etiqueta también se puede utilizar en elementos tipo **<article>** para definir la cabeza y el pie del bloque.

El siguiente código muestra la etiqueta **<header>** utilizada como encabezamiento de artículo y no como encabezamiento de la página web que sería el caso más habitual.

```
<article>

  <header> // Cabecera de la página

    <h1> Cabecera más importante </h1>

    <h3> Segunda cabecera más importante </h3>

    <p> Información adicional </p>

  </header>

  <p> Texto del artículo </p>

</article>
```

- **<nav>**: Se utiliza para introducir los menús de la web y/o botones de la página web. Se puede introducir cualquier etiqueta en su interior, aunque lo recomendado es usar listas **** si se desea construir un menú. Esta etiqueta trabaja inicialmente como un elemento de bloque. Los diseñadores introducen la etiqueta **<nav>** dentro de la etiqueta **<header>**.

Se podría utilizar la etiqueta **<div>** para crear el menú de la web, pero con esta etiqueta además de otorgarle un sentido semántico a esta zona, estaremos facilitando la indexación de nuestro contenido a los motores de búsqueda ya que reconocerán fácilmente cada parte de la web.

El siguiente código es un menú sencillo con 4 opciones.

```
<nav> // Menú de la página.

  <a href="/html/"> HTML </a>

  <a href="/css/"> CSS </a>

  <a href="/js/"> JavaScript </a>

  <a href="/jquery/"> jQuery </a>

</nav>
```

- **<section>** y **<article>**: Con estas dos etiquetas hay la controversia cuál de ellas va dentro de la otra. Es evidente que son divisiones dentro de la web (como si fuese una capa) pero en este caso no queda claro si un artículo está compuesto por secciones o una sección está compuesta por artículos. La tendencia actual es ubicar la etiqueta **<article>** dentro de **<section>**.

Posteriormente, dentro de una etiqueta **<article>** se pueden introducir otras etiquetas como **<p>**, ****, **<h>** u otras que se consideren necesarias.

El siguiente código muestra un ejemplo clásico donde la etiqueta **<section>** trabaja conjuntamente con la etiqueta **<article>** dentro de una zona de la web.

```
<section>

  <article>

    <h2> Google Chrome </h2>

    <p> Google Chrome es gratuito. </p>

  </article>

  <article>

    <h2> Safari </h2>

    <p> Safari es de Pago. </p>

  </article>

</section>
```

- **<aside>**: Se utilizará para introducir contenido en la web que NO se considera principal. Un uso habitual es para bloques publicitarios o temática no relacionada directamente con la temática de la web como enlaces externos o calendario de eventos. El uso puede ser muy variado.

```
<aside>
```

```
    <h4> Bloque publicitario </h4>
```

```
    <p> Un ejemplo de la etiqueta es para publicidad </p>
```

```
</aside>
```

- **<footer>**: Se utiliza para englobar todo el contenido que se ubica en la parte inferior de la web como el copyright, autor, política del portal, publicidad, formas de pago, etc... Es un pie de página y tiene la misma función que **<header>** pero en la parte inferior.

Igual que el resto de las etiquetas semánticas, esta etiqueta ayuda a realizar una correcta indexación de contenidos por parte de los motores de búsqueda.

```
<footer> // Pie de la página.
```

```
    <p> Información contacto:
```

```
        <a href="mailto:correo@hp.com"> someone@example.com </a>
```

```
    </p>
```

```
</footer>
```

- **<details>**: Se utiliza para **definir detalles adicionales de la página** Web. Este elemento crea una herramienta que se expande cuando hace clic en ella para mostrar información adicional La parte visible se define con la etiqueta **<summary>**.
- **<summary>**: Define el título para la etiqueta **<details>**. Esta etiqueta una vez introducida desplegará un pequeño contenido cuando sea pulsada.

El atributo '**open**' fuerza a desplegar el contenido de los detalles ya que por defecto será necesario pulsar el triángulo negro para poder visualizarlo.

```
<details open>
```

```
<summary> Copyright 1999-2014.</summary>
```

```
<p> Todos los derechos reservados </p>
```

```
</details>
```

▼ Copyright 1999-2014.

Todos los derechos reservados

- **<figure>**: Esta etiqueta no solo sirve para insertar figuras o imágenes, sino que permite la inserción de cualquier elemento externo. Son elementos que tienen que ver con el contenido, pero son independientes, como las imágenes. Los navegadores asignan por defecto márgenes laterales a esta etiqueta.

A continuación, se muestra un ejemplo incluyendo también la etiqueta **<figcaption>**. Esta etiqueta no es imprescindible, pero si deseamos ver la web en diferentes dispositivos con esta etiqueta le decimos el tipo de elemento que se desea mostrar.

- **<figcaption>**: Se utiliza para etiquetar elementos. Siempre irá a continuación del objeto y dentro de la etiqueta **<figure>**.

```
<figure>
```

```

```

```
<figcaption> Visita Pulpit Rock en Noruega </figcaption>
```

```
</figure>
```



Fig.1 - Vista Pulpit Rock en Noruega.

- **<picture>**: Esta etiqueta se utiliza para **insertar una imagen** dentro de una zona del documento. Trabaja en combinación con la etiqueta **<source>** para **ofrecer múltiples imágenes** en diferentes resoluciones. Esta etiqueta carga diferentes imágenes y el navegador selecciona la más adecuada en cada momento según la configuración especificada del dispositivo.

En el primer ejemplo, la etiqueta **<picture>** dispone de tres imágenes cargadas en su interior. Por defecto aplicará la imagen que se encuentra con la etiqueta **** (*flowers1.jpg*), pero en el momento que el ancho de la pantalla de visualización supere los 465px cargará la imagen *'flowers2.jpg'*, y si aumenta hasta superar los 650px asignará la imagen *'flowers3.jpg'*.

Ejemplo 1:

```
<picture> // Imagen según viewport
  <source media="(min-width: 650px)" srcset="flowers2.jpg">
  <source media="(min-width: 465px)" srcset="flowers2.jpg">
  
</picture>
```

En el segundo ejemplo disponemos de 3 imágenes. Por defecto se cargará la imagen *'default_landscape.jpg'* de la etiqueta ****. El resto de las imágenes se cargarán en función de si cumplen o no los requisitos del atributo **'media'** (media queries). En estos requisitos primero se evalúa el ancho de la pantalla (si es superior a 450px o 750px) y posteriormente la orientación de la página o dispositivo (*apaisada – landscape, o vertical – portrait*).

Ejemplo 2:

```
<picture>
  <source srcset="normal_landscape.jpg" media="(min-width: 750px)
  and (orientation: landscape)">
  <source srcset="normal_portrait.jpg" media="(min-width: 450px)
  and (orientation: portrait)">
  
</picture>
```

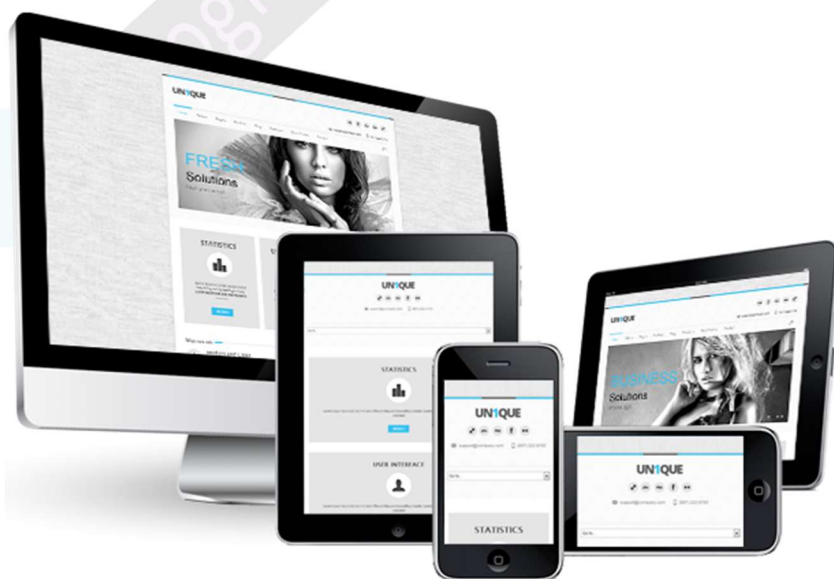

Es importante tener presente la gran cantidad de formatos de salida de una página web. Esto puede variar de forma considerable entre marcas y modelos. Disponer de una **página 'Responsive' que se adapte el máximo número de dispositivos** es fundamental para llegar al mayor número de usuarios.

Es complicado que nuestra web funcione perfectamente en todos los tipos de dispositivos y con todos los navegadores existentes. Aunque todos los navegadores son intérpretes de código no todos lo realizan de la misma forma, de ahí que muchas páginas se visualicen con pequeños cambios entre diferentes navegadores.

En los últimos años los Smartphones con grandes pantallas y resoluciones muy elevadas están convirtiendo a este dispositivo en un criterio fundamental a la hora de diseñar nuestra página o aplicación web. Si nuestra página o aplicación web será visualizada mayoritariamente en dispositivos móviles nuestro diseño le dará prioridad ('**mobile first**').

Responsive Web Design: El diseño de páginas web utilizando esta característica hace que las páginas se visualicen correctamente en todo tipo de dispositivos. Por definición, una página web debe verse bien en cualquier dispositivo (tablet, móvil, ordenador, etc...).

Para poder realizar una web que se adapte a cualquier dispositivo se utilizan las hojas de estilo CSS y se configuran, entre otras muchas cosas, las unidades para que sean relativas a los elementos y al espacio disponible (porcentajes, em, etc..).



Técnicas de diseño web: Existen cuatro formas de realizar el diseño de una web.

1. **Tablas en HTML:** **Sistema en desuso** por su complicación para configurar y modificar a través de las hojas de estilo, son muy poco flexibles. Inicialmente era la forma de diseñar las páginas web generando mucho trabajo de codificación. Actualmente las tablas se utilizan para **mostrar datos tabulados** que se extraen de bases de datos mediante PHP u otros sistemas (Python, Ruby).
2. **CSS con elementos flotantes:** Es el sistema clásico que permite ubicar los elementos de forma 'flotante' en la web dependiendo del dispositivo donde se visualiza la web. Es similar al sistema CSS FlexBox pero menos evolucionado. Es una buena solución si nuestra página no está excesivamente sobrecargada de elementos.
3. **CSS Frameworks:** Son librerías de elementos contruidos que permiten generar páginas web de forma rápida, con un diseño muy bueno y sin tener que configurar el código de la página a nivel de formato. Bootstrap, Foundation, Kube y otros han tenido un gran auge en los últimos años. Entre desarrolladores hay tantos que están a favor de su utilización como en contra, aquí se abre uno de los grandes debates, *'Frameworks so o frameworks no'*.
4. **CSS FlexBox:** Es un nuevo modelo que se basa en un sistema de cajas o contenedores optimizado para los diferentes dispositivos que usan los usuarios de una web.
5. **CSS Grid:** Es un sistema de distribución de elementos utilizando un sistema de rejilla, es decir, se configurar una cuadrícula o rejilla y dentro de esta se van introduciendo los diferentes elementos como cajas normales o flexibles, según el diseño.

Multimedia – Video y Audio

Debido a las inconsistencias entre distintos navegadores durante muchos años, era preciso utilizar las etiquetas **<object>** (que formaba parte de la recomendación oficial de HTML) y **<embed>** (que no formaba parte de la recomendación oficial) para poder introducir elementos de video o audio dentro de una página web. El código que se debía generar era demasiado extenso para cubrir todas las posibilidades existentes tanto por los diferentes tipos de formatos y navegadores.

Insertar un video o audio actualmente es tan sencillo como:

Ejemplo 1: `<video src="excursion.ogg"> </video>`

Ejemplo 2: `<audio controls autoplay>`

`<source src="discurso.ogg" type="audio/ogg">`

`<source src="discurso.mp3" type="audio/mpeg">`

Texto si el navegador no puede mostrar el audio.

`</audio>`

Todo y con eso las etiquetas **<video>** y **<audio>** presentan un problema en la actualidad que es la **guerra de códecs** existentes entre fabricantes ya que el soporte de estos códecs no es uniforme para todos los navegadores incluyendo a los más populares.

Códec: (Codificador – Decodificador). Es un programa o dispositivo hardware capaz de codificar y decodificar una señal o flujo de datos digitales, es decir, convierte los datos digitales en secuencias de video o audio para que los humanos las podamos entender. Hay multitud de códecs, pero en diseño web los más utilizados son:

- *Ogg Theora*
- *H.264/MPEG-4* (el códec más compatible actualmente)
- *VP8/WebM*

Aspectos a tener en cuenta a la hora de diseñar páginas utilizando estas etiquetas.

- NO todas las versiones de los navegadores (sobremesa, tablet, móvil, etc.) dan soporte al 100% a estas etiquetas.
- Los elementos multimedia suelen venir en muchos formatos diferentes debido a la poca estandarización que ha habido desde sus inicios. Una misma página web puede contener diferentes tipos de formatos de archivo. Es recomendable **unificar formatos** del mismo tipo de información.
- Algunos formatos **requerían (plugins)** para poder ser ejecutados como, por ejemplo, Flash (archivos con extensión .SWF). Actualmente hay muchas páginas que precisan de estos programas externos para visualizar los archivos multimedia correctamente, pero la tendencia actual es que vayan desapareciendo para convertirlo en un estándar. Mozilla, por ejemplo, ha deshabilitado por defecto todos los elementos que necesitan plugins para ser ejecutados y es el usuario quien debe activarlos si lo desea.
- HTML5 pretende simplificar y estandarizar la gestión de archivos multimedia. Aún no ha llegado al final del camino en términos de estandarización, pero ese es el objetivo final.

Los formatos de video reconocidos que soportan y tienen compatibilidad 100% con el estándar HTML5 son .OGG, .WEBM, y .MP4. Además de estos formatos hay otros tipos que se deben reconocer porque aún hay gran número de páginas que los contienen y los utilizan.

Archivo Video	Descripción
.MPEG	Muy popular, utilizado por todos los navegadores. Todo y con eso no está soportado por el estándar HTML5. De todas formas es posible que todos los navegadores reconozcan el formato.
.AVI	Formato de Microsoft. Utilizado en cámaras, videos y SO Windows, pero no soportado por todos los navegadores.
.WMV	Formato de Microsoft. Utilizado en cámaras, videos y SO Windows, pero no soportado por todos los navegadores.
.MOV	Quicktime, desarrollado por Apple. Funciona bien en dispositivos Apple, fuera de estos sistemas operativos será necesario instalar el plugin Quicktime.

.RAM	Real Video. Diseñado para video streaming, aún se utiliza pero no ha recibido el visto bueno de HTML5, por tanto, es posible que acabe desapareciendo.
.SWF	Flash. Desarrollado por Adobe. Necesita plugins para ser visualizado. Tiene los días contados.
.OGG	Desarrollado por Xiph.Org Foundation (organización sin ánimo lucro que crea formatos multimedia y software libre) Soportado por el nuevo estándar HTML5.
.WEBM	Desarrollado por los gigantes web (Mozilla, Google, Adobe y Opera).Compatible 100% con HTML5.Es el recomendado actualmente.
.MP4	Desarrollado por el Moving Pictures Expert Group. Soporta HTML5 y es recomendado por Youtube. El éxito radica en que muchas páginas cargan sus videos desde enlaces de Youtube.

Los formatos de audio que soportan y tienen compatibilidad con HTML5 son .MP3, .WAV, y .OGG.

Además hay otros formatos que se deben reconocer porque muchos navegadores los pueden aceptar:

Archivo Audio	Descripción
.MIDI	Primer formato digital de audio. Funciona bien en todos los sistemas operativos pero no en los navegadores.
.WMA	Desarrollado por Microsoft. Funciona bien en sistemas operativos Windows, pero no por todos los navegadores.
.AAC	Desarrollado por Apple para ellos y formato de iTunes. No soportado por todos los navegadores web.
.WAV	Desarrollado por IBM y Microsoft. Funciona bien en todos los sistemas operativos. Soportado por HTML5.
.OGG	Desarrollado por Xiph.Org Foundation (organización sin ánimo lucro que crea formatos multimedia y software libre) Soportado por HTML5.
.MP3	Formato comprimido de alta calidad. Soportado por todos los navegadores. Soportado por HTML5.
.MP4	Es de video, pero también puede ser utilizado por audio. Tiene soporte en todos los navegadores.

El siguiente código muestra la inserción de un elemento de video. Se introduce el archivo en 3 formatos y **el navegador ejecuta el más adecuado**. No todos los navegadores reproducen todos los formatos.

```
<video width="320" height="240" controls autoplay loop  
poster="/imagen/fondo.gif" preload="none"/>  
  
    <source src="movie.mp4" type="video/mp4">  
  
    <source src="movie.ogv" type="video/ogg">  
  
    <source src="movie.webm" type="video/webm">  
  
    <p> Texto si el navegador no puede mostrar el video. </p>  
  
</video>
```

- Los atributos '**width**' y '**height**' especifican el tamaño de la ventana que contendrá el archivo de video. Se debe intentar establecer cierta proporcionalidad entre el archivo y la ventana.
- El atributo '**controls**' perteneciente a la etiqueta **<video>** permite mostrar una **serie de controles** debajo del video (play, pause, volumen, línea de reproducción, y descarga). Los controles se pueden insertar si no deseamos que se reproduzca de forma automática.
- El atributo '**autoplay**' comenzará a reproducir el archivo en el momento que esté disponible. El video primero se tiene que descargar y posteriormente reproducir.
- El atributo '**loop**' realiza la reproducción ininterrumpida del video mientras no se pause.
- El atributo '**src**' de la etiqueta **<source>** enlaza con el video que se desea cargar. Es posible incluir el mismo video en diferentes formatos por si el navegador no es capaz de reproducir el archivo. En este caso siempre mostrará el primer video que pueda ejecutar.
- El atributo '**type**' de la etiqueta **<source>** especifica el **formato del archivo a reproducir**. Aun es necesario informar de la extensión del archivo que va a ser descargado.
- El texto inferior solo se mostrará si no se pudiese reproducir ningún video. Es un valor por defecto que se carga para informar de la incidencia al usuario sino solo se mostraría la caja.

Atributos adicionales para la etiqueta <video>:

- '**muted**' especifica que la salida de video o audio debe estar silenciada. No siempre es conveniente realizar un 'autoplay' con el audio activado.
- '**poster**' especifica **la imagen de espera que se mostrará mientras se carga el video** o hasta que sobre el video se pulse el botón de reproducción.
- '**preload**' especifica cuando y como se debe descargar el video en la página. Una página con muchos videos es posible que no sea necesario cargarlos todos a la vez en la web ya que puede ralentizar la carga del sitio.

```
<video preload="auto|metadata|none">
```

Atributos adicionales para la etiqueta <source>:

- '**media**' determina el tipo de dispositivo donde será visualizada la página web. Este tipo de acción se suele realizar desde CSS, pero también es posible realizarlo desde esta etiqueta. Los valores más utilizados en esta etiqueta para los diferentes tipos de dispositivos son:
 - All: Disponible para todos los dispositivos, es la opción por defecto.
 - Screen: Pantallas de ordenadores.
 - Tv: Dispositivos de Tv, baja resolución y limitaciones de desplazamiento con barra.
 - Projection: Para proyectores de video.
 - Aural: Sintetizadores de voz.
 - Braille: Dispositivos para Braille
 - Handheld: dispositivos pequeños, de pantalla pequeña y ancho banda limitado.
 - Print: Modo vista preliminar.

Además, es posible combinar el tipo de dispositivo del atributo '**source**' con una serie de valores relacionados con el aspecto, orientación, tamaño, etc... Estos valores serán enlazados con la conjunción 'and'. A continuación, se muestran unos ejemplos con los posible valores y tipos de dispositivos.

Ancho mínimo.

```
<source media="screen and (min-width:500px)"
```

Alto mínimo.

```
<source media="screen and (max-height:700px)"
```

Especifica el ancho de la pantalla de destino.

```
<source media="screen and (device-width:500px)"
```

Especifica el alto de la pantalla de destino.

```
<source media="screen and (device-height:500px)"
```

Puede tener dos valores de orientación 'portrait' o 'landscape'

```
<source media="all and (orientation: landscape)"
```

Especifica el radio de ancho-alto de la pantalla de destino.

```
<source media="screen and (aspect-ratio:16/9)"
```

Especifica los bits por color de la pantalla de destino

```
<source media="screen and (color:3)"
```

Especifica el número de colores que puede utilizar la pantalla de destino.

```
<source media="screen and (min-color-index:256)"
```

Especifica la densidad de píxeles del dispositivo de visualización

```
<source media="print and (resolution:300dpi)"
```

Especifica el dispositivo de salida '*interlace*' o '*progressive*'

```
<source media="tv and (scan:interlace)"
```

Especifica si la salida del dispositivo es 'grid' o 'bitmap'

```
<source media="handheld and (grid:1)"
```

La etiqueta **<track>** se utiliza para **especificar los subtítulos** y otros elementos que deben ser visibles cuando los medios de comunicación se están reproduciendo.

```
<video width="320" height="240" controls>

  <source src="forrest_gump.mp4" type="video/mp4">

  <source src="forrest_gump.ogg" type="video/ogg">

  <track src="subtit_en.vtt" kind="subtitles" srclang="en" label="English">

  <track src="subtit_es.vtt" kind="subtitles" srclang="de" label="Deutsche">

</video>
```

- **'kind'** especifica el tipo de elemento adicional que complementa al video *'captions'* (sordos), *'chapters'*, *'descriptions'* (ciegos), *'metadata'* (metadatos no visibles) y *'subtitles'* (este claramente es el más utilizado).
- **'srclang'** especifica **el idioma de los datos del texto** de la pista. Es preciso si el atributo *'kind'* tiene la opción marcada *'subtitles'*.
- **'title'** especifica el título de la pista de texto y lo utiliza el navegador cuando enumera las pistas de texto disponibles.
- **'src'** determina el archivo que contiene el texto con los subtítulos del video. Este archivo está formado por **'cues'** que son intervalos que determinan en que momento debe mostrarse el texto del subtítulo dentro del video. La extensión más utilizada es **.vtt** y el archivo se puede configurar desde HTML o JavaScript. A continuación, se muestra un fragmento de un archivo **.vtt**.

```
WEBVTT FILE

railroad
00:00:10.000 --> 00:00:12.500
Poco inspirados por la corteza de tierra de ferrocarril

manuscrito
00:00:13.200 --> 00:00:16.500
que tocó el plomo hasta las páginas de tu manuscrito
```



Cada 'Cue' dispone de un identificador, del tiempo que será mostrado, y del texto que contendrá.

Los archivos '*cue*' no son exclusivos del entorno web, y técnicamente es cualquier archivo con texto Unicode que sea capaz de especificar como se distribuyen las pistas de datos **<tracks>**. La siguiente imagen muestra un archivo '*cue*' para pistas de audio.

```
TITLE "Live in Berlin, 1998"
PERFORMER "Faithless"
FILE "faithless - live in berlin.mp3" MP3
TRACK 01 AUDIO
  TITLE "Reverence"
  PERFORMER "Faithless"
  INDEX 01 00:00:00
TRACK 02 AUDIO
  TITLE "She's My Baby"
  PERFORMER "Faithless"
  INDEX 01 06:42:00
TRACK 03 AUDIO
  TITLE "Take The Long Way Home"
  PERFORMER "Faithless"
  INDEX 01 10:54:00
```

Otra opción muy extendida es la utilización de los archivos '*.srt*'. Estos se usan para proporcionar subtítulos a un video **en forma de archivo separado**, en lugar de codificarlo directamente dentro del propio video. Esto genera un tamaño de archivo marginalmente menor y deja los subtítulos completamente opcionales para los usuarios que deseen utilizarlos. Algunos editores de video permiten incrustar o fusionar un archivo '*.srt*' o '*.vtt*' con un determinado video.

El audio solo se podía introducir con anterioridad a través de plugins (como Flash o QuickTime). Desde HTML5 la etiqueta **<audio>** especifica el estándar para introducir sonido en una web. La tendencia

```
<audio controls autoplay>
```

```
<source src="horse.ogg" type="audio/ogg">
```

```
<source src="horse.mp3" type="audio/mpeg">
```

Texto si el navegador no puede mostrar el audio.

```
</audio>
```

Los atributos de la etiqueta **<audio>** son los mismos que la etiqueta **<video>**. Los atributos son '*autoplay*', '*controls*', '*muted*', '*preload*', '*src*'.

Los atributos para la etiqueta **<source>** dentro de la etiqueta **<audio>** también son los mismos que en el caso de la etiqueta **<video>**.

Videos de Youtube en HTML

Es la forma más fácil de reproducir archivos en la web. Históricamente el principal problema de subir archivos a Internet radicaba en tener que convertir los archivos de video a los diferentes formatos si deseábamos que se visualice bien en todos los navegadores. Si esta labor se deja en manos de Youtube solo hay que linkar el enlace desde la página web de Youtube con la etiqueta **<iframe>**.

Los pasos para utilizar el sistema de Youtube son:

1. Subir el video a Youtube
2. Visualizar el identificador del video (el enlace).
3. Definir un elemento **<iframe>** en la página para incrustar el video (recomendado).
4. Introducir un atributo 'src' con el enlace de Youtube.
5. Modificar la altura y anchura del elemento **<iframe>** y cualquier otro atributo en la etiqueta.

```
<iframe width='420' height='315' src="https://youtu.be/IH2"></iframe>
```

Los atributos configurables para videos enlazados desde Youtube son los siguientes:

- **'autoplay'** reproduce automáticamente el video cuando se carga la página. Los valores pueden ser 1 (automáticamente) y 0 (de forma manual).

```
<iframe width="420" height="315"src="https://www.youtube.com/embed/XGSy3_Czz8k? autoplay=1"></iframe>
```

- **'loop'** reproduce automáticamente el video (1) o lo realizara de forma manual (0).

```
<iframe width="420" height="315"src="https://www.youtube.com/embed/XGSy3_Czz8k?playlist=XGSy3_Czz8k&loop=1"></iframe>
```

- **'controls'** muestra los controles de reproducción (1) o no los muestra (0).

```
<iframe width="420" height="315"src="https://www.youtube.com/embed/XGSy3_Czz8k?controls=0"></iframe>
```

Plugins en HTML

Los plugins ("enchufes") son pequeñas aplicaciones que se instalan sobre una aplicación principal dotándola de una **funcionalidad adicional al navegador**. Los plugins más conocidos en web son los applets de Java. Photoshop fue pionero en este concepto de programa extensible mediante módulos.

Applet Java: Es una aplicación creada en el lenguaje Java que se incrusta en un documento HTML. Para poder utilizar los applets en un navegador es preciso activar la Java Virtual Machine. De esta forma es posible mostrar programas realizados con otros lenguajes de programación en una página web como, por ejemplo:

- Un juego de Pinball online.
- Jugar al poker online.
- Una aplicación que genere gráficos a partir de unos datos.

Es posible añadir plugins a una página web (no al navegador) con el propósito de mostrar mapas, buscar virus, verificar identidades, etc... Mediante las etiquetas **<object>** y **<embed>** se pueden añadir plugins directamente en la web. Las dos etiquetas hacen prácticamente lo mismo.

La etiqueta **<object>** es compatible con todos los navegadores y lo que hace es incrustar un objeto dentro de un documento HTML. Se suele utilizar para lectores de PDF, Flash Players, Reproductores y otros. Esta etiqueta permite introducir elementos HTML dentro de elementos HTML.

Incrustar un objeto flash

```
<object width="400" height="50" data="bookmark.swf">
</object>
```

Incrustar una página web

```
<object width="100%" height="500px" data="snippet.html">
</object> // Mejor con etiqueta <iframe>
```

Incrustar imágenes, poco habitual pero también lo permite la etiqueta.

```
<object data="audi.jpeg">
</object> // Mejor con etiqueta <picture>
```

La etiqueta **<embed>** es compatible con todos los navegadores, pero la especificación HTML5 determina que si no es estrictamente necesario se recomienda el uso de otras etiquetas. El siguiente código de programación muestra el mismo código que el utilizado con la etiqueta **<object>**.

```
<embed width="400" height="50" src="bookmark.swf">
<embed width="100%" height="500px" src="snippet.html">
<embed src="audi.jpeg">
```

La etiqueta **<embed>** y **<object>** tienen gran multitud de atributos que se han reducido considerablemente desde la actualización a HTML 5. El resto de atributos o acciones que se podían realizar han quedado absorbidas por CSS o Javascript. Los atributos soportados por HTML5 para estas etiquetas son:

- **'data'** especifica la dirección URL donde se encuentra el recurso.

```
<object width="400" height="400" data="helloworld.swf"></object>
```

- **'form'** permite introducir el objeto dentro de un formulario cuando esta fuera.

```
<form action="/action_page.php" id="form1">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
</form>

<object data="helloworld.swf" form="form1" name="ob1"></object>
```

- **'type'** determina el tipo de dispositivo donde se visualizará el objeto. El tipo de archivos que se pueden incrustar están disponibles en la [IANA Media Types](#) que es un listado con todos los posibles objetos incrustables en un documento HTML. La lista dispone más de 1000 tipos de archivos (imagen, video, audio y aplicación).

```
<object width="400" height="400" data="helloworld.swf" type="application/vnd.adobe.flash-movie"></object>
```

- **'name'** especifica el nombre del objeto para referirse al él.
- **'width'** y **'height'** permite asignar un ancho y un alto al objeto.

```
<object data="hello.swf" width="400" height="400" name="ob1"> </object>
```