

HTML 5

1. Tutorial de HTML 5

HTML son las siglas con las que se conoce el lenguaje creado para construir las páginas que forman un **sitio web**. Hiper Text Mark Language o lenguaje de marcas de hipertexto.

Nació en 1991 derivado de un lenguaje empleado para publicaciones científicas. Desde entonces se han realizado varias modificaciones que dieron lugar a la aparición en 1998 de la versión HTML 4, en la que aparecían los scripts y los estilos CSS.

Desde la versión 4 el HTML ha sufrido pocas modificaciones. Hasta esta última versión, HTML 5, todas las especificaciones del lenguaje han sido realizadas y normalizadas por el consorcio [W3C](#), que detuvo la evolución del HTML en la revisión 4.01 para comenzar a desarrollar un sustituto el XHTML.



Pero internet ha ido ocupando prácticamente todos los sectores de actividad humanos: tanto lúdicos como laborales. La especificación de HTML no se ha adaptado a estos cambios y los diseñadores de páginas han necesitado usar **soportes externos, como los conocidos plugins** para manejar video o audio. Era necesario una revisión a fondo de las especificaciones del HTML.

Ante la **paralización del desarrollo de HTML por parte de la W3C** una comunidad de empresas conocida como [WHATWG](#) desarrolló las primeras especificaciones del actual **HTML5**, posteriormente el consorcio W3C se sumó al proyecto y actualmente existe una coordinación entre ambos organismos. La filosofía del actual HTML5 es no cerrar en una versión, sino mantenerlo en constante evolución. Es decir, al parecer no existirá una versión HTML 6, incluso HTML5 pasará a denominarse simplemente HTML

En la actualidad todas las modificaciones y ampliaciones de HTML vienen especificadas en la web de WHATWG

Esta revisión (que **ya puede considerarse definitiva**) pretende solventar las deficiencias del HTML4 y adaptar este lenguaje a la realidad de los sitios web actuales: **multimedia, conectividad** por medios diferentes (iPad, teléfonos, ordenadores), **manejo de datos, animación, diseño adaptativo**.... Las especificaciones de esta versión se adaptan a la realidad del diseño de páginas web, y algo muy importante: lleva aparejada la **estandarización de los diferentes**

“



HTML 5 será al parecer la última versión numerada del lenguaje, La próxima revisión se llamará HTML e irá siendo enriquecida o modificada sin mostrar un número de versión

1991	HTML
1994	HTML 2
1996	CSS 1 + JavaScript
1997	HTML 4
1998	CSS 2
2000	XHTML 1
2002	Tableless Web Design
2005	AJAX
2009	HTML 5

navegadores. HTML5 pretende que una página web se vea igual con cualquier navegador.



HTML5 sigue siendo un lenguaje de etiquetas, como la versión anterior con la que es totalmente compatible, pero aporta nuevos elementos y elimina otros que se han quedado obsoletos o que eran parches para la especificación original del lenguaje.

Con esta versión del lenguaje se busca que una página web sea visible en cualquier medio, tenga interactividad con el usuario, permita trabajar con o sin conexión, permita manejar el contenido multimedia...

Con **HTML5** la separación entre contenido y formato del documento se hace aún más fuerte. Por ello es indispensable contar con el lenguaje de estilos conocido como CSS (CSS3 en la actualidad). Si las etiquetas usadas por HTML definen la estructura del documento, el CSS define la forma en que el documento va a ser visto.

La tercera pata para crear páginas web con HTML5 es **JavaScript**. La interactividad con el usuario que usa las páginas web queda en manos de este renovado lenguaje, que ahora cuenta con librerías y funciones para interactuar con las entrañas del HTML5, que proporciona para ello interfaces (APIs) que permiten manejar datos o arrastrar y soltar imágenes, o realizar gráficos directamente en la página web o manipular videos.

Los applets Java han desaparecido del diseño web. Por su parte las aplicaciones Flash están prácticamente extintas. Para sustituirlo está el formato gráfico SVG y, sobre todo, el elemento web **Canvas**, con el que realizar aplicaciones es una tarea bastante simple.

2. Nuevas etiquetas en HTML 5

La versión HTML5 supone una modificación profunda de la versión anterior (HTML 4). Según parece es la última versión del lenguaje, es decir no habrá un HTML6, HTML7 etc. El lenguaje pasa a llamarse HTML. Y supone un cambio en profundidad.¹

Hasta ahora las etiquetas servían para describir como se coloca o muestra un contenido (<div>, ,) o cuál es su función (<a href>, <form>). La versión actual extiende la interpretación de las etiquetas, ahora todas tienen un **significado semántico**, es decir, indican el **papel que juegan en la página**. Ejemplo: un texto en no solo se ve en negrita para destacarlo, sino que indica un texto importante en el contenido. O un elemento <nav> significa que su contenido es un menú de navegación.

Recuerda que todas las actualizaciones de este lenguaje se pueden seguir en WHATWG

¹ La versión más reciente de HTML introduce nuevas etiquetas para colocar el contenido en los documentos web. Con la estrategia actual en la definición de HTML no se van a crear nuevas versiones del lenguaje, sino que va añadiendo funcionalidades. No habrá al parecer un HTML6, sino que el actual HTML se va modificando

Tabla resumen de etiquetas²

Etiquetas	Descripción
<article>	Representa una parte independiente del contenido de un documento, como una entrada de blog o un artículo de periódico.
<aside >	Representa una pieza de contenido que solo está ligeramente relacionada con el resto de la página.
<audio>	Define un archivo de audio.
<canvas>	Esto se utiliza para representar gráficos de mapa de bits dinámicos sobre la marcha, como gráficos o juegos.
<command>	Representa un comando que el usuario puede invocar.
<datalist>	Junto con el nuevo atributo de lista para la entrada, se puede usar para hacer cuadros combinados
<details>	Representa información adicional o controles que el usuario puede obtener a pedido
<dialog>	Elemento para facilitar la interactividad con el usuario. Amplía las funcionalidades de <alert>
<embed>	Define contenido interactivo externo o complemento.
<figure>	Representa una pieza de contenido de flujo independiente, normalmente referenciada como una sola unidad del flujo principal del documento.
<footer>	Representa un pie de página para una sección y puede contener información sobre el autor, información de derechos de autor, etcétera.
<header>	Representa un grupo de ayudas introductorias o de navegación.
<hgroup>	Representa el encabezado de una sección.
<keygen>	Representa el control para la generación de pares de claves.
<mark>	Representa una secuencia de texto en un documento marcada o resaltada con fines de referencia, debido a su relevancia en otro contexto.

² El uso de estas etiquetas no es obligatorio, aunque su uso facilita la edición y modificación de las páginas web. las etiquetas deben escribirse en minúsculas. A veces las veras en mayúsculas simplemente para destacarlas.

<meter>	Representa una medida, como el uso del disco.
<nav>	Representa una sección del documento destinada a la navegación.
<output>	Representa algún tipo de resultado, como un cálculo realizado a través de secuencias de comandos.
<progress>	Representa la finalización de una tarea, como la descarga o la realización de una serie de operaciones costosas.
<ruby>	Junto con <rt> y <rp> permiten marcar anotaciones Ruby.
<section>	Representa un documento genérico o una sección de aplicación
<time>	Representa una fecha y/u hora.
<video>	Define un archivo de vídeo.
<wbr>	Representa un salto de línea opcional.

Para Formularios Descripción

datalist	Lista de opciones para entrada en inputs
keygen	Generación de claves y envíos de claves públicas
output	Mostrar un resultado, utilizado por scripts

Campos Input Descripción

color	Selector de color, que podría estar representado por una rueda o un selector de muestras
date	Selector de fecha del calendario
datetime-local	Visualización de fecha y hora, sin configuración ni indicación de zonas horarias
datetime	Visualización completa de fecha y hora, incluida una zona horaria.
email	El tipo de entrada debe ser un correo electrónico.
month	Selector de un mes dentro de un año dado
number	Un campo que contiene solo un valor numérico
range	Selector numérico dentro de un rango de valores, normalmente visualizado como un control deslizante
search	Término a suministrar a un motor de búsqueda. Por ejemplo, la barra de búsqueda en la parte superior de un navegador.
tel	El tipo de entrada debe ser número de teléfono.

time	Indicador y selector de hora, sin información de zona horaria
url	El tipo de entrada debe ser tipo URL.
week	Selector de una semana dentro de un año dado

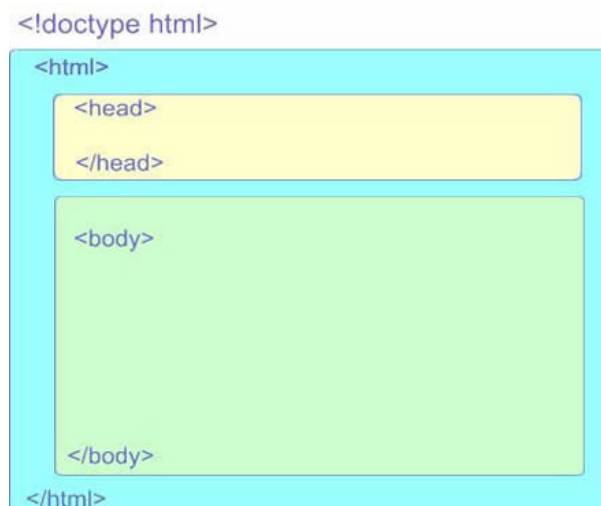
3. Primeros pasos

Las herramientas necesarias son pocas: un **editor de textos** será suficiente. Si dispones de un editor de páginas web como kompozer o similar, pues también lo puedes usar. Se trata de que entiendas y manejes el lenguaje HTML5, luego podrás usar con más eficiencia editores complejos como DreamWeaver (entorno Windows) o una solución de Open Source como Aptana (entorno Mac, Windows o Linux) y otros. aunque lo más extendido actualmente es **Visual Code**.

La otra herramienta necesaria es tener a mano alguna referencia del **lenguaje de estilos CSS3**. Te será más útil a medida que vayas avanzando y haciendo cosas por tu cuenta.

Para empezar, como se dice en los apartados anteriores, al crear una página web es **fundamental cimentarla bien con una estructura clara**. Vemos como se estructura una página web. Sabes lo que es una página web, ahora veremos **como es una página web**.

La página web es un documento HTML con la estructura interna que ves en esta imagen de una página:



La página comienza con la declaración del tipo de documento que se va a visualizar, `<!doctype html>` como ves se trata de una declaración muy simple y fácil de memorizar, es la primera diferencia de HTML5 frente a la versión anterior, HTML4. Cualquier página en HTML5 comienza con esa línea. Luego ves un **contenedor** de nombre `<html></html>` que contiene otros dos **contenedores hijos**:

- **HEAD:** No es visible al mostrar la página web. Contiene información sobre la página y su contenido, así como enlaces a ficheros externos necesarios para mostrar la página web, como los archivos con definiciones de estilos CSS o scripts JavaScript. Es aquí donde le vas a colocar el título de la página, el que

se ve en la ventana del explorador, la descripción resumida del contenido, las llamadas a archivos externos necesarios (CSS y JavaScript).

- **BODY:** Es el contenido, estructura y apariencia de la página web. Es lo que se ve en el navegador. Contiene todo lo que va a ser visto en la ventana del explorador. Aquí colocarás tablas, bloques de texto, imágenes, videos.

Como ves cada parte viene marcada con una etiqueta con dos límites: uno de apertura y otro de cierre. La apertura es una palabra encerrada entre `<>` y el cierre la misma palabra encerrada entre `</>`. Algunas etiquetas no necesitan el cierre. Ya las irás viendo.

4. HTML: Como crear páginas Web

Cuando hablamos de crear una web con **HTML**³ realmente usamos un conjunto de lenguajes:

- **HTML** propiamente dicho, lenguaje para definir la estructura y contenido de las páginas web.
- **CSS** conjunto de reglas para definir **cómo** mostrar la página y su contenido
- **JavaScript** lenguaje de programación para hacer páginas dinámicas e interactivas.

Si hablamos de páginas web dinámicas basadas en datos variables, como un blog o una página de comercio electrónico, habría que añadir dos elementos más al proceso de creación:

- **PHP** lenguaje de programación genérico orientado a la web.
- **MySQL** lenguaje para la gestión de la información almacenada en una base de datos.

Todas estas tecnologías trabajan juntas para presentar una página web en pantalla o para crear una aplicación de escritorio. HTML5 es más que una simple revisión de la versión HTML4.



³ HTML 5 es un lenguaje semántico. Una etiqueta con significado **semántico** indica que al aplicarla a un contenido de la página web, define el papel que tiene ese contenido en la estructura de toda la página.

Las etiquetas en la página web.

La página web tiene un **contenido** (la información) y unas **etiquetas** o marcas especiales para indicar al navegador como se debe mostrar esa información. Con las etiquetas se describen los elementos que forman la página web y con ello como se va a mostrar su contenido.

Y las etiquetas no son más que marcas para describir partes de la página o formatos de texto. Las reconocerás porque aparecen encerradas entre ángulos <>, así

```
<strong>esto es negrita</strong>
```

Indica que el texto encerrado entre **** y **** aparece en negrita.

Suelen ir por pares: apertura **** y cierre **** (fíjate en la barra /). Hasta HTML 4 la cosa era así de simple.

Pues bien: **HTML5** incrementa el **sentido semántico** de sus etiquetas. Es decir, **las etiquetas tienen significado en sí mismas** para definir la función de los elementos de la página web, es decir, para que sirven esos elementos. Por si no queda claro, si queremos indicar que una palabra es el título de la página podemos modificar el tamaño de la letra, su color su grosor mediante estilos:

```
<div id="titulo" style="font-size: 20px; font-weight:bold">Zona de juegos</div>
```

Y vale, este título aparecerá en negrita y con un tamaño grande. Visualmente se ve que es el título. Pero si ponemos

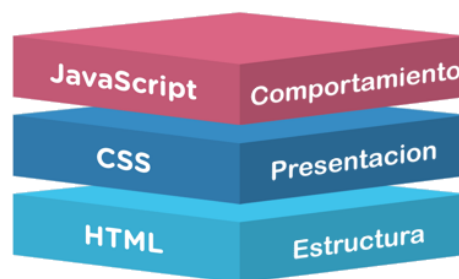
```
<h1>Zona de juegos</h1>
```

“**Zona de juegos**” aparecerá en la página destacado, como antes, pero la etiqueta **<h1>** indicará al explorador, o al robot que analice nuestra página, que esto es un titular importante en la estructura de la página. Esto es lo que significa que la etiqueta (**<h1>** en este caso) tiene **significado semántico propio en cuanto a la estructura de la página**. La etiqueta **<h1>** destaca el texto y además indica la función de ese texto dentro de la página (es un encabezado en este caso)

Para usar HTML5 es muy importante plantear claramente la estructura del documento, es como asegurar los cimientos de un edificio. Esta cimentación no solo permite que las páginas se comporten de manera similar en cualquier entorno de visualización (en cualquier explorador, en el pc de sobremesa, una tablet o un móvil por ejemplo) sino que facilita la tarea a los robots que indexan nuestra web en los buscadores.

Las etiquetas por si solas no siempre proporcionan información sobre como se verá la página web en una pantalla, para esto necesitamos la ayuda de las hojas de estilos **CSS** (la versión actual es la 3). No basta con conocer las etiquetas y reglas del **HTML5**, es necesario conocer también como usar las definiciones de estilo que **nos permitirán organizar y dar formato al contenido**.

La actual versión de las hojas de estilos **CSS** es sumamente potente y permiten incluso animaciones. Un buen uso de los estilos permite cambiar todo el aspecto (no solo colores) de un sitio web cambiando solo un archivo, y lo que es más importante: permiten que la **página web se muestre adaptada al dispositivo** (teléfono, móvil,



tablet, pc de sobremesa, televisión, impresión...). Esto último se conoce como diseño **responsive**.⁴

Por último, si queremos sacar el máximo jugo de nuestro trabajo al diseñar páginas web debemos manejar el lenguaje de programación **JavaScript**. Con su ayuda podemos utilizar las **APIS** (interfaces de programación) que nos ofrece **HTML5**. Por ejemplo, si usamos la etiqueta **<video>** para insertar videos en nuestra página podemos permitir que el usuario pare, proyecte, modifique sonidos, etc. O se pueden manejar datos desde nuestra página. Para esto nos ayudaremos de **JavaScript**. El manejo de estas funciones es para usuarios con buenas competencias en programación.

5. Creando una página Web

Hemos visto que la estructura elemental de una página web son en esencia documentos en html y las etiquetas mínimas necesarias. Ahora verás el código que debes escribir para **crear una página web** usando HTML (la versión HTML 5 simplifica bastante el código de las páginas web).

```
<!doctype html>
<html lang="es-ES">
<head>
<meta charset="utf-8">
<title>mi primera página</title>
</head>
<body>
  Hola Mundo: Este es el contenido de mi primera página web. Mi
  Hola Mundo.
</body>
</html>
```

Este es el contenido de una página web básica. Este código es más simple que el usado en versiones anteriores de HTML.⁵

La primera línea no necesita más explicaciones: solo define el tipo de documento. Es mucho más simple que la usada por los documentos escritos en HTML4.

<html></html> Marca el inicio y fin del documento. Como ves lleva un atributo **lang** con el valor "es-ES". Este atributo es opcional. Indica el idioma en que se escribe la página web. Puede usarse en otros bloques de la página web.

<head></head> marca el inicio y fin de la cabecera del documento. Dentro va información sobre la página web, en este caso el título de la página, pero no se mostrará en el navegador.

⁴ Un buen conocimiento de HTML5, CSS y JavaScript te va a permitir crear webs con diseño responsive

⁵ El elemento **charset** nos asegura que los caracteres especiales del idioma se muestran correctamente. utf-8 asegura la compatibilidad del idioma en que esté escrita la página

<title></title> Etiquetas que encierran el título de la página que se mostrará en la barra del navegador. Este valor debe cuidarse pues es útil para que los robots de los buscadores localicen la página y la sitúen en un buen lugar.

<body></body> Marca la parte del documento donde se pone el contenido de la página web en sí, es decir, lo que se mostrará en el navegador.

Fíjate en las etiquetas usadas en esta página básica, verás que:

- Las etiquetas tienen dos partes una de **apertura** y otra de **cierre** (salvo excepciones) rodeando al elemento al que afectan.
- Una etiqueta puede llevar atributos: valores que modifican su función
- La apertura se escribe entre dos ángulos <head>, <body>, <p>.
- El cierre se escribe entre ángulos pero con la barra invertida antes del nombre </head>, </body>, </p>.
- Las etiquetas no se pueden solapar, pero pueden estar contenidas unas en otras: <title> está contenido en <head>.

```
<head>
<title>
</head>
mi primera página</title>
<body>
```

Las etiquetas de cierre en muchas etiquetas pueden omitirse según el estándar si en el contexto se deduce el cierre, por ejemplo que marca el final de un elemento en una lista podría omitirse si a continuación de ese elemento aparece otro. Es decir que la etiqueta de cierre está implícita, aunque pueda no estar escrita, por ello lo aconsejable es **escribir siempre las etiquetas de cierre** de forma explícita.

```
<ul>
  <li>Advertencias</li>
  <li>Contenido</li>
</ul>
<!-- sería igual que -->
<ul>
  <li>Advertencias
  <li>Contenido
</ul>
```

Los navegadores web pueden dejar pasar algunos errores de sintaxis en el código de la página, pero no debes fiarte de esto. Que tu página se vea bien no es suficiente, debe estar bien escrita. Por tanto, intenta siempre **escribir el código web sin errores** de ningún tipo. Hay editores que advierten de los fallos de sintaxis que se puedan cometer, y además la W3C tiene herramientas online para ayudarte a detectar errores por mal uso de las marcas HTML y CSS ([W3C validator](#)).

Dentro de HEAD: metadatos

El apartado **HEAD**⁶ del documento HTML contiene información útil para el intérprete del documento web (el navegador) o para sistemas que rastreen nuestra página. Son datos que no se van a ver en el navegador, pero que contienen información

⁶ Los estilos o los scripts pueden ir también fuera de la sección HEAD. A veces se hace esto para disminuir el tiempo de acceso a la página

sobre la página web, son los llamados **metadatos**. Contiene varios tipos de etiquetas. Ya has visto alguna, como la etiqueta **<title>**, ahora veremos otras: **<meta>** y **<links>**.

Las etiquetas **<meta>** proveen información extra sobre la página y puede tener diferentes funciones. Como ves no necesitan etiqueta de cierre. Las más usadas

<meta charset="UTF-8"> Indica el juego de caracteres usados en la página. Los más comunes son UTF-8 (Unicode) y ISO-8859-1 (Caracteres latinos). Los admitidos puedes [verlos aquí](#).

<meta name="Description" content=" "> Hace un resumen breve de la página, una descripción útil para directorios y buscadores de páginas web, aunque no incide mucho en el posicionamiento dentro de los resultados de búsqueda.

<meta http-equiv="refresh" content=" " url=" "> Útil para recargar la página actual o enviar al visitante a otra (indicada en el valor opcional **url**) después de transcurridos el número de milisegundos indicados en **content**.

<meta name="viewport" content="width=device-width, user-scalable=no"> Utiliza en el diseño de web responsive con el objeto de ajustar el ancho de la página al tamaño de la pantalla del dispositivo con el que la vemos.

Luego algo menos usadas:

<meta name="author" content=" "> Contiene el nombre del autor de la página web.

<meta name="generator" content=" "> Se puede usar para escribir el programa usado para crear la página.

Ignorables:

<meta name="keywords" content=" "> En content escribimos una lista separada por comas con las palabras más significativas de nuestra página. Era útil para buscadores, pero hoy tiene poco valor. La pongo para que la conozcas por si la ves, pero puedes ignorarla.

Encontrarás otras muchas meta etiquetas para objetivos muy concretos, como las etiquetas destinadas a que las redes sociales extraigan información para mostrar cuando una página es compartida. En este ejemplo tienes las meta propias de la red Facebook y las de Twitter. Son etiquetas utilizables por ambas redes definidas en el [Protocolo Open Graph](#), de donde viene la inicial **og**: que ves en el código de ejemplo:

```
<meta property="og:title" content="Desarrollo Web">
<meta property="og:description" content="Aprende a crear sitios web">
<meta property="og:image"
content="https://creatuweb.com/thumbnail.jpg">

<meta property="og:url" content="http://creatuweb.com/index.htm">

<meta name="twitter:card" content="summary_large_image">
<!-- Recomendables -->
<meta property="og:site_name" content="Aprende a crear sitios web">
```

```
<meta name="twitter:image:alt" content="texto alta para la imagen">
<!-- Necesarias para Analiticas de cada red-->
```

```
<meta property="fb:app_id" content="app_id_del_sitio" />
```

```
<meta name="twitter:site" content="@website-username">
```

(Ejemplo de etiquetas meta para redes sociales)

Recursos y definiciones

Esta sección también incluye recursos⁷ que pueda necesitar la página, tanto externos como definidos in situ.

Los recursos externos se incluyen con una etiqueta **<link>**, con ella se incorpora a la página web recursos contenidos en archivos externos, como estilos, códigos JavaScript, imágenes o iconos. Estos recursos se localizan mediante el atributo **href**, que contendrá una url señalando al archivo a incluir. El tipo de recurso enlazado se especifica en el atributo **rel**.

Un recurso externo habitual son las hojas de estilo, archivos con definiciones de formato para los elementos de la página: colores, tipos de letras, márgenes, bordes, etc., por ejemplo

```
<link href="/css/estilos.css" rel="stylesheet" type="text/css">
```

Esta línea diría que la página web necesita el archivo **estilos.css** situado en la carpeta **css** del sitio web (/css). El argumento **<type="text/css">** indica el tipo de archivo enlazado, en caso de archivos css no es necesario escribirlo.

Otro elemento habitual es el marcado con las etiquetas **<script src=""></script>** para incluir un archivo JavaScript, es decir, con código para ejecutar en la página. Por ejemplo

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

Esta línea diría a la página que cargue el archivo **codigomenu.js** situado en la carpeta **/scripts** del sitio web (se usa el atributo **src**). Este elemento puede colocarse en cualquier parte de la página, no solo en **HEAD**.

Y los recursos internos serían los definidos en la propia página.

Tenemos las etiquetas **<style></style>** para contener definiciones de estilo CSS

Y la etiqueta **<script></script>** para incluir código en la propia página.

```
<head>
<meta charset="UTF-8">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1, user-scalable=no">
```

⁷ Una página web suele necesitar de archivos externos con recursos como fuentes de letras, reglas de estilo CSS y scripts

```

<title>Grupo Laweb</title>
<link rel="shortcut icon"
href="http://www.grupoweb.com/favicon.ico">
<link
href="http://fonts.googleapis.com/css?family=Roboto:400,300,700"
rel="stylesheet" type="text/css">
<link rel="stylesheet"
href="assets/bootstrap/css/bootstrap.min.css">
<link rel="stylesheet" href="assets/animate/animate.css">
<link rel="stylesheet" href="assets/animate/set.css">
<link rel="stylesheet" href="assets/style.css">
<script>
var link = document.createElement('link');
link.type = 'image/x-icon';
link.rel = 'shortcut icon';
link.href = 'favicon.ico';
</script>
</head>

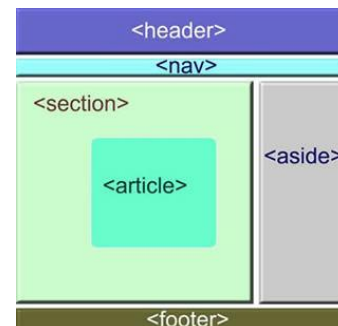
```

(Ejemplo de una sección Head de una página web)

El cuerpo: BODY y Semántica HTML5

Después de crear la sección **HEAD** del documento el siguiente paso será definir el cuerpo: la parte visible de todo documento, lo que mostrará el explorador cuando acceda a tu página, en inglés se conoce como contenido **flow content**. Está señalado por la etiqueta **<body></body>**.

Actualmente nuestra página puede ser visitada a través de múltiples medios: ordenadores de sobremesa, tabletas, teléfonos..., por ello es muy importante que el navegador interprete correctamente la estructura del documento. Esta es la principal novedad del HTML5: las etiquetas con significado relativo a la **estructura del documento**, no solo describen como se verá el contenido en pantalla. Quienes ya conocen HTML: las tablas y los bloques div siguen existiendo pero tienen otra finalidad. Las tablas son para presentar datos, catálogos, etc. Y los **div son bloques genéricos para organizar el contenido**. Esas son las recomendaciones⁸, la página va a funcionar, aunque no las sigas.



Las partes de body: semántica HTML 5

Como apuntábamos en la descripción del lenguaje html, los documentos web tienen etiquetas con significado que van más allá de organización y formateo de la página. Poseen un significado semántico: indican también el papel que juega cada parte en la página, si es un título o una zona para menús o barras de navegación, etc.

- **Main:** Es un bloque que enmarca el contenido principal de la página, la información que se pretende comunicar en el documento. Dentro pueden existir otros bloques anidados, pero el bloque main debe ser único en cada página web, no pueden existir dos etiquetas main. Jerárquicamente su

⁸ Las recomendaciones del estándar HTML5 son eso, recomendaciones. Las páginas funcionarán igual aunque pongas la barra de navegación como un tabla por poner un ejemplo

elemento padre solo puede ser html, body, div o form. Habitualmente se usa como elemento hijo de body.

- **header:** Marca el lugar para colocar encabezados de la página o partes de ella. Aplicado a la página completa podría usarse para contener el logo, título y subtítulo de la página web.
- **nav:** Marca un espacio para una barra de navegación: un menú con enlaces que apuntan a distintas páginas del sitio web. Puede estar en cualquier parte de la página, y pueden haber más de una, tanto en horizontal como en vertical. Solo identifica el elemento que contiene la barra, no la construye.
- **Aside:** Espacio para información, datos o contenidos relacionado con el contenido que la rodea, sin llegar a formar parte del contenido principal de la página. Usado por ejemplo para banners de publicidad, enlaces a otros lugares del sitio, enlaces externos, comentarios. También podría ser un lugar para un índice de contenidos o para un bloque **nav**.
- **footer:** Un espacio para un pie de contenido. Aplicado a una página podría contener, por ejemplo, datos del autor, contacto, copyright y otros.
- **section:** Un espacio para el contenido principal de la página. Puede haber varios espacios **<section>** por supuesto, pero deben estar relacionados con la información relevante de la página, con su objetivo.
- **article:** Una parte de la página autocontenida o independiente, es decir, un contenido de la página que puede ser extraído de la misma sin que pierda significado. Por ejemplo en un blog, un post sería un ejemplo perfecto de **<article>**.

Esta es una estructura general, es un esquema y las etiquetas no son tan rígidas como puedan parecer. Por ejemplo, **<header>** marca un espacio de titular, que puede ser el título de la página pero también puede estar dentro de una **<section>** o de un **<article>** para marcar su título. Incluso podría usarse para contener un elemento nav con un menú de navegación.

De la misma forma **<section>** puede contener uno o más apartados **<article>**, pero también al revés: un **<article>** puede estar organizado en varios elementos **<section>**. Una página de libros puede tener un **<section>** global que contenga un par de elementos **<article>**: ebook y papel. El **<article>** ebook puede contener un **<section>** para novedades y otro para críticas.

Estas son **las etiquetas HTML5 fundamentales** en la *organización de una página web*, aunque existen más con un significado ligado a la estructura del documento, pero las veremos más adelante.

Las direcciones web: URL

Como ya habrás leído las páginas web son un ejemplo de un tipo de documentos denominados **documentos de hipertexto**⁹, es decir, documentos que contienen información directamente visible (como las páginas de un libro) y además formas de acceder a otras fuentes de información, normalmente relacionada con el documento. Estas otras fuentes pueden ser una imagen, un video, un audio, otro documento u otra página web.

⁹ Los enlaces son los que definen el apelativo de hipertexto de las páginas web

Al hablar de **enlaces web** nos referimos a como construir estos accesos para llegar a otras fuentes de datos. **Los enlaces o links son elementos interactivos de la página que al ser activados nos dan acceso a contenido en otro lugar.** Estos elementos suelen destacarse de alguna forma: color, subrayado, una imagen, un botón.

Por esto es necesario algún mecanismo para localizar cualquier recurso en la web. Sin entrar en detalles técnicos podemos decir que cualquier elemento conectado a internet se identifica por un nombre o dirección del estilo

<https://www.espaciolatino.com>. A estas direcciones se les llama **Universal Resource Locator**, o **URL**, es la forma de localizar cualquier página o recurso en internet. En general tiene el siguiente formato:

protocolo:// máquina:puerto /ruta/ fichero?parametros=value

<https://www.example.com/correo/contactar.htm?asunto=comentar>

que tiene diferentes partes:

- **Protocolo:** indica como se transfiere la información, según su contenido y finalidad. Los valores que pueden encontrarse en las redes son
- **http:** Es el protocolo usado para transmitir documentos HTML, es decir, el que habitualmente usamos para ver las páginas en nuestro navegador.
- **https:** Es similar al anterior pero con la particularidad de que la información viaja codificada mediante técnicas de encriptación.
- **ftp:** Es un protocolo para la transmisión de ficheros (**File Transfer Protocol**). Permite intercambiar ficheros con sitios en los que se ejecuten servidores *ftp*. Los navegadores actuales permiten acceder a estos servidores FTP, pero lo más normal es usar unos programas al efecto denominados **clientes de FTP**.
- **mailto:** Este protocolo sirve para acceder a servidores de correo y se usa para enviar o recibir correos electrónicos.
- **news:** Mediante este protocolo accedemos a los denominados grupos de noticias, listas de distribución de mensajes relativos a temas concretos. Habitualmente se accede a estos servidores mediante el cliente de correo.
- **telnet:** Es un terminal de acceso remoto en modo texto. Es un sistema bastante inseguro por lo que no es habitual en servidores donde la seguridad esté bien pensada
- **Máquina:puerto:** Es la dirección del ordenador donde está situado el recurso. Tiene dos partes:
 - Máquina:** servidor propiamente dicho.
 - Puerto:** número del puerto por donde acceder al servidor, no siempre es necesario (las webs suelen usar el 80 en conexiones http y el 443 en las https).
- **Ruta:** Es la dirección del recurso ya dentro del servidor.
- **Fichero:** Es el nombre del recurso al que queremos acceder

- **Parámetros=value:** Aquí pueden ir datos de acceso para ser usados por la página de destino (GET). Estos datos enviados en la url se procesan por algún script o por algún programa de lado servidor (PHP, ASP).

https://www.example.com/correo/contactar.htm?asunto=comentar

El protocolo es *https*, secured Http. Se encuentra en el sitio *www.example.com* y dentro de la carpeta *correo*. La página buscada es *contactar.htm* y usa el parámetro *asunto=comentar*

O contado de otra manera: se busca la página *contactar.htm* que está en la carpeta *correo* del servidor *www.example.com*, utilizando una *conexión segura* y a esa página se le envían el dato *asunto* cuyo valor es "comentar".

Como ves la máquina no es más que el servidor con el que se va a conectar y no siempre es necesario escribir el número de puerto. Los puertos son como caminos de entrada o salida al servidor y son gestionados por un programa. En el caso de la url del ejemplo el puerto usado es el 443 (es el valor por defecto y no es necesario escribirlo). En servidores de correo el puerto de entrada suele ser el 21. Cualquier servidor tiene una gran cantidad de puertos algunos asociados a servicios específicos (como el servidor web, el servidor SSH, telnet, ftp, mail) y otros de uso libre que pueden asociarse a un servicio, por ejemplo, a veces verás sitios web con una dirección del estilo

https://www.example.com:8080

Simplemente significa que el servidor web opera con el puerto 8080 para recibir y enviar datos.

URL absolutas y relativas.

En la página web vas a encontrar elementos que necesitan usar direcciones URL como por ejemplo los hiperenlaces, las imágenes, los audios y videos, los objects y los links del head entre otros. Habitualmente estas URL se asignan a los atributos **src**, **srcset**, **href**, **cite** y **data**. También verás que se usan en algunas definiciones de estilos CSS, como para definir una imagen de fondo.

Ejemplos de uso de URL,s:

```
<head>
<script src="https://www.example.com/helper.js"></script>
<link href="https://www.example.com/estilos.css" rel="stylesheet"
type="text/css">
</head>
<body>
<a href="https://www.example.com/libros.htm">Ir a la página
libros.htm</a><br>

```

Como ves el uso de las URL no tiene ningún secreto. Pero si miras códigos fuentes de diferentes páginas web (algo recomendable) verás asignaciones como estas:

```
<head>
<script src="/helper.js"></script>
<link href="/estilos.css" rel="stylesheet" type="text/css">
</head>
<body>
```



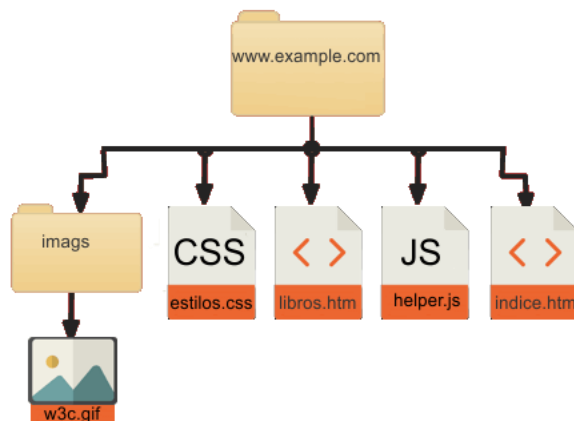
```
<a href="libros.htm">Ir a la página libros.htm</a><br>

```

En este primer ejemplo se han usado **URL Absolutas** en ellas se escribe la dirección del recurso completa, incluyendo el sitio web en el que se encuentra: `www.example.com`. Verás que a veces no se escribe el protocolo `http:` o `https:`, el navegador decide si usar el uno o el otro. Las urls absolutas son imprescindibles si enlazamos sitios externos a nuestra web.

En el segundo ejemplo se han usado **URL Relativas**. En estas el recurso se localiza a partir de la página donde está el enlace o bien desde la raíz del sitio web donde está la página. No pueden usarse para sitios externos a nuestra web,

Para entender lo de las URLs relativas supón que la URL de la página de ejemplo es `https://www.example.com/clase/indices.htm` y que existe una carpeta con esta URL `https://www.example.com/imags/`. Esta imagen es el esquema de la estructura del sitio



La dirección de la hoja de estilos *estilos.css* o del script *helper.js* comienzan con la barra invertida: es como escribir `https://www.example.com/estilos.css`. Ahorramos letras.

En el caso de *libros.htm* la página está en el mismo directorio o carpeta donde está la página *indice.htm*. Y fíjate que para localizar la imagen se usa el sistema de los **puntos**: dos puntos indican directorio o carpeta padre, en este caso `www.example.com`, es como subir un nivel para describir la localización del fichero.

6. Marcos: los iframe

El estándar HTML 5 no acepta los marcos o frames usados en la anterior versión de este lenguaje para organizar varias páginas en una única ventana (una página contenedora con varias páginas hijas), pero mantiene la alternativa introducida por Explorer, los **iframe**¹⁰, aunque no contempla los mismos atributos, que deben sustituirse por reglas de hojas de estilo CSS.

¹⁰ Los *iframe* permiten el uso de una página externa dentro de tu página lo que puede ser útil para compartir un menú por todas las páginas de un sitio

Este elemento permite mostrar una página web dentro de otra. Este marco puede ser colocado en cualquier parte de la página usando adecuadamente los estilos CSS. Es algo parecido a lo que en programación sería un include o import.

La configuración de este elemento es muy simple, lo único imprescindible es el atributo **src** para indicar la página que debe incluirse dentro del **iframe**. Además, por supuesto admite cualquier definición de estilo CSS, por lo que podrás configurar sus dimensiones, bordes, márgenes.... Es usada, por ejemplo, para colocar anuncios, en las páginas de juego, suele usarse también para colocar juegos en la página.

Los atributos usados en este elemento son los siguientes:

Atributo	Uso
height	Altura del marco en pixels
width	Anchura del marco en pixels
name	Asigna un nombre a un marco, de esta manera luego podemos usarlo como destino de un enlace.
sandbox	Conjunto de restricciones para el contenido del marco
src	Indica la URL del documento HTML que se verá en el iframe .
srcdoc	Es código HTML que se mostrará en el <iframe> en lugar de una página externa
seamless	Trata el iframe como parte de la página contenedora, por ejemplo, para aplicar estilos de la página contenedora o permitir el acceso a ésta desde el iframe .

El nuevo atributo **sandbox** es un elemento de seguridad para evitar ataques o puntos débiles ante ataques crossdomains. Puede usarse sin ningún valor, con lo cual se aplican todas las restricciones al contenido del iframe. O se le puede asignar una lista de valores para levantar una o varias restricciones.

Si el contenido es de tu mismo dominio, una página que tu controlas, realmente el riesgo de ataque mediante contenido es mínimo o incluso nulo. Pero si el contenido es una página de otro dominio, y tu no controlas su contenido, el riesgo es máximo, pues en esa página podría existir código que se ejecutaría en tu dominio sin tu control, un ataque crossdomain. Sandbox permite encerrar el **iframe** en una especie de caja fuera de la cual no puede hacer nada. Es un contenido puramente estático.

Los valores que acepta el atributo sandbox son:

Valor	Uso
allow-forms	Admite envío de formularios dentro del iframe

allow-pointer-lock	Habilita el uso de API,s desde el iframe
allow-popups	Habilita Pop-ups
allow-same-origin	Considera el iframe como del mismo origen que la página contenedora. Esto permite que el iframe acceda al DOM de la página contenedora
allow-scripts	Permite ejecución de scripts
allow-top-navigation	Puede cambiar el origen de la página contenedora.

Un ejemplo que puedes probar es:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Un ejemplo de iframe con página externa</title>
</head>
<body>
<iframe src="/tutorhtml5/ejemplos-html5/ejm_iframe_cont.htm"
width="400" height="180" style="border: 2px solid red">
</iframe>
</body>
</html>
```

Que es un ejemplo simple de como usar el iframe.

Como ves el elemento no tiene contenido, todos sus datos están en los atributos. Observa el elemento style para definir un borde igualmente podrías establecer por ejemplo un color de fondo. El estilo lo puedes también asignar mediante el atributo **class**, utilizando un estilo predefinido.

7. Página Ejemplo HTML completa

Como primer ejercicio vamos a construir una página con las directrices del HTML 5 vistas hasta el momento:

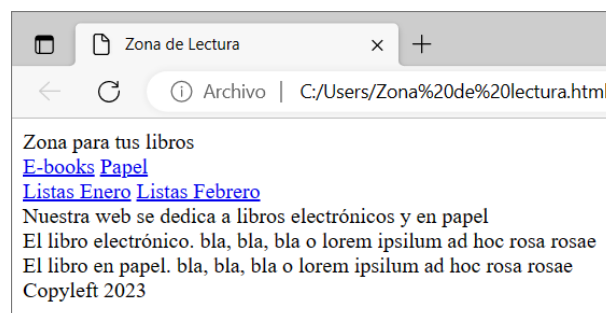
```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Zona de Lectura</title>
</head>
<body>
```

```

<header>Zona para tus libros</header>
<nav>
<a href="ebooks.htm">E-books</a>
<a href="enpapel.htm">Papel</a>
</nav>
<aside>
<a href="listasenero.htm">Listas Enero</a>
<a href="listasfebrero.htm">Listas Febrero</a>
</aside>
<section>
Nuestra web se dedica a libros electrónicos y en papel
<article>
El libro electrónico.
bla, bla, bla o lorem ipsum ad hoc rosa rosae
</article>
<article>
El libro en papel.
bla, bla, bla o lorem ipsum ad hoc rosa rosae
</article>
</section>
<footer>Copyleft 2023</footer>
</body>
</html>

```

Si la abres en un navegador web veremos algo un poco simple, no se parece en nada a una página web al uso.



Lo que ocurre es que estas etiquetas tan **solo definen cada parte en la página**. Si lees el código se ve claramente lo que es cada elemento que has escrito: la cabecera, el contenido de la página, el pie... Pero el navegador no lo muestra bonito, resaltado. Para eso hace falta un poco más de trabajo. Está claro que lo que hemos hecho es definir la **estructura de la página**.

Si quieres que esas partes de la página se muestren de forma diferente debes aplicarles valores a sus atributos o aplicarles un estilo CSS. Por ejemplo, si quieres que footer aparezca en un tamaño de letra de 10px y en negrita podrías añadirle un atributo style:

```

<footer style="font-size: 10px; font-weight: bold">Copyleft
2023</footer>

```

8. Maquetar la página: Líneas y bloques

Una vez decidida y escrita la estructura de la página web toca formatearla para que aparezca en el navegador de una manera clara, atractiva y fácil de utilizar. Toca ir viendo etiquetas HTML básicas, si ya las conoces esto puede servirte de repaso o te lo saltas directamente.

Cuando se habla de elementos de línea (o **inline**) se trata de elementos que aparecen siguiendo el flujo del texto sin cambiar de línea. Mientras que la disposición en bloque¹¹ introduce un cambio de línea. **Por defecto cada elemento pertenece a uno u otro tipo**, pero este comportamiento **puede cambiarse mediante los estilos CSS**, concretamente mediante la propiedad **display**: puede ser entre otras, block, inline-block, flex....

**
 o
: Etiqueta que rompe el flujo del texto con un salto de línea. No define ningún bloque como el párrafo. No forma una unidad. Es una etiqueta individual, no tiene cierre. Este tipo de etiquetas se puede cerrar con /> (propio de XHTML) como final en lugar de un solo >. No se debe usar para introducir líneas en blanco para separar bloques de texto, para eso es mejor usar CSS, concretamente la propiedad **margin.

<p></p>: Etiqueta de bloque para encerrar un párrafo. Todo lo encerrado se muestra en un bloque que rompe el flujo de texto, con una línea en blanco antes y después. No se pueden anidar, es decir, un párrafo no puede contener otros párrafos. Tampoco puede contener elementos de lista (li, ol) ni estar contenido en una lista. El sentido de esta etiqueta es mostrar texto agrupado relacionado entre sí.

<div></div>: Etiqueta que define un bloque. Todo su contenido se comporta como un todo compacto. Solo se usa para dar formato al documento mediante estilos CSS. Antes de aparecer las etiquetas de estructura (header, section, etc.) se usaba este elemento como marca de estructura. Con HTML5 esta etiqueta define un bloque genérico. Pueden anidarse: un bloque **div** puede contener otros bloques **div**.

****: Etiqueta que define un bloque en línea, es decir, que no va acompañado de saltos de línea ni antes ni después, como ocurre con los dos anteriores. Su contenido se comporta como un todo compacto. Se usa para dar formato a parte del documento mediante estilos CSS y/o para identificar un bloque del contenido.

Con estas etiquetas vamos a darle un poco de formato a nuestro ejemplo, página de libros. Vamos a crear párrafos, líneas y algún bloque div sin más pretensiones.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Zona de Lectura</title>
</head>
<body>
<header>Zona para tus libros</header>
<nav>
<a href="ebooks.htm">E-books</a><br>
```

¹¹ Los bloques pueden encerrar texto, otros bloques, imágenes. Los contenedores son bloques padre y los contenidos bloques hijo

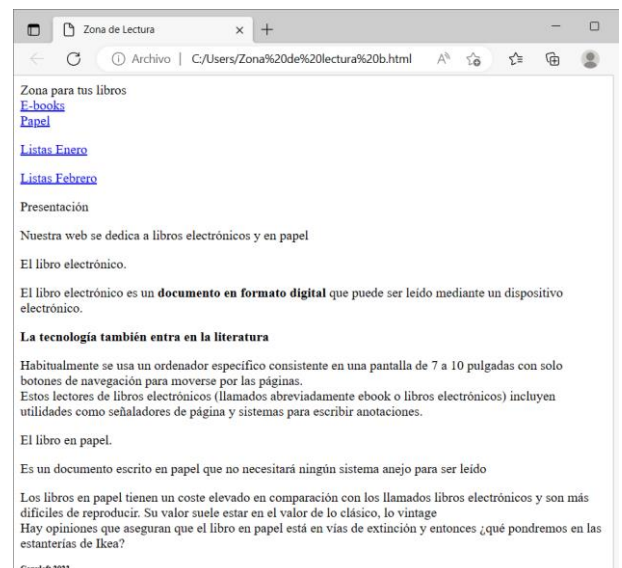
```

<a href="enpapel.htm">Papel</a>
</nav>
<aside>
<p><a href="listasenero.htm">Listas Enero</a></p>
<p><a href="listasfebrero.htm">Listas Febrero</a></p>
</aside>
<section>
<header>Presentación</header>
<p>Nuestra web se dedica a libros electrónicos y en papel</p>
<article>
<header>El libro electrónico. </header>
<p>El libro electrónico es un <span style="font-weight:
bold">documento en formato digital</span>
que puede ser leído mediante un dispositivo electrónico.</p>
<div style="font-weight: bold">La tecnología también entra en la
literatura</div>
<p>Habitualmente se usa un ordenador específico consistente en una
pantalla de 7 a 10 pulgadas con solo botones de navegación para
moverse por las páginas.<br>
Estos lectores de libros electrónicos (llamados abreviadamente ebook
o libros electrónicos) incluyen utilidades como señaladores de
página y sistemas para escribir anotaciones.</p>
</article>
<article>
<header>El libro en papel.</header>
<p>Es un documento escrito en papel que no necesitará ningún sistema
anejo para ser leído</p>
<p>Los libros en papel tienen un coste elevado en comparación con
los llamados libros electrónicos y son más difíciles de reproducir.
Su valor suele estar en el valor de lo clásico, lo vintage<br>
Hay opiniones que aseguran que el libro en papel está en vías de
extinción y entonces ¿qué pondremos en las estanterías de Ikea?</p>
</article>
</section>
</p>
<footer>Copyright 2014</footer>
</body>
</html>

```

Al abrirlo con un navegador veremos algo como esto:

No es una maravilla pero ya has comprobado los efectos de las etiquetas **<p>** y **
** las de bloque **** o **<div>**. Observa como el bloque **<div style="font-weight: bold">La tecnología...** está escrito en línea pero crea un bloque que rompe el flujo del texto creando una nueva línea, a diferencia de span que se mantiene en el sitio. A diferencia del párrafo el bloque div admite bloques hijos. Los div,s son bloques todo terreno.



Con los **estilos CSS** estas características de los elementos de la página web pueden cambiarse y adaptarlo a las necesidades del documento concreto. Incluido por ejemplo la ruptura de línea que caracterizan a los elementos tipo bloque.

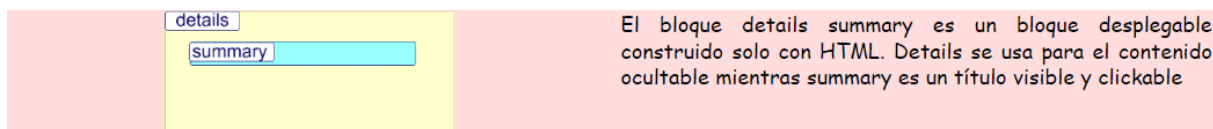
```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Zona de Lectura</title>
</head>
<body>
<div>Defino un boque div normal como este</div>
y ves que rompe la línea de texto.
Sin embargo <div style="display:inline"><strong>Este actúa como un
elemento span</strong></div>
en línea
</body>
</html>
```

(Modificamos el comportamiento del bloque div).

9. Bloque dinámico: details

Un menú desplegable solo con HTML.

Una de las últimas adiciones a HTML es el bloque **details - summary**¹², un bloque con comportamiento dinámico que no necesita que definamos ningún estilo o que programemos un evento con JavaScript. Basta código HTML para plegar o desplegar el contenido de este tipo de bloques



El uso de este bloque es muy simple:

Details: contiene la parte oculta. Cuando está visible adquiere el atributo **open**

Summary: es el título visible y clicable para abrir o cerrar details.

Este ejemplo es el código usado en la muestra de definición de CSS

```
<details>
<summary>Estilos CSS</summary>
El lenguaje usado para controlar el aspecto como se ve de la página
web
</details>
```

Este es el ejemplo mínimo que puedes encontrar. Lo normal es que uses CSS para darle un aspecto algo más acorde con tus gustos. Por ejemplo:

¹² Con el bloque details summary podemos crear menús desplegables sencillos. No hay posibilidades de animaciones si no es empleando CSS y/o JavaScript


```

<style>
.boton{
  border: 1px solid gray;
  width: fit-content;
  padding: 0 10px;
  background: linear-gradient(to right, lightgray, white,
lightgray);
  cursor: pointer;
} .def{
  border: 1px solid lightgray;
  width: 200px;
  box-shadow: 2px sado 2px sado gray
}
</style>
Estilo botón:<br>
<details class="blq">
  <summary class="boton">Estilos CSS</summary>
  <div class="def">
    El lenguaje usado para controlar el aspecto como se ve de la
página web
  </div>
</details>

```

Con lo que has visto ya serás capaz de crear un menú. El bloque **summary** serviría como encabezado o título del menú, no le puede poner un link porque el click del ratón sirve para abrirlo.

El cuerpo restante de **details** contendrá los items del menú como enlaces apuntando a las urls. Esta lista puedes ponerla como líneas normales o como una lista desordenada (ul), o como un bloque flex. Cualquier estructura que mantenga una lista de entradas.

```

<style>
.boton{
  border: 1px solid gray;
  width: fit-content;
  padding: 0 10px;
  background: linear-gradient(to right, lightgray, white,
lightgray);
  cursor: pointer;
}
.def{
  border: 1px solid lightgray;
  width: 200px;
  box-shadow: 2px 2px 2px 2px gray
}
</style>
<body>
Estilo botón:<br>
<details class="blq">
  <summary class="boton">Estilos CSS</summary>
  <div class="def">

```

```
El lenguaje usado para controlar el aspecto como se ve de la
página web    </div>
</details>
```

En este caso cada ítem del menú se ha puesto como un bloque con un efecto hover para resaltar el ítem seleccionado en cada momento y con cursor tipo pointer. El menú se abre o cierra pulsando sobre su encabezado.

Ya solo queda un detalle que seguro que te estás preguntando o te vas a preguntar pronto ¿Y si no quiero ese triángulo en el inicio del bloque summary? Bueno ese marker o marcador viene del tipo de display que usa este bloque: **list-item** que es el que introduce la viñeta. Es fácil de cambiar mediante una regla de estilo CSS: cambiar el display y usar un pseudo selector **:before**, que llevará una propiedad content con la viñeta que queramos ponerle.

También se puede usar un **after** si quieres una viñeta después del título. O ninguno si es que simplemente quieres quitar la viñeta.

```
<style>
.boton{
  display:inline ;
  border: 1px solid gray;
  width: fit-content;
  padding: 0 10px;
  cursor: pointer;
}
.blq .boton:before{
  content:"\2630";
  display:inline-block;
  width:18px;
}
.blq[open] .boton:before{
  content:"\268A";
  display:inline-block;
  width:18px;
}
.links{
  text-decoration:none
}
</style>
Estilo botón:<br>
<details class="blq">
  <summary class="boton">Estilos CSS</summary>
  <ul class="links">
    <li><a href="#">itemt1</a></li>
    <li><a href="#">itemt2</a></li>
  </ul>
</details>
```

Como ves este bloque es totalmente personalizable y tiene bastantes aplicaciones: definiciones, menús, respuestas ocultas, elementos ocultos...

10. Cuadro de diálogo

Los scripts de una página web pueden comunicarse con el usuario para mostrar mensajes mediante el cuadro **alert** y pueden recibir entradas del usuario mediante **prompt** o usando formularios **form**.

HTML presenta un elemento que permite aunar alert con prompt y formularios. Se trata del bloque **dialog**¹³, si bien para aprovecharlo al máximo es necesario utilizar al menos un script.

Interaccionando con los usuarios: HTML dialog

La marca **<dialog></dialog>** crea un bloque para mostrar mensajes en la página web y recoger respuestas del usuario. Las respuestas pueden ser procesadas por javascript.

Este nodo tiene un atributo: **open**, cuando existe el bloque se muestra en la ventana del navegador. Habitualmente contiene algún botón como aceptar, ok, cerrar, abortar o el que quieras ponerle dentro. Al pulsar cualquiera de ellos el cuadro de diálogo se cerrará.

Para utilizar este cuadro necesitamos de JavaScript¹⁴. Es bien fácil:

- El dialogo se abre.
- Muestra su contenido con algún botón.
- El cuadro se cierra al pulsar el botón.

Para que funcione de esta manera se necesita usar eventos que llamen al script para abrir o cerrar la ventana.

Lo mejor para ver su funcionamiento es un ejemplo.

```
<button onclick="abrir()">
Abrir dialogo </button>
<button onclick="cerrar()">
Cerrar dialogo </button>
<dialog id="msg">
¡Hola, esta es una ventana muy simple
</dialog>
<script>
function abrir(){
  let dlg = document.getElementById("msg");
  dlg.show();
}
function cerrar(){
  let dlg = document.getElementById("msg");
  dlg.close();
} </script>
```

¹³ El cuadro o bloque dialog es una alternativa muy flexible a los alert y prompt usados por JavaScript

¹⁴ El cuadro de diálogo se va a usar normalmente con un script. Los eventos que soporta este elemento HTML son **close** y **cancel**

Este ejemplo tan solo abre o cierra la ventana de diálogo.

Para JavaScript este elemento es un objeto que puede mostrarse **show()**, mostrarse como modal **showModal()** y cerrarse **close()**. Y sus propiedades son el atributo **open** para mostrar su estado (visible u oculto) y **returnValue**, valor devuelto según el botón pulsado si se usa con un elemento form.

```
<button onclick="abrir()">
Abrir dialogo </button>
<div id="respuesta"></div>
<dialog id="msg" onclose="resultado()">
¿Cuál es tu color preferido?
<form method="dialog">
<button value="rojo">Rojo</button>
<button value="azul">Azul</button>
</form>
</dialog>
<script>
function abrir(){
    let dlg = document.getElementById("msg");
    dlg.show();
}
function resultado(){
    let dlg = document.getElementById("msg");
    let resp = document.getElementById("respuesta");
    resp.innerHTML = dlg.returnValue; } </script>
```

En este otro ejemplo el script recoge un valor dependiendo del botón pulsado. Como ves se usa un form con el atributo **method="dialog"**. Esto facilita el procesamiento de las ventanas de diálogo.

11. Etiquetas de titulares

Las etiquetas de titulares o cabeceras sirven para **enfatar el texto** que va a actuar como un título de algún elemento de la página: una sección, toda la página, un apartado, un bloque.

Un ejemplo sencillo: un libro de texto. Cada capítulo suele tener un título en letras grandes y muy llamativas, luego puede llevar un subtítulo con letras algo más pequeñas, a continuación, van los apartados del tema cuyo título es menos impactante que el título del capítulo, luego dentro de cada apartado pueden aparecer subapartados con un título algo menos impactante, pero resaltado respecto al texto del capítulo. Pues cada estilo usado para esos títulos es lo que se llaman etiquetas de cabecera o titulares (headings) en HTML.

Existen 6 niveles, de mayor a menor importancia: <h1> hasta <h6>.

Esto sería titular en estilo H1

Esto sería titular en estilo H2

Esto sería titular en estilo H3

Esto sería titular en estilo H4

Esto sería titular en estilo H5

Esto sería titular en estilo H6

Estas etiquetas tienen apertura y cierre de la forma `<h1>texto</h1>`. Pueden modificarse mediante CSS. No conviene abusar de estas etiquetas, se deben usar de forma razonable.¹⁵

Por ejemplo: una página web solo tiene un título, por tanto basta con una sola etiqueta **h1**. Tendrá más utilidad cuando aparezcas en una lista de un buscador. Luego las secciones de la página podrían tener títulos en **h2**, estos ya si pueden repetirse.

Puedes tener varias etiquetas h1 en la página, pero siempre para títulos raíz de algún bloque independiente dentro del documento. Puedes usar h1 para el título de **article**, que es un contenido que no pierde su sentido si se lee independiente de la página.

¿Y en las **section**? En principio también las **section** podrían tener títulos de nivel 1, pero habitualmente son elementos hijos de un elemento principal superior, no por flujo si no por semántica, en esos casos normalmente tiene más sentido utilizar niveles por encima del 1 para sus cabeceras.

Por último, tener en cuenta que se comportan como un elemento de párrafo, es decir que añaden un espacio en blanco por delante y por detrás. Y, como ocurre con los párrafos, no se deben anidar unas dentro de otras.

Modificando el código del ejemplo con estos nuevos estilos, nos debería quedar algo como esto:

```
<body>
<header>
<h1>Zona para tus libros</h1>
</header>
..... <header>
<h2>Presentación</h2>
</header>
.... <h3>El libro electrónico. </h3>
..... <h3>El libro en papel.</h3>
..... <footer>
<h5>Copyright 2014</h5>
</footer>
```

Como ves cada titular tiene un aspecto distinto en función del nivel o importancia dentro de la página. De todas maneras ese formato puede cambiarse con etiquetas de estilo, incluso podrías tener etiquetas h3 de mayor tamaño y más destacadas que las h1.

¹⁵ Es recomendable no excederse en el uso de cabeceras **h1** en cada página. Al menos debes usar una destinada a ponerle un título a esa página. Los artículos pueden tener sus propias **h1**

12. Atributos

Casi todas las etiquetas de HTML suelen llevar un atributo¹⁶ con un valor que define algo relativo al contenido al que se refiere esa etiqueta. Por ejemplo, aquí estamos definiendo un identificador al párrafo:

```
<p id="Documentales">
Los documentales no son películas, son grabaciones que exponen un
hecho real. Pueden usar técnicas propias del cine.</p>
```

Existen atributos específicos que solo se pueden usar con ciertas etiquetas, como **width** o **height** para indicar tamaño en una imagen (**img**), y atributos genéricos que valen para cualquier elemento de la página web, como **class** para aplicar una clase de **estilo CSS** a un elemento de la página.

En este capítulo se muestran los atributos generales aceptados por todos o la mayoría de los navegadores web. Verás que algunos de ellos solo tienen sentido si se usan además programas JavaScript en el diseño de la página.

Atributos genéricos

Estos atributos genéricos pueden aparecer en cualquier etiqueta de la página, valen tanto para un párrafo `<p></p>` como para un `<div></div>` o un bloque en línea ``. Casi todos son booleanos, es decir, que aceptan el valor **true** o **false**, salvo algunos que aceptan texto o números.

Atributo	Descripción
accesskey	Sirve para especificar una tecla que servirá para dar el foco al elemento. El valor es la tecla y se activa junto a la tecla ALT o ALT+SHIFT, depende del navegador. Solo tiene sentido en elementos que puedan tener el foco, como campos de formularios o enlaces. <code>comedia</code>
class	Sirve para indicar el nombre de la clase o clases de una hoja de estilos CSS que se aplica al elemento. Estas clases estarán definidas en el propio documento (HEAD) o en un archivo externo. Se pueden asignar varias clases separando sus nombres con espacios. <code><div class="cabecera"></code> <code>Aparecerá en el formato definido para la</code> <code>clase <i>cabecera</i> en una hoja de estilos CSS </div></code>
contenteditable	Sirve para indicar si un contenido es editable. Para que tenga utilidad se debe utilizar junto a un script, es decir, programando. El usuario puede modificar el elemento, pero esa modificación

¹⁶ Los atributos son como propiedades que se asignan a un elemento del documento html

	<p>no afecta a la página almacenada en el servidor solo a la copia que ve en ese momento en su navegador.</p> <p><code><p contenteditable="true"> Este párrafo puede ser modificado, prueba a escribir aquí, pero al actualizar la página se pierde todo</p></code></p>
data-*	<p>Este atributo se usa para almacenar información en la página. El asterisco es un comodín (puede ser cualquier palabra). Es útil para dar información extra a los robots de búsqueda entre otras cosas. Este atributo merece un capítulo aparte.</p>
dir	<p>Para la dirección del texto. No demasiado útil a menos que escribas páginas con contenido en idiomas que no se lean de izquierda a derecha, como el árabe que se lee de derecha a izquierda. Los valores que admite son:</p> <ul style="list-style-type: none"> - ltr: izquierda a derecha - rtl: derecha a izquierda - auto: en función del idioma de la página
draggable	<p>Marca el elemento al que se aplica como arrastrable o no, es decir, si puede seleccionarse y arrastrarse para soltar en alguna parte de la página. Estas operaciones hacen uso de programación JavaScript.</p>
hidden	<p>Sirve para indicar que el elemento al que se aplica no es relevante y el explorador no lo mostrará. Realmente no es demasiado útil en el entorno web, donde se usan los estilos para hacer esta ocultación (display:none o visibility:(hidden o collapse)).</p>
id	<p>Este atributo permite asignar un identificador al elemento al que se aplica. Un script o programa puede hacer uso de este valor para manipular el elemento o puede usarse una hoja de estilos para definir un estilo para ese elemento. No deben repetirse en una misma página web.</p> <p><code><table id="atributos">... </table></code></p>
lang	<p>Para páginas en diferentes idiomas este valor nos da el idioma en el que está escrito el elemento. Se usan los códigos ISO de abreviaturas de los idiomas como es para español, fr para francés, en para inglés... Útil para los buscadores</p>
spellcheck	<p>Si le damos el valor true el elemento al que se aplica será comprobado ortográficamente. útil en elementos de formularios donde se vaya a escribir texto.</p>
style	<p>Con este valor podemos definir el estilo (el aspecto) que tendrá el elemento como tipo de letra, colores, tipo de listas.... No se aconseja el uso de estilos en línea, es preferible usar archivos externos, pero puede ser útil para un caso concreto y puntual o para ser manejado por scripts. Para entender el uso por completo es necesario conocer el lenguaje de estilos CSS</p>

	<code><div style="height: 100px; width:200px">Cuadro de 100x200 </div></code>
tabindex	Cuando varios elementos pueden tener el foco dentro de una página el valor aquí almacenado da su número de orden cuando se salta de uno a otro con la tecla TAB o ENTER. La utilidad más evidente es un formulario con varios campos para editar. Aquí podemos definir el campo al que se pasa cuando se termina de rellenar cada input.
title	Se puede anotar información sobre el elemento de manera que al pasar el ratón por encima se muestra una pequeña caja con el texto anotado en este atributo. Muy útil para facilitar el uso de las páginas. Coloca el ratón sobre el ejemplo de abajo. <code><p title="Los géneros de lectura">Los géneros de literatura que puedes encontrar en esta web son los de la lista... </p></code>

Algunos de estos atributos¹⁷ los volveremos a ver dentro de los elementos que se van a ir tocando en este tutorial.

13. Etiquetas de formato HTML

Cuando leemos un texto vemos que algunas palabras o fragmentos están escritos en **negrita** o en *cursiva* o en un tamaño o color diferente, sabemos que eso implica que ese texto es especial por algún motivo: por su importancia o su significado o por enfatizarlo o porque es una cita, etc. Y este significado lo conocemos porque sabemos leer, es el aspecto visual, su estilo, lo que nos ayuda a interpretarlas. Le damos un **formato diferente**.¹⁸

En HTML 5 esa diferenciación del texto no es solo visual, va más a nivel de significados: si un robot recorre nuestra página y ve una etiqueta la interpreta. Por ejemplo, si ve un texto entre las etiquetas **** interpreta que ese fragmento es de suma **importancia** para el contenido global de la página.

En HTML5 las **etiquetas de formato son también etiquetas de nivel semántico**. A veces el aspecto que dan al texto no es tan relevante como deseamos, para eso tenemos las hojas de estilo. Mediante CSS *podemos redefinir el formato de cualquier etiqueta* de HTML. Así que si vas a poner un texto en negrita o cursiva o subrayado debes pensar porque y para qué lo haces, que quieres transmitir con eso.

En esta tabla de formatos tienes un resumen de las etiquetas para dar cambiar el aspecto del texto en HTML, se usan en el código de la página y su significado va en función del contexto:

¹⁷ De los atributos genéricos aquí citados quizás los más utilizados sean *class*, *id* y *style*. Son útiles para aplicar reglas de estilo los elementos de la página y para identificarlos y modificarlos o consultarlos desde un script.

¹⁸ Aunque dos etiquetas apliquen un estilo predefinido similar, para HTML 5 no son etiquetas iguales, es el caso de ** y **: visualmente iguales, significados diferentes

Etiqueta	Formato	Ejemplo
	Texto en negrita, solo destaca una parte del texto	texto en negrita
	Da énfasis al texto, es algo importante , que se quiere recalcar en el texto.	texto enfatizado
<i>	Pone el texto en cursiva, se suele usar para palabras textuales.	Saludó con un <i>"Hola"</i>
<small>	Pone el texto algo más pequeño, usado en notas a pie de página, lo que conocemos como letra pequeña en contratos.	Texto <small>small, pequeño</small>
	Texto en negrita, indica texto muy importante	Aviso importante
<sub>	Para escribir subíndices	El término a₂: El término a ₂
<sup>	Para escribir superíndices	E=m·c²: E=m·c ²
<ins>	Indica texto insertado, usado para resaltar modificaciones realizadas en el documento.	Texto <ins>cambiado</ins>
	Indica texto borrado, tachado, usado para resaltar modificaciones del documento	Eliminar esta parte
<mark>	Para marcar un texto, pone un fondo amarillo y texto negro. Marca un texto de especial relevancia	<mark>Aviso</mark> No tocar esta parte

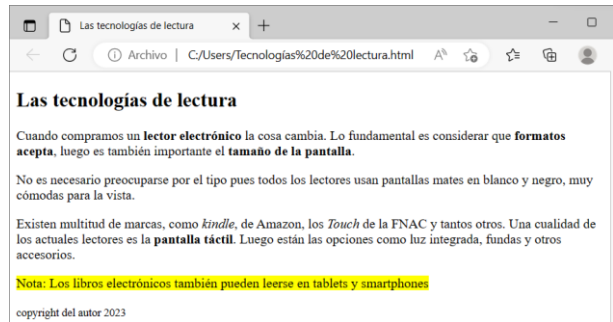
Todas estas etiquetas se usan con apertura y cierre (y) y dan al texto un aspecto predefinido: negrita, cursiva, tamaños, color... pero todas pueden modificarse mediante reglas de estilo CSS, bien sea mediante los atributos **class** o **style**, o redefiniendo directamente la etiqueta (por supuesto con las reglas CSS).

Fíjate que, a diferencia de HTML4, estas etiquetas de formato realmente son importantes por su significado **semántico**, es decir, además del aspecto de un fragmento de texto, dice algo de su función en el documento (por ejemplo, importancia o resultado de cambios). Y esto último es **importante para los buscadores**.

Por ejemplo, quizás te hayas preguntado **¿en qué se diferencian y ?** Ambos ponen el texto en negrita. Pero no son etiquetas exactamente iguales, un robot que explore la página verá diferencias en esos textos. En este caso el

elemento **** solo resalta el texto al que se aplica para fijar la atención sobre el mismo, mientras que al usar **** lo que se pretende es remarcar la importancia de un texto o palabra. No son sinónimos, aunque el efecto visual sea el mismo.

```
<!doctype html>
<html lang="es-ES">
<head>
<meta charset="utf-8">
<title>
mi primera página</title>
</head>
<body>
<article>
<h1>Las tecnologías de lectura</h1>
<p>Cuando compramos un <b>lector electrónico</b>
la cosa cambia. Lo fundamental es considerar que <strong>formatos
acepta</strong>, luego es también importante el <strong>tamaño de la
pantalla</strong>. <p>No es necesario preocuparse por el tipo pues todos los lectores usan pantallas mates en blanco y negro, muy
cómodas para la vista.</p>
<p>Existen multitud de marcas, como <em>kindle</em>, de Amazon, los
<em>Touch</em>
de la FNAC y tantos otros. Una cualidad de los actuales lectores es
la <strong>pantalla táctil</strong>. Luego están las opciones como
luz integrada, fundas y otros accesorios.</p>
<mark>Nota: Los libros electrónicos también pueden leerse en tablets
y smartphones</mark>
</article>
<p><small>copyright del autor 2019</small></p>
</body>
</html>
```



Cualquiera de las etiquetas puede modificarse con tan solo unas líneas de código CSS. Pruébalo añadiendo este código a la página anterior:

```
<!doctype html>
<html lang="es-ES">
<head>
<meta charset="utf-8">
<title>
mi primera página</title>
<style>
em {
font-weight: bold;
font-style:italic;
text-decoration:underline;
}
mark{
background-color: lightgrey;
}
</style>
```

```
</head>
<body>
Ahora la etiqueta em <em>está en negrita, cursiva y subrayado</em>.
</body>
</html>
```

Ahora puedes comparar ambas y verás que sin cambiar nada en la sección body los elementos se ven diferentes. Si pones esto en la página verás que ahora la etiqueta **** no solo escribe en cursiva el texto marcado con ****, sino también en negrita y con el texto subrayado. Prueba a usar las distintas etiquetas para comprobar sus efectos. Y si ya has mirado algo de CSS puedes también redefinirlas con reglas de estilo para comprobar los efectos de cada una.

Respecto a los estilos, si estas reglas las reescribes **en un archivo de estilo independiente**, al modificarlas se verán afectados todas las páginas que usen ese archivo de estilos; algo que te permite cambiar el aspecto de todo tu sitio sin necesidad de ir modificando página por página.

14. Etiquetas para citas textuales

Existen etiquetas que sirven para indicar que un fragmento de texto es una cita procedente de otras fuentes o está relacionado con otros documentos.¹⁹ Es como cuando leemos una noticia vemos un texto que viene copiado literalmente o representa una cita literal de lo dicho por una persona, solemos ver esto entrecomillado o con cursiva o ambas cosas a la vez. Esto se indica en HTML5 mediante etiquetas (¡cómo no!) que visualmente suelen modificar el texto afectado poniéndolo en cursiva. Como siempre estas etiquetas pueden redefinirse para que muestre el texto con el formato que queramos.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<article>
<p>Hoy en día cada vez usamos más los llamados ebooks, y ¿qué es un
<abbr title="Electronic Book">EBook</abbr>? </p>
<blockquote>Los E-books son la versión digital de los tradicionales
libros impresos, los cuales pueden ser visualizados a través de un
programa o dispositivo diseñado para aquello.</blockquote>
<p>Así es como lo define la web <cite>ConceptoDefinicion</cite>.</p>
Aunque también puede referirse al <q>dispositivo móvil que está
diseñado sólo para visualizar libros digitales</q>
</article>
<footer>Autor: <address>Crea Tu Web</address></footer>
</body>
</html>
```

¹⁹ Las etiquetas de texto tienen más importancia por el significado que aportan que por el estilo que definen

Analiza el código y observa las etiquetas usadas. En común tienen que dan información sobre el origen el texto del elemento.

Etiquetas	Descripción	Ejemplo
<abbr>	Para definir una abreviatura. Utiliza el atributo title con un texto como valor, que se mostrará al pasar el ratón sobre la abreviatura o acrónimo que tengamos marcado.	Pasa el ratón por la abreviatura CSS.
<address>	Marca información de contacto como el autor o propietarios de un documento, por eso suele usarse en una la sección article y en footer . El estilo por defecto es cursiva y es un elemento tipo bloque: lleva aparejado un salto de línea antes y después.	Autor <i>Crea tu web.</i>
<blockquote>	Marca un bloque de texto cuyo contenido está extraído desde otro lugar. Es un bloque, con una línea en blanco antes y después. Lleva una indentación, o sea, un margen izquierdo mayor que el resto del documento. Puede usar el atributo cite para indicar el origen del contenido, aunque no será visible.	¿Qué es un ebook? Un libro electrónico, libro digital o ciberlibro, conocido en inglés como e-book o eBook, es la publicación electrónica o digital de un libro.
<cite>	Esta etiqueta indica que el texto es un título de un documento o una obra. Habitualmente aparece en cursiva.	Mi libro favorito es <i>Los asquerosos</i> , una interesante crítica social.
<q>	Se usa para marcar un texto que es una cita textual de otro. Es un elemento en línea y para destacarse aparece encerrado entrecomillas. También puede señalarse la fuente del texto mediante el atributo cite	Ya sabemos que La principal actividad del W3C es desarrollar protocolos y directrices que aseguren el crecimiento de la Web a largo plazo

Como es habitual estas etiquetas dan al texto un formato o estilo por defecto que puede ser modificado mediante la correspondiente definición de estilo con las reglas CSS. Lo importante para HTML 5 sigue siendo el significado que dan al texto.

No obstante, es habitual ver usos fuera de lo recomendado, por ejemplo **blockquote**, que se suele usar simplemente para indentar el texto. Es cómodo, pero para eso podemos usar un bloque **div** con el margen que queramos.

La etiqueta **blockquote**, viene muy bien para indentar un bloque de texto **pero no es su finalidad**. Esta etiqueta se usa para indicar que un texto está extraído de otro lugar. Algo parecido a la etiqueta **q**.

Solo aclarar que aunque aparentemente estas etiquetas no aportan gran cosa al aspecto del documento, no debemos olvidar que nuestra página debe aparecer en buscadores y los robots de estos buscadores analizan las páginas. Estos **robots utilizan las etiquetas para interpretar lo que están escaneando**. Así que mejor usar las que ya existen que inventar otras que los robots no conocen, y si se desea modificar el formato por defecto siempre podemos hacerlo mediante el uso de estilos CSS.

Y ahora vamos a modificar para aplicar estilos que sustituyan a los predefinidos. Necesitamos usar la etiqueta `<style>` para redefinir estos elementos:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<style>
abbr{text-decoration: underline double} blockquote {font-
style:italic; width: 400px} cite{color:blue} q{font-style:italic}
</style>
</head>
<body>
<article>
<p>Hoy en día cada vez usamos más los llamados ebooks, y ¿qué es un
<abbr title="Electronic Book">EBook</abbr>? </p>
<blockquote>Los E-books son la versión digital de los tradicionales
libros impresos, los cuales pueden ser visualizados a través de un
programa o dispositivo diseñado para aquello.</blockquote>
<p>Así es como lo define la web <cite>ConceptoDefinicion</cite>.</p>
Aunque también puede referirse al <q>dispositivo móvil que está
diseñado sólo para visualizar libros digitales</q>
</article>
<footer>Autor: <address>Crea Tu Web</address></footer>
</body>
</html>
```

Aunque no hayamos visto aún CSS, no te será difícil seguir el código fuente de la página modificada: hemos puesto un subrayado doble a la etiqueta de abreviatura, el `blockquote` está ahora en cursiva y además le definimos una anchura para que no ocupe todo el ancho de la página, el elemento `cite` ahora es azul y el entrecomillado está en itálica y eliminamos el salto de línea de `address`.

Los elementos definidos con esta etiqueta mantienen su significado semántico, pero el aspecto que muestran lo hemos personalizado con los estilos CSS.

15. Etiquetas tipográficas

Aún existen otras etiquetas que aquí agrupamos como **etiquetas tipográficas**²⁰ (`code`, `pre`, `samp`...) cuyo efecto visual es mínimo, en general aparecen como cursiva

²⁰ Estas etiquetas definen como se escribe el texto en el flujo de la página

o con alguna fuente de texto diferente, aunque por supuesto usando estilos CSS este aspecto puede ser alterado a nuestro gusto.

Como las restantes etiquetas de texto su valor reside en un significado propio, que robots analizadores de las páginas van a interpretar para entender mejor el significado de un fragmento del contenido. En este grupo el aspecto visual también dice mucho del contenido de estos elementos.

```
<!doctype html>
<html><head>
<meta charset="utf-8">
<title>Documento sin título</title>
</head>
<body>
Para comprar un ebook en nuestra tienda instala nuestro programa<br>
Ve a la carpeta <var>c:\ebooks</var>
y escribe esto:<br>
<kbd>install.exe</kbd><br>
A continuación verás una ventana que dice<br>
<samp>La instalación fue correcta</samp><br>
Y recuerda<br>
<pre>
#####  #####  #####  # # # # # # # # # # # #####  #####  # # # # # # # # # #
# # # # # # #####  #####  #####  #####  # # </pre>
</body>
</html>
```

Observa este ejemplo de uso de las etiquetas tipográficas. Salvo `var` las restantes dan el aspecto de texto de consola por su fuente monospace. Y fíjate en el uso del elemento **pre**: mantiene el texto tal y como se escribe en el código fuente de la página: respeta los saltos de línea y espacios, si lo quitas verás como las marcas del dibujo se acumulan en una línea, y no se ve la figura.

En general estos tipos son muy útiles para escribir código en la página web. Y como siempre, no solo por el aspecto visual sino por el significado semántico de las etiquetas.

Etiqueta	Descripción	Ejemplo
<code>	El texto que encierra es código de programa. El ancho de carácter es fijo y no implica salto de línea antes y después.	Para imprimir usa la orden <code>print</code>
<kbd>	El texto que encierra se interpreta como entrada desde teclado. El ancho de carácter es fijo	Escribir el texto Los libros entretienen
<samp>	El texto que encierra se interpreta como un texto de salida, resultado de algo. El ancho de carácter es fijo.	El resultado de la orden dir es 15/05/2018 20:15 854 list.sql
<var>	En un código indica un identificador de variable o constante física o elemento de programa, útil en un contexto matemático.	$E = m c^2$

<pre>	Se usa para texto preformateado, respeta los espacios en blanco y los saltos de línea (sin necesidad de o), es decir, el texto aparece tal y como se escribe en el código fuente la página. Se usa un tipo de letra con ancho de carácter es fijo (monospace).	En este texto se respetan los espacios y tabulaciones
-------	--	---

16. Enlaces en la página web

Como ya habrás leído las páginas web es un ejemplo de **documento de hipertexto**, es decir, documentos que contienen información directamente visible (como las páginas de un libro) y además formas de acceder a otras fuentes de información, normalmente relacionada con el documento.

Estas otras fuentes pueden ser una imagen, un video, un audio, otro documento u otra página web. Esta propiedad es la que enriquece y da utilidad a las páginas web.²¹

Para construir estos accesos a otras fuentes de datos se usan los **hiperenlaces** (hyperlinks): **elementos interactivos de la página que al ser activados nos muestran contenido de otro lugar**. Habitualmente se abrevia y se les llama links o enlaces web. Estos elementos de la web suelen destacarse de alguna forma: color, subrayado, una imagen, un botón para llamar la atención del usuario.

La etiqueta **<a href >** nos sirve para definir estos enlaces web. Lo que esté encerrado en esta etiqueta será un elemento interactivo que puede ser activado por el usuario, normalmente con un click del ratón o con el dedo en el caso de pantallas táctiles.

Si escribiste la página del primer ejemplo ya has usado este elemento:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Ejercicio1: Zona de Lectura</title>
</head>
<body>
<header>Zona para tus libros</header>
<nav>
<a href="ebooks.htm">E-books</a>
<a href="enpapel.htm">Papel</a>
</nav>
</body>
</html>
```

²¹ Los enlaces son elementos inevitables en el contenido de cualquier página web

Como ves el atributo **href** tiene como valor la URL del recurso de destino. La url puede ser tanto absoluta como relativa. Y no solo apuntar a una página externa, puede apuntar a un elemento concreto de una página.

El contenido del elemento puede ser texto o una imagen. Si se usa un texto este aparece por defecto en azul y subrayados: E-books, aunque mediante los estilos CSS este aspecto se puede modificar.

Este atributo es el fundamental para definir el elemento hyperenlace. Existen otros atributos

Atributos más habituales del elemento `<a href>`

Atributo	Uso	Valores
href	Indica el destino del enlace	Url del destino
target	Donde abrir el destino del enlace	_blank, _self, _parent, _top, frame
download	Es un enlace para descargar un recurso	URL del recurso
rel	Información sobre la relación entre el destino del enlace y la página actual. Pueden darse varios valores separados por un espacio	alternate, author, bookmark, external, help, license, next, nofollow, noreferrer, noopener, prev, search, tag
ping	Envía una petición POST a una o varias direcciones web	Lista de Url,s.
type	Nombre del tipo de recurso apuntado.	Identificador MIME

El atributo **target** permite definir donde se va a abrir la página llamada con href. Se puede abrir en la misma ventana o en otra nueva o en un marco. Los valores lo dicen claro:

- **_blank:** la nueva página se abre en una nueva ventana o pestaña del navegador. La página original sigue abierta.
- **_self:** es la opción por defecto y no es necesario escribirla, la página se abre en la ventana actual, sustituye a la página origen.
- **_parent:** si el enlace está en un marco la página se abre en la ventana padre, la que contiene al marco.
- **_top:** si existen marcos la página llamada sustituye a todos los marcos, que se cierran y solo queda la página nueva.
- **Frame:** Es el nombre de un marco (iframe) en el que se abrirá la nueva página. Es el atributo name del iframe. El mecanismo de los marcos es útil para construir índices compartidos por todo un sitio. Una página con un índice o barra de navegación abriría las páginas en un iframe. Si se quiere modificar el menú solo hay que hacerlo en una página, la página padre o contenedora del iframe donde se cargan las páginas señaladas en el menú.

```

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
<base href="/tutorhtml5/ejemplos-html5/">
</head>
<body>
<p>Pulsa los enlaces para rellenar el iframe</p>
<a href= "ejercicio01.htm" target="ejemplos">Ejemplo 1</a><br>
<a href= "ejercicio02.htm" target="ejemplos">Ejemplo 2</a><br>
<iframe name="ejemplos" style="with: 600px; height: 600px; border:
1px solid blue"></iframe>
<p>Página contenedora</p>
</body>
</html>

```

Las marcas de enlaces internos

El atributo **href** de los hiperenlaces pueden apuntar como destino a un punto concreto de la página de destino. Podría hacer que el hiperenlace apunte a un apartado de la página, o al pie de página. Es decir, se puede definir como destino una referencia dentro de una página.

El destino dentro de una página se marca mediante un elemento **<a>** sin **href**, solo necesita un identificador (**atributo id**). Ahora tendríamos un ancla u objetivo para un hiperenlace. El valor del atributo **id** la identifica y su url sería la url de la página seguida del símbolo **#** y el nombre de la referencia. Por ejemplo, si pulsaras un enlace *ir a la cabecera* la página hace scroll y sube sola al inicio.

Pero además podemos hacer que se abra la nueva página y se desplace hasta el ancla que le indiquemos, basta con añadir a la url el símbolo **#** seguido del nombre del ancla

```

<a href="#cabecera">Ir a la cabecera</a>
<a href="loslibros.htm#papel">Ir a libros</a>

```

En este ejemplo el primer enlace lleva a una referencia de la misma página con **id="cabecera"** haciendo el scroll necesario para que esa referencia quede en la parte superior de la ventana del navegador.

Mientras que el segundo enlace abriría la página **loslibros.htm** y haría el scroll vertical necesario para mostrar la página a partir del ancla con **id="papel"**.

17. Listas HTML

Las listas²² son una forma de presentar texto ordenado y, usando estilos CSS, se pueden convertir en un sencillo sistema para crear menús de lo más sofisticado. En

²² Las listas son un excelente elemento para crear menús de navegación. Aunque suelen verse como listas verticales, nada impide que se muestren en horizontal dándole el valor **inline-block** o **table-cell** al estilo **display** de CSS

la página de ejemplo que te proponía al hablar de los formatos de cabecera tenías varias líneas con enlaces construidas con saltos de línea
. Pues te propongo que la modifiques con este código:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Zona de Lectura</title>
</head>
<body>
<h1><header>Zona para tus libros</header></h1>
<nav>
<ul>
  <li><a href="/ebooks.htm">E-books</a></li>
  <li><a href="/enpapel.htm">Papel</a></li>
</ul>
</nav>
<aside>
<p>Litas por meses</p>
<ol>
  <li><a href="../listasenero.htm">Listas Enero</a></li>
  <li><a href="../listasfebrero.htm">Listas Febrero</a></li>
</ol>
</aside>
<section>
<header><h2>Presentación</h2></header>
<p>Nuestra web se dedica a libros electrónicos y en papel</p>
<dl>
<dt><b>El libro electrónico</b></dt>

<dd>El libro electrónico es un <b>documento digital</b> que permite
leer un texto mediante un ordenador.
Habitualmente se usa un ordenador específico consistente en una
pantalla de 7 a 10 pulgadas con solo botones de navegación para
moverse por las páginas.<br>
Estos lectores de libros electrónicos (llamados abreviadamente ebook
o libros electrónicos) incluyen utilidades como señaladores de
página y sistemas para escribir anotaciones.</dd>
<dt><b>El libro en papel</b></dt>
<dd>Es un documento escrito en papel que no necesita ningún sistema
anejo para ser leído<br>
Los libros en papel tienen un coste elevado en comparación con los
llamados libros electrónicos y son más difíciles de reproducir. Su
valor suele estar en el valor de lo clásico, lo vintage</dd>
</dl>
</body>
</html>
```

En este ejemplo tienes los tres tipos de listas que puedes crear con HTML

- ****: estas etiquetas marcan las listas desordenadas. Los elementos de estas listas se marcan con las etiquetas ****. Estas listas tienen un icono

delante del texto del ítem de lista. Por defecto es un punto negro, pero es personalizable vía CSS. En estas listas el orden en que aparezcan los elementos no altera el significado o el resultado de la información

- ****: estas etiquetas marcan las listas ordenadas. Los elementos de estas listas se marcan con las etiquetas ****. Cada ítem de la lista es precedido por un número o una letra. Es personalizable vía CSS. En estas listas el orden en que se citan los elementos es importante para la información.
- **<dl></dl>**: estas etiquetas marcan las listas de definiciones. desordenadas, las listas de descripciones. Los elementos de estas listas tienen dos partes **<dt></dt>** un encabezado y un texto **<dd></dd>**. Estos párrafos que estás leyendo (desde hasta aquí) está formateado como lista de definiciones. Es un tipo de lista aconsejable si queremos destacar un elemento y un texto relacionado con él.

Listas anidadas

Las listas pueden anidarse, es decir, puedes usar una lista como elementos de otras listas: Una lista ordenada dentro de una sin ordenar o viceversa o listas de descripciones insertadas en listas ordenadas. Lo que quieras, lo que no puede meter dentro de una lista son párrafos. Si necesitas saltos de línea debes usar la etiqueta
.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Listas de ventas</title>
</head>
<body>
Los más vendidos
<ul>
  <li><b>Enero</b>
    <ol>
      <li>Los asquerosos</li>
      <li>Un cuento perfecto</li>
      <li>Loba negra</li>
    </ol>
  </li>
  <li><b>Febrero</b>
    <ol>
      <li>Sapiens</li>
      <li>Reina roja</li>
      <li>Un mundo Feliz</li>
    </ol>
  </li>
</ul>
</body>
</html>
```

Tienes una lista no ordenada para los meses y dentro de cada mes hemos insertado una lista ordenada. Puedes aplicar este esquema a cualquier tipo de listas. He dejado el ejemplo así de esquemático para no cargar la lectura, pero los títulos podrían ser perfectamente enlaces

No hemos usado aún estilos, solo etiquetas para resaltar el nombre el mes y darle un aspecto más claro.

Tipos de listas

Las listas tienen diferentes objetivos dentro de la página web. Son la forma más adecuada de presentar información de forma esquemática y son muy útiles para construir barras de navegación.

Sea cual sea el uso, es necesario elegir bien el tipo de lista a emplear.²³

Listas desordenadas

La etiqueta **** significa unordered list, literalmente lista desordenada²⁴, y se corresponde con las viñetas en los procesadores de texto: son listas donde cada elemento está precedido de un símbolo, como un punto o un disco. EL formato es el que sigue:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Miembros del claustro</title>
</head>
<body>
<h2>El equipo</h1>
<ul>
<li>Juan Pérez García</li>
<li>María López</li>
<li>Luis Pérez</li>
<li>Ana Díaz</li>
</ul>
<h2>Las Aulas</h2>
<ul type="circle">
<li>Aula 101</li>
<li>Aula 201</li>
<li>Aula 202</li>
<li>Aula 100</li>
</ul>
<h2>Días</h2>
<ul type="square">
<li>Lunes</li>
<li>Miércoles</li>
<li>Viernes</li>
</ul> ...
```

²³ Hay diferentes tipos de lista aplicables con diferentes fines: menús, lista de datos, esquemas...

²⁴ Las listas no ordenadas son las listas más simples de HTML: simplemente muestran sus ítems uno tras otro con una viñeta a su izquierda

Prueba el ejemplo y verás las diferentes formas de viñeta o bullet, y si no se quieren usar las viñetas se puede poner la propiedad `list-style-type` a `none`.

La viñeta puede sustituirse por un icono, pero para ello se necesita CSS con el pseudo selector `before`.

También se puede usar la propiedad de estilo `list-style-image`, a la que se le da el valor de la url de nuestra imagen `url('imagen.gif')`.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Formas de pago </title>
<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.0.11/css/all.css" >
<style>
.lista{
list-style:none}
.lista li::before{
content:'\f35a';
font-family: "Font Awesome 5 Free";
margin-right: 4px;
}</style>
</head>
<body>
<h2>Formas de pago</h2>
<ul class="lista">
<li>Pay Pal</li>
<li>Visa</li>
<li>Trasferencia</li>
</ul>
```

Es simple: primero se le quita la viñeta por defecto y luego se añade otra mediante el selector **before**. Puedes usar un icono como en el ejemplo o un símbolo como +, \$, # o incluso en `content` podrías poner una imagen `url('imagen.gif')`

La etiqueta **** admite estos parámetros:

Parámetro	Significado	Resultado
compact	Indica al navegador que debe representar la lista de la manera más compacta posible.	<ul style="list-style-type: none">• Primer elemento• Segundo elemento
type="disc", "circle", "square", "none"	Indica al navegador el dibujo que precederá a cada elemento de la lista. Para mayor flexibilidad se admite también como parámetro de .	<ul style="list-style-type: none">• Tipo disc○ Tipo circle▪ Tipo square

Pero actualmente se tiende a usar más las reglas CSS para definir el formato de las listas, en este caso en lugar del atributo `type`, se usaría la propiedad **list-style-type** dentro de una definición de estilo CSS o en línea con el atributo **style**: `style="list-style-type: circle"`

Las listas ordenadas

Las listas también pueden presentarse con sus elementos ordenados²⁵. HTML no ordena solo añade una viñeta alfanumérica que va cambiando para cada ítem de la lista, como 1, 2, 3 o a, b, c.

El tipo de ordenación o de índice se controla con el argumento `type` (1, a, A, i, I) o por las reglas de estilo CSS con la propiedad **list-style** (decimal, upper-alpha, lower-alpha, upper-roman, lower-alpha, y otros)

Las listas ordenadas admiten diferentes formas de numerar sus elementos, orden inverso y modificar los números de los elementos que la forman. Son útiles para situaciones donde el orden de los elementos sea importante, por ejemplo: pasos en una receta de cocina, capítulos en un curso de aprendizaje, orden de importancia de los elementos, etc.

- **type**: Este atributo permite cambiar la forma de numerar los elementos de las listas ordenadas. El valor puede ser
 - 1: para números árabes, es el valor por defecto.
 - I, i: para números romanos en mayúsculas o en minúsculas (I, II, III)
 - A, a: usar letras mayúsculas o minúsculas.
- **start**: Con este atributo puedes especificar el número de orden por donde comenzar a contar los elementos. Es un valor numérico, de manera que si el tipo de índice es alfabético (`type="a"`) y se pone **start="2"**, los elementos empiezan a numerarse por la letra **b**.
- **reversed**: Es una marca sin atributo para ordenar los índices en forma descendente. La lista no cambia de orden lo que cambia es el orden de los números o las letras que se usen para numerar cada elemento.

Listas de definiciones

Este tipo de lista²⁶ utiliza se utiliza para crear listas con dos niveles para sus ítems: un encabezado y un bloque. Habría una jerarquía de niveles entre ambos niveles. Los niveles se señalan mediante indentaciones: el bloque aparece indentado respecto a su encabezado.

Las etiquetas **<dl>/dl>** define la lista y en lugar de bloques **** se utilizan **<dt>** para el encabezado y **<dd>** para el bloque dependiente. Es como si tuviéramos una lista de conceptos y sus definiciones (el texto indentado). Para entenderlo lo mejor es verlas en marcha:

```
<dl>  
<dt>HTML</dt>
```

²⁵ Una lista ordenada se caracteriza por presentar delante de cada elemento un carácter numérico o alfabético para indicar un orden

²⁶ Las listas de definición permiten mostrar encabezadas y texto relacionado. Los elementos **dd** son bloques pero se pueden redefinir como bloques inline, eliminando el salto de línea entre encabezados y texto


```
<dd>Lenguaje para construir documentos web</dd>
<dt>CSS</dt>
<dd>Reglas para definir la apariencia de una página web</dd>
<dd>Se usan en style o en archivos independientes</dd>
</dl>
```

Con los estilos CSS pueden cambiarse la apariencia de encabezados y definiciones, por ejemplo, colocando los primeros en **negrita** y el resto en *cursiva*.

La etiqueta **<dl>** sólo admite como parámetro el ya conocido **compact**, que tiene el mismo comportamiento que con los otros dos tipos de lista anteriores. Un elemento **<dt>** puede contener varios elementos **<dd>**.

```
<head>
<meta charset="utf-8">
<style>
dl {
display:grid;
grid-template-columns: 20px auto;
}
</style>
</head>
<body>
<dl>
<dt>HTML</dt>
<dd>Lenguaje para construir documentos web</dd>
<dt>CSS</dt>
<dd>Reglas para definir la apariencia de una página web</dd>
</dl>
</body>
```

Este ejemplo se ve como romper la estructura de la lista de definición para que encabezado y texto aparezcan en una misma línea. Más fácil que dar estilos basados en float o displays inline.

18. Las tablas en HTML 5

¿Qué es una tabla? Una serie de datos ordenados en filas (horizontal) y columnas (vertical). La intersección de filas y columnas forman las celdas.²⁷

Las celdas de las tablas son ideales para organizar el texto crear apartados y maquetar la página. Son más cómodas que usar los elementos div. Pero las tablas no se inventaron para eso sino para mostrar datos. HTML5 recomienda que solo se usen para eso, mientras que para maquetar y organizar ha potenciado enormemente el uso de los bloques **<div>**.

²⁷ Las especificaciones de HTML 5 inciden en el aspecto semántico de los elementos web y esto hace que las tablas se recomienden solo para datos tabulares: una catalogación, una lista de precios, una lista de valores ... Siempre que se pueda es recomendable seguir estas recomendaciones

El uso excesivo de tablas puede enlentecer la página web. El explorador debe esperar a tener toda la tabla para distribuir el contenido, si bien las velocidades que se manejan hoy día en internet hacen que este retraso sea pequeño, que sepas que existe.

Caption: La Tabla

columna 1	columna 2	columna 3
celda col 1 fila 1	celda col 2 fila 1	celda col 3 fila 1
celda col 1 fila 2	celda col 2 fila 2	celda col 3 fila 2
celda col 1 fila 3	celda col 2 fila 3	celda col 3 fila 3
celda col 1 fila 4	celda col 1 fila 4	celda col 1 fila 4
celda col 1 foot	celda col 2 foot	celda col 3 foot

En esta tabla tienes los elementos básicos de una tabla, no todos son obligatorios. Existen varias zonas o secciones dentro de la tabla.

- **La tabla:** La tabla se construye con las etiquetas `<table></table>`. Es el contenedor padre de las restantes secciones
- **El caption:** Se forma con las etiquetas `<caption></caption>` y es donde se puede colocar el título de la tabla.
- **Cabecera:** Se crea con las etiquetas `<thead></thead>`, y las celdas que contienen se consideran cabeceras, títulos de grupos de celdas, habitualmente columnas. Siempre va a aparecer por encima del cuerpo. Contienen las celdas marcadas como `<th></th>` con el texto que se quiera colocar como cabeceras. El contenido va a aparecer destacado, por defecto en negrita.
Tiene un atributo **scope** cuyo valor indica a que grupo de celdas se aplica la cabecera, por defecto a la columna en que aparece (`scope="col"`), pero puede también aplicarse a su fila (`scope="row"`) a su sección (`scope="rowgroup"`) o a su grupo de columnas (`scope="colgroup"`).
- **Cuerpo:** Se forma con las etiquetas `<tbody></tbody>` Es donde se colocan las celdas con los datos de la tabla.
- **Pie:** Se crea con las etiquetas `<tfoot></tfoot>`. Siempre va a aparecer bajo el cuerpo Se puede usar bajo las columnas para totales, resumen o notas.
- **Las filas:** Se marcan con las etiquetas `<tr></tr>`, que contienen las celdas marcadas como `<td></td>`, dentro de estas se escriben los datos.

Como ves la única sección con un estilo diferenciado es la sección `<thead>` las restantes deben redefinirse con un estilo CSS o etiquetas de formato si se quieren resaltar, por ejemplo, para las secciones **caption** o **tfoot**.

```
<table style="width:300px">
<caption>Mis libros</caption>
<thead>
<tr>
<th style="width:59%">categoria</th>
<th style="width:41%">ejemplares</th>
```

```

</tr>
</thead>
<tbody>
<tr>
<td>Infantiles</td>
<td>12</td>
</tr>
<tr>
<td>Drama</td>
<td>10</td>
</tr>
<tr>
<td>Historia</td>
<td>23</td>
</tr>
<tr>
<td>Terror</td>
<td>14</td>
</tr>
</tbody>

<tfoot style="font-weight:bold">
<tr>
<td>Total</td>
<td>59</td>
</tr>
</tfoot>
</table>

```

Por supuesto todas las secciones no son imprescindibles. Las únicas que no deben faltar son <table>, <tr> y <td>. Es decir, las que definen la tabla y permiten colocar el contenido.

Existe una sección que no se ha usado en el ejemplo y que puede tener cierta utilidad. Se trata de los grupos de columnas, es una sección <colgroup></colgroup> que contiene elementos <col>. En principio se usa para estilos, con ella puedes establecer el fondo y el ancho de las columnas, aunque esto también puedes hacerlo con las cabeceras de columnas.

```

<table>
<colgroup>

<col span="2" style="background: lightgrey">
<col style="background: aliceblue">
</colgroup>
<tbody>
<tr>
<td>Texto</td>
<td>Informática</td>
<td>ESO</td>
</tr>
<tr>
<td>Ejercicios</td>

```

```
<td>Matemáticas</td>
<td>Bac</td>
</tr>
</tbody>
</table>
```

El elemento `colgroup` no puede afectar al estilo de celdas o filas, pues no son sus elementos hijos. Realmente este elemento no es excesivamente útil, es más práctico usar estilos CSS.

Reformando tablas

Las uniones o fusiones de celdas se consiguen con los atributos **colspan**, celdas en distintas columnas, y **rowspan**, para celdas en distintas filas. El valor de estos atributos es el número de celdas que van a combinarse.²⁸

Por ejemplo, en el sitio de libros que estamos usando para los ejemplos tienes una tabla con la situación de libros por tipos:

```
<table style="width:400px" border="1">
<caption>
Situación de libros </caption>
<colgroup >
<col style="width: 25%">
<col style="width: 25%">
<col style="width: 25%">
</colgroup>
<thead>
<tr>
<th >Género</th>
<th >Tipo</th>
<th >Lugar</th>
</tr>
</thead>
<tbody>
<tr >
<td rowspan="3" >Texto</td>
<td >ESO</td>
<td>1A</td>
</tr>
<tr>
<td >Bachillerato</td>
<td>2B</td>
</tr>
<tr>
<td>Universidad</td>
<td>2C</td>
</tr>
<tr>
<td rowspan="2">Lectura</td>
```

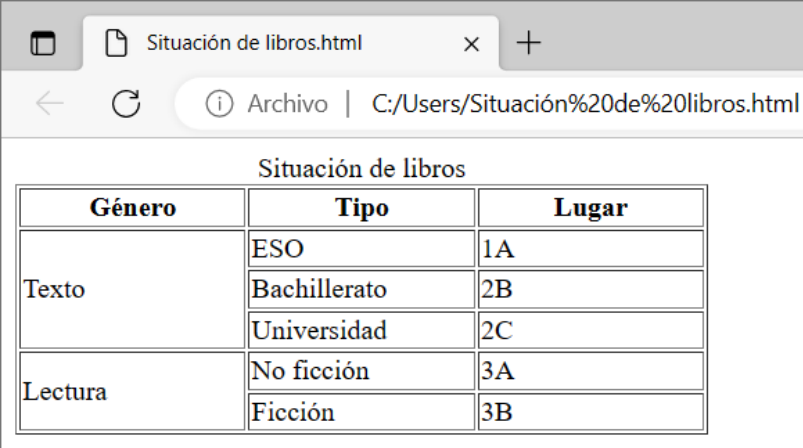
²⁸HTML 5 deja el formato de las tablas totalmente al uso de estilos CSS, aunque reconoce las antiguas tablas de HTML 4 por compatibilidad

```

<td>No ficción</td>
<td>3A</td>
</tr>
<tr>
<td>Ficción</td>
<td>3B</td>
</tr>
</tbody>
</table>

```

En esta otra combinación unimos dos celdas de la sección de cabecera, siguen existiendo dos columnas, pero en la celda superior se unen. En este ejemplo verás que se ha puesto bordes para que se vea mejor el efecto de la combinación. Las combinaciones de celdas solo tienen una condición: que las celdas sean contiguas. Por lo demás no hay límites: se pueden hacer combinaciones de celdas en varias partes de una tabla para lograr una presentación de datos de acuerdo con nuestras necesidades.



The screenshot shows a web browser window with the title 'Situación de libros.html'. The address bar shows the file path 'C:/Users/Situación%20de%20libros.html'. The table displayed is titled 'Situación de libros' and has three columns: 'Género', 'Tipo', and 'Lugar'. The table content is as follows:

Género	Tipo	Lugar
Texto	ESO	1A
	Bachillerato	2B
	Universidad	2C
Lectura	No ficción	3A
	Ficción	3B

19. Los Formularios HTML

Los formularios son el método usado en las páginas web para recibir datos de los usuarios. Pueden usarse para enviar correo a través de una página, para recoger comentarios de los visitantes, para crear un blog o un foro, para hacer búsquedas y un largo etcétera.

Son necesarios en todas aquellas situaciones en las que el usuario deba facilitar algún dato a la página web para que esta haga algo. Si estoy en una página de búsquedas necesito un formulario para que el visitante me diga que desea buscar.

Y por usuario en este caso no nos referimos solo a los visitantes del sitio, sino también a un administrador que deba actualizar una base de datos con información necesaria para presentar en su página web.

Un ejemplo sencillo de formulario:

```

<form id="alojamiento" method="post" action="/editor/verdatos.php">
<p>Titulo
<input name="titulo" type="text">
</p>
<p>Tipo <label>
<input type="radio" name="tipo"
value="E" id="digital">
Ebook</label>
<label>
<input type="radio" name="tipo"
value="P" id="papel">
Papel</label><br>
<br>
<input type="submit" name="button" id="button" value="Enviar">
</form>

```

Como ves en ese código de ejemplo el formulario es un elemento delimitado por las etiquetas **<form>****</form>**. Yo lo divido en tres partes:

- **Encabezado** con datos de gestión del formulario, es la etiqueta **<form>** con sus atributos.
- **Elementos del cuerpo**, donde el usuario interacciona escribiendo o clicando para marcar valores, en los **campos** que lo forman.
- **Elementos de botones** para enviar los datos o resetear y borrarlos para volver a empezar.

En general el formulario necesita un programa para procesar los datos que se escriben en él, suelen ser programas del lado servidor que se ejecutan en éste, aunque también pueden ser enviados a scripts que se ejecutan en el ordenador del usuario. Recuerda que una página web solo sirve para mostrar información estática, si hay que hacer algo más o utilizas información dinámica se necesitan programas o scripts.

Todos los elementos de un formulario poseen estos atributos comunes:

- **name:** Sirve para identificar al elemento.
- **value:** Es el valor del elemento.
- **disabled:** Bloquea el elemento y no deja que se entre datos en él.
- **form:** Contiene el atributo **id** del formulario al que pertenece el elemento.

Nos detendremos en cada parte de este elemento para ver sus funciones:

<form method="" action=""></form>: Son las etiquetas de inicio y fin, entre ellas están los controles. Los atributos esenciales son method y action

- **method:** es la forma en que se pasan los datos. Tiene dos valores posibles GET y POST. El primero pasa los datos en la url, se ven en la barra de direcciones del explorador. POST pasa los datos al programa dentro de la llamada, o sea, no se ven.
- **action:** es el nombre del programa o script que procesará los datos entrados en el formulario. O sea, el que recibirá el contenido del formulario y los utilizará según diga el programa: almacenarlos en una base de datos, actualizar la página, realizar una búsqueda....

Con esto ya tendremos definido el formulario, ahora queda definir donde y como se deben teclear los datos. Estos son los campos que forman el formulario. Si rellenas un formulario en papel, el lugar donde escribes son el equivalente material de los campos del formulario. Fácil.

Para comenzar a practicar y entender los formularios usaremos el control más elemental y versátil los elementos **input**

<input>: Es el campo básico del formulario, sirve para realizar acciones como escribir o finalizar el formulario. Existen muchos tipos, y admite multitud de atributos. Para empezar a usarlo y entender los ejemplos basta con esta información sobre él:

Tipo	Explicación
text	Un campo usado para escribir texto o números: datos alfanuméricos.
radio	Para elegir una de varias opciones
check	Para campos tipo si/no
button	Son botones: para enviar los datos o para borrar o cancelar lo escrito

Existen más tipos de datos para INPUT estos son los básicos aceptados por cualquier navegador. Pero HTML5 introduce más posibilidades y muy interesantes. Estas son aceptadas por los navegadores más extendidos (Firefox, Chrome, Explorer):

Tipo	Explicación
password	muestra asteriscos en lugar del dato tecleado
email	Valida el dato como un email
number	Para entrar números (puede validar valores)
range	Valores en un rango de forma gráfica
url	Valida el dato como una dirección web

Existen otras para introducir colores, fechas, ...

```

<form id="Alta" method="post"
action="/editor/anotar.php">
Titulo
<input name="titulo" type="text">
</p>
<p>Autor
<input name="autor" type="text">
</p>
<p>Clase<br>
<label>
<input type="radio" name="clase"
value="De texto" id="Clase_t">
Texto</label>

```

The screenshot shows a web browser window with the title 'Ejemplo formulario b.html'. The address bar shows 'Archivo | C:/Users/Ejemplo'. The form contains the following elements: a text input for 'Titulo', a text input for 'Autor', a 'Clase' section with two radio buttons labeled 'Texto' and 'Lectura', a 'Tipo' section with two checkboxes labeled 'Ebook' and 'Papel', a text input for 'Más info:', and two buttons at the bottom labeled 'Enviar' and 'Reset'.

```

<label><input type="radio" name="clase" value="De lectura"
id="Clase_l">
Lectura
</label></p>

```

```

<p>Tipo<br>
<label>
<input type="checkbox" name="tipoE" value="ebook"
id="ebook">Ebook</label>
<label>
<input type="checkbox" name="tipoP" value="papel" id="papel">
Papel</label>
<p>
<label>Más info:
<input type="url" name="info" id="info">
</label>
</p>
<input type="submit" value="Enviar">
<input type="reset" value="Reset">
</form>

```

Que es un ejemplo de un formulario completo

Como ves es muy simple. Ahora se trata de entender como se crea un formulario mínimo. En este ejemplo hay datos de texto (Titulo y Autor) datos de opción excluyente (Clase), de opciones (Tipo) y una url (Más info) para que veas como se auto chequea el campo cuando entras una url no válida. Y todo acaba con un **botón para enviar** los datos al programa escrito en **action** (el atributo **action** indica una URL (dirección) a la que enviar los datos del formulario. Esta URL debe ser un elemento capaz de recibir los datos del formulario y procesarlos. Normalmente es un programa escrito en un lenguaje de **programación web** (PHP, Java, ...) que se ejecutará en el servidor web).

Validar datos

Pero antes de enviar los datos para ser procesados por action, puedes verificar si son correctos con un sistema muy sencillo. Le añades un evento `onsubmit="return validar()"`. Este script (**validar**) se debe encargar de leer los datos de los campos del

formulario y verificar que son correctos. Si todo está bien devolverá **true** (return true) si algo no fue bien devolvería **false** (return false) y el formulario no se enviaría.

```
<form id="Alta" method="post" action="/editor/verdatos.php"
onsubmit="return validar()">
Titulo
<input name="nombre" type="text">
<input type="submit" value="Enviar">
<input type="reset" value="Reset">
</form>
<script>
function validar(){
    if(event.target.nombre.value == "Pedro")
return false;
else
return true;
}
</script>
```

(En este ejemplo no puedes enviar el nombre "Pedro")

Ampliación del concepto de formularios en HTML 5

Hemos visto que en los formularios se usaban los argumentos básicos de un formulario: **method** y **action**. Pero este elemento es un componente de la página complejo: es un contenedor de campos de datos. Como cualquier otro elemento acepta los atributos globales de HTML y además posee una serie de atributos propios que aumentan su flexibilidad y utilidad.²⁹ Aquí te los resumo todos y los usaremos en un ejemplo. Recuerdo que los argumentos no son obligatorios.

Attribute	Description
accept-charset	Código de caracteres usados. Lo habitual es usar el juego UTF-8, sistema internacional y el ISO-8859-1 para idiomas latinos. Por defecto se usa el de la página en que se encuentre el formulario
action	Dirección del programa al que se envían los datos del formulario. suele ser un programa escrito en php o asp, y se ejecuta en el servidor.
autocomplete	Indica si los campos aceptan o no auto relleno.
enctype	Se indica como deben codificarse los datos cuando se envían por el método POST, no se usa en el caso de GET. <ul style="list-style-type: none">• text/plain: para depurar• multipart/form-data: para enviar archivos• application/x-www-form-urlencoded: valor por defecto

²⁹ Es más seguro usar el método POST para el envío de datos al programa de gestión

Attribute	Description
method	Es el método como se envían los datos. El valor por defecto es GET <ul style="list-style-type: none"> • GET: los datos se agregan a la url del programa: validar.php?nombre="Juan"&edad="12". Por defecto. • POST: los datos van con la petición al servidor de forma invisible para el usuario. Es más seguro
name	Es el nombre que se le da al formulario para referenciarlo en otros elementos o en scripts.
novalidate	Si aparece este valor el formulario se envía sin validar, es decir, sin comprobar los campos con condiciones.
target	Se refiere a donde se abrirá la página resultante del procesamiento de los datos. Por defecto este valor es _self : la misma ventana donde está la página con el formulario <ul style="list-style-type: none"> • _self : Ventana actual • _blank: Ventana nueva • _parent: Ventana padre de la actual • _top: Ventana padre de todas • frame: nombre de un iframe

Excepto **accept-charset** y **autocomplete**, los restantes atributos pueden ser sustituidos por el correspondiente atributo formaction, formmethod, formenctype, formnovalidate, formtarget de los elementos **<input type="submit">**,

<input type="image"> o **<button>**, los usados para cerrar el formulario y enviar los datos. Especialmente importante es el atributo **enctype** cuando el formulario es utilizado para enviar archivos (**input type="file"**). En esta utilización es necesario que este atributo tenga el valor **multipart/form-data**.

```
<p>No validar los datos</p>
Acepta cualquier valor para url (no valida) <form
action="/editor/verdatos.php" novalidate >
Web de títulos:
<input type="url" name="web">
<input type="submit" value="Enviar" name="envio">
</form><br>
```

```
Este otro avisa si no la url no es correcta <p>Ejemplo de target</p>
<form action="/editor/verdatos.php" target="_blank">
Web de títulos: <input type="url" name="web">
<input type="submit" value="Enviar" name="envio">
</form>
```

En el primer ejemplo tienes el atributo novalidate y cuando envías una url mal escrita el formulario te avisa y te obliga a colocar una url válida.

En el segundo ejemplo la página de resultado del formulario se abre en otra pestaña del navegador.

Entrada de datos

La otra parte fundamental en un formulario son los campos o controles de datos. Has visto en la introducción a los formularios como utilizar algunos tipos de elementos:

<input>: para entrar datos, existen otras formas de hacerlo como listas de valores o textos largos, entre otros.

<select>: Permite ofrecer una lista cerrada de opciones, es como la típica lista desplegable de Windows. No permite teclear texto que no se encuentre en la lista. Dentro de select se pueden establecer grupos de valores.

<textarea>: Si el texto de INPUT se nos queda pequeño podemos usar el área de texto, un cuadro que abarca varias líneas para entrar el texto que deseemos. Entre otros admite los atributos **rows** (filas) y **columns** (columnas) para definir su tamaño.

<datalist>: Ofrece una lista de valores predeterminados para un campo <input>. Al seleccionar el input se despliega la lista con valores sugeridos. Es un autocompletar con valores determinados por la página. Este elemento se asocia mediante su atributo **id** y a un **input** con el mismo valor en el **list**.

Algunos de estos elementos poseen atributos para modificar su comportamiento o aspecto.

Estudia este ejemplo de uso de estos campos. El tamaño del campo de texto (textarea) está definido como 4 filas y 40 columnas.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
</head>
<body>
<form action="/editor/alquilar.php" method="post">
<label>Titulo <select name="Titulo" >
<option value="Los asquerosos">Los asquerosos</option>
<option value="La reina roja">La reina roja</option>
<option value="Solaris">Solaris</option>
</select>
</label>
<p>
<label>Días <input type="number" name="Dias" onfocusout="coste.value
= Dias.value*2+' Euros'"></label>
</p>
<p>
Precio: <output name="coste" for="Dias">
Euros</output>
</p>
<p>Observaciones</p>
<textarea name="Observaciones" cols="40" rows="4">
Puedes anotar cualquier observación
Observaciones
```

```

<textarea name="Observaciones" cols="40" rows="4">
Puedes anotar cualquier observación </textarea>
<label>Escribe la marca de tu Ebook:
<input list="ebooks" name="Ebook">
<datalist id="ebooks">
<option value="Xiaomi">
<option value="Kobo">
<option value="BQ">
</datalist>
</label>

<input type="submit" value="Enviar">
</form>

```

(Uso de diferentes campos de entrada de datos)

Cuando pruebes el código anterior fíjate que el control **datalist** se parece al **select**, pero, a diferencia de éste, **datalist** admite que teclees un texto que no esté en la lista.

Otros elementos

Aparte de los campos de entrada o de acción contruidos con el elemento input, existen otros componentes de los formularios para facilitar su uso, hacerlos más accesibles y más intuitivos. Alguno lo has visto en los ejemplos que hemos venido utilizando.

<output></output>: Es un campo de salida, es un control actualizable por el formulario mediante algún script. Se usa para mostrar el resultado de una operación o de una acción del usuario. Este valor no es enviado con los datos del formulario al programa que lo procese, pues no es un dato provisto por el usuario.

- **for**: contiene una lista separada por espacios de los identificadores de los elementos con que se actualiza su valor
- **name**: nombre del control, necesario para el script que lo actualiza.
- **form**: id del formulario al que pertenece.

<label></label>: Se trata de un elemento de texto explicativo sobre el elemento del formulario al que esté ligado. Se puede usar conteniendo al elemento **input** enlazado o fuera de él. Incluso puede estar fuera del formulario. Sus atributos específicos son:

- **for** Su valor es el id del control input al que se refiere, utilizado cuando el label no contiene al input.
- **form** Su valor es el id del formulario al que pertenece. Utilizado cuando el label está fuera del form.

<fieldset></fieldset>: Permite agrupar visualmente un conjunto de controles relacionados entre sí. Dibuja un marco que encierra a los controles agrupados incluidos en el fieldset.

- **disabled**: Si aparece este valor el conjunto de controles está deshabilitado, no pueden editarse
- **name**: Es un nombre para identificar al conjunto de campos

- **form:** Contiene el id del formulario al que pertenece, usado cuando el fieldset se define fuera del elemento **form**. Los controles incluidos deben contener este mismo atributo.

<legend></legend>: Es un elemento hijo de fieldset utilizado para ponerle un título a un conjunto de campos agrupados. el texto que se introduzca en este elemento aparece sobre el borde de fieldset.

Agrupamos los datos del cliente en un **<fieldset>** con nombre dado por el elemento **<legend>**. Se usa un elemento **<output>** para mostrar el total del alquiler calculado con el número de días.

Como ves el elemento **input** para días tiene un atributo **onchange**, su valor es un código javascript que se ejecuta cuando cambia el valor del campo (**onchange** es un evento, una señal que avisa de que ha ocurrido algo, en este caso un cambio).

```
Datos del cliente<br>
<form action="/editor/alquilar.php" method="post">
Precio del alquiler 2€/dia <p>
<label>Días <input type="number" name="Dias" onchange="coste.value =
Dias.value*2+' Euros'"></label><br>
Total: <output name="coste" for="Dias">
Euros</output></p>
<fieldset>
<legend>Datos del cliente</legend>
<label>
Nombre <input type="text" name="nombre"></label><br>
<label>Apellidos <input type="text" name="apellidos"></label><br>
</fieldset>
<input type="submit" value="enviar">
</form>
```

(Elementos avanzados de form)

Al probar este formulario observa que el valor calculado en **output** (total) no es pasado al programa señalado en **action**. Solo se usa en el formulario a modo de información para el usuario. Si ese valor fuera necesario el programa que gestiona el formulario lo debería volver a calcular.

20. Entrada de datos en Formularios HTML

La entrada de datos al formulario se puede realizar con distintos tipos de elementos. El elemento input es quizás el más completo en esta tarea, pero tiene limitaciones y es completado con los elementos **select**, **datalist** y **textarea**.

Al igual que con los restantes datos enviados al programa de servidor, la información recogida con estos tipos de campo también debe ser **filtrada y comprobada** para evitar fisuras de seguridad en el sitio web.

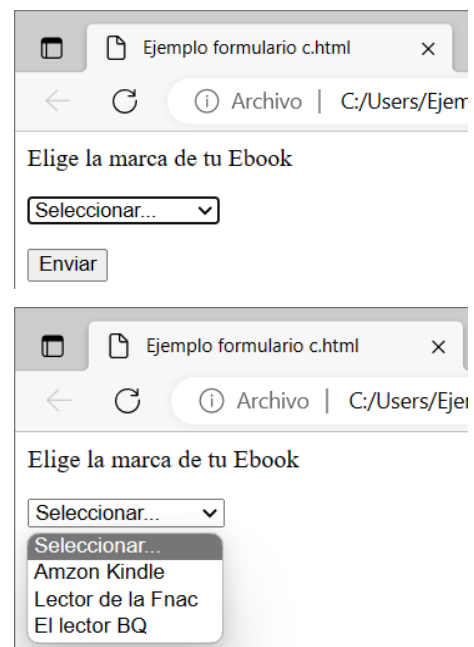
Select: listas desplegables

El elemento `select`³⁰ permite mostrar una lista de valores desde la que el usuario puede seleccionar uno o más de ellos. Al seleccionar el elemento se despliega una lista con valores entre las que el usuario selecciona uno (por defecto) o varios (si posee el atributo **multiple**).

Atributo	Uso
<code>autocomplete</code>	Permite completar automáticamente lo tecleado por un usuario. No tiene sentido en este elemento
<code>disabled</code>	Bloquea el control para que no pueda editarse
<code>multiple</code>	Indica si pueden seleccionar más de un valor
<code>required</code>	Si es obligatorio rellenar este campo
<code>size</code>	Alto del control, en número de filas.

La lista de valores se construye con elementos `<option></option>`, esta etiqueta encierra el texto que se verá en la lista, mientras que en su atributo **value** anotamos el valor que se enviará al programa que procesa el formulario. También puede llevar el atributo **selected**, para aquella o aquellas opciones (si es múltiple) que deban aparecer como seleccionadas por defecto. Si no se usa el valor seleccionado por defecto es el primero de la lista.

```
<form action="/editor/verdatos.php"
method="post">
<label>Elige la marca de tu Ebook
</label>
<p>
<select name="marcas">
<option value="">Seleccionar...</option>
<option value="kindle">Amzon
Kindle</option>
<option value="fnac">Lector de la
Fnac</option>
<option value="bq">El lector BQ</option>
</select>
</p>
<input type="submit" value="Enviar">
</form>
```



En este ejemplo se dan tres posibles respuestas. Observa que el valor (**value**) es lo que envía el formulario al programa, el texto de cada opción solo se usa para mostrarlo en el navegador.

³⁰ Para campos con un conjunto cerrado de respuestas posibles lo recomendable es usar el elemento **select**

Datalist: listas completar

El elemento **datalist** se parece al select porque ofrece una lista de valores, pero en este caso son valores sugeridos. El valor enviado con el formulario es el valor tecleado en un campo input.³¹

Este elemento solo funciona con los elementos input numéricos o de textos. No posee ninguno de los atributos usados en los elementos de los formularios.

```
<form action="/editor/verdatos.php" method="post">
<label for="generos">¿Qué genero de literatura prefieres?</label>
<div>
<input name="generos" list="generos">
<datalist id="generos" >
  <option value="Ficcón">
  <option value="Histórica">
  <option value="Divulgación">
  <option value="Poesía">
</datalist>
</div>
```

En este ejemplo puedes elegir de la lista de opciones o teclear un valor no contenido en esa lista. Se trata solo de ofrecer algunos valores posibles. Si has usado el formulario con anterioridad y se coloca el atributo **autocomplete= "on"**, el navegador añade a esta lista otros valores usados anteriormente.

Texto multilinea

Observa que el **tamaño del área de texto es modificable por el usuario**, en la esquina inferior derecha verás un pequeño triángulo: si pulsas y arrastras con el ratón verás que el tamaño del área se modifica.³² Puedes desactivar este redimensionado mediante la regla de estilo CSS `resize`:

```
<form action="/editor/verdatos.php" method="post">
<label for="sugerencias">Escribe tus sugerencias<p>
<textarea name="sugerencias" cols="40" rows="5" maxlength="140"
placeholder="Máximo 140 caracteres" style="resize:none">
</textarea>
</p>
<input type="submit" value="Enviar">
</form>
```

(Un área de texto no dimensionable)

Elemento INPUT en el formulario

El elemento **input**³³ quizás sea el elemento fundamental de cualquier formulario, no solo es la *cajita* en la que escribimos, abarca también los botones tipo enviar (**submit**) o resetear (**reset**) y más. Como **habrás** leído en estas páginas existen muchos tipos de campo INPUT, estos tipos corresponden con el tipo de dato que se

³¹ Las listas de completar permiten ofrecer valores posibles para un campo del formulario. El navegador puede completar esta lista

³² Los campos de texto deben filtrarse muy bien en el programa de gestión

³³ los elementos **input** son las más variados y flexibles para entrar datos en un formulario

espera en el campo: texto, número, un password, un email.... Estos tipos permiten que el navegador los valide y los complete con posibles valores para facilitar su uso.

Existen elementos input para cualquier tipo de dato que puedas necesitar. Las opciones de lista (datalist), el selector de fechas o el de colores facilitan enormemente el uso del formulario por parte del usuario. Cada input pasa su valor al programa que gestiona el formulario, configurado mediante el atributo **action** o **formaction**.

Una **cuestión elemental de seguridad** es que este programa siempre debe verificar los datos que recibe. Los datos deben ser del tipo y contenido previstos por el programa. Hay que tener mucho cuidado por ejemplo cuando esos datos se van a usar en una página que use una base de datos, se debe evitar que puedan introducirse órdenes SQL que comprometan la seguridad del sitio.

Botones input

Los inputs pueden actuar como botones para cerrar y enviar el formulario o para realizar alguna acción en la página. Este grupo de inputs son los tipos **button**³⁴, **submit**, **reset**, **image**. Y todos comparten los siguientes atributos:

atributo	sustituye a	uso
for	--	valor del id del form al que pertenece el control
formaction	action	url del programa
formnovalidate	novalidate	no comprobar los campos
formmethod	methof	forma de envío GET/ POST
formtarget	target	Destino de la página de resultado
formenctype	enctype	Tipo de código de caracteres

Excepto el atributo **form**, los restantes sobrescriben al correspondiente atributo del elemento **<form>**. El atributo **for** es útil cuando el botón está fuera del elemento **<form></form>**.

Dentro de este grupo de botones nos encontramos con diferentes tipos (**<input type= ...>**)

- **submit**: Al pulsar este botón se envía el formulario al programa que lo procesa con todos los datos tecleados.
- **reset**: Al pulsar todos los campos se resetean, se borra lo escrito.
- **image**: Es un botón **submit** pero con una imagen en lugar de un texto. Cuando se usa este tipo de botones para enviar los datos del formulario también se envía las **coordenadas X, Y** de la imagen donde se ha pulsado con el ratón. Posee algunos atributos extra por la imagen.
 - **Alt**: Un comentario sobre la imagen.

³⁴ El elemento HTML **<button>** puede utilizarse en lugar de cualquiera de los botones creados con **input**.

- **width, height:** para el tamaño: ancho y alto.
- **src:** la url de la imagen.
- **button:** Es un botón genérico al que se le asocia un script mediante un evento.

En ellos el atributo **value** es el texto que aparece en el botón, salvo en el caso de tipo imagen.

En HTML existe un elemento bastante similar a estos botones input, se trata del elemento **<button>/button>**. Su atributo type permite que funcione como:

- Un botón sin comportamiento predeterminado (**type="button"**), sería como un **<input type="button">**.
- Un botón de envío (**type="submit"**) sería similar a **<input="submit">**.
- Un reset (**type="reset"**) sería similar a un **<input type="reset">**.

Por lo demás comparte todos los atributos señalados para este tipo de elementos input.

Inputs numéricos y fechas

Estos elementos input engloban los campos que permiten entradas desde el teclado con datos numéricos y de tipo fecha y hora. HTML5 añade atributos que enriquecen y facilitan la entrada de información. Los atributos comunes a estos controles son:

Atributos específicos de números y fechas		
autocomplete	on off	Habilita autocompletar para cuando se está escribiendo texto. Hace más cómoda la entrada de texto, pero también puede ser inseguro.
list ³⁵	datalist_id	Referencia a un elemento <datalist> con las opciones o valores predefinidos para un input. Esos inputs son listas desplegables.
max	número o fecha	Valor máximo admitido en el campo.
min	número o fecha	Valor mínimo admitido en el campo
placeholder	texto	El valor escrito en este atributo aparecerá dentro del campo y desaparece automáticamente al entrar con el cursor. Útil para dar información al usuario sobre lo que se espera que teclee.
readonly		El campo es de solo lectura, no puede escribirse en él.
required		Este campo debe ser rellenado antes de enviar el formulario

³⁵ El uso del atributo **list** requiere un campo **datalist**, excepto para los colores

step	number	Para campo numéricos nos da el intervalo entre valores consecutivos. Por ejemplo, si es 5 y los valores a teclear están entre 1 y 10 solo serán válidos 1, 6, 11, ...
------	--------	---

Los tipos de input que comparten estos atributos son campos que aceptan números, fechas y rangos de valores. Estos elementos son los tipo: number, range y relativos a fechas y horas: date, month, week, time, local-date, local-time.

Algunos de estos atributos se usan también en otros tipos de input, y con la misma finalidad. O sea, vas a volver a verlos. Y ahora este ejemplo de formulario con este tipo de datos.

```
<form action="/editor/verdatos.php"
method="post">
<p>Alquiler de EBooks por días</p>
<label>Fecha de comienzo:</label>
<input type="date" name="inicio"></label>
<label><br>
<br>
¿Cuantos días (de 1 a 4)? :
<input type="number" min="1" max="4"
name="dias" required>
</label><br>
<br>
<label for="precio">Precio </label>
<input type="text" value="1.25 €/dia"
name="precio" readonly><br>
<br>
<label>Califica este servicio (0 a 10)</label>
<input type="range" min="0" max="10"
name="puntos">
</label></br>
<input type="submit" name="Enviar">
</form>
```

Como ves el control **range** queda como una especie de slider o deslizador para números enteros por defecto entre 0 y 100, límites que puedes modificar con los atributos **min / max**.

Inputs para texto

Por defecto el elemento input considera lo que se teclea como texto³⁶ (type="text"), pero HTML 5 permite distinguir algunos tipos particulares dentro del texto y los valida de acuerdo a unas reglas adecuadas a cada tipo como, **email**, **password**, **search**, **url**, **telephone**. Ya se ha visto el uso de url en algún ejemplo para mostrar el uso de validate. Ahora nos detendremos un poco en ellos.

³⁶ El uso de expresiones regulares permite controlar el formato de datos introducido en un campo

Atributos específicos para textos		
autocomplete	on off	Habilita autocompletar para cuando se está escribiendo texto. Hace más cómoda la entrada de datos, pero también puede ser inseguro.
list	<i>datalist_id</i>	Referencia del elemento <datalist> que contiene las opciones o valores predefinidos para un input. Esos inputs son listas desplegables.
maxlength	<i>número</i>	Número máximo de caracteres del texto tecleado.
minlength	<i>número</i>	Número mínimo de caracteres del texto tecleado.
pattern	<i>regexp</i>	Expresión regular o patrón al que debe ajustarse el texto tecleado.
placeholder	<i>texto</i>	El valor escrito en este atributo aparecerá dentro del campo y desaparece automáticamente al entrar con el cursor. Útil para dar información al usuario sobre lo que se espera que teclee.
readonly		El campo es de solo lectura, no puede escribirse en él.
required		Este campo debe ser rellenado antes de enviar el formulario
size	<i>número</i>	Ancho en caracteres del cuadro de INPUT.

Los definidos como email, url, telephone son realmente tipo text con un **pattern** ya definido. Las expresiones regulares son una especie de reglas que usan comodines para definir un formato de texto determinado. Por ejemplo, un código postal podría definirse como una serie de cinco números, esta definición se escribiría como [0-9]{5}. Puedes usar las expresiones regulares de JavaScript.

El tipo **email** admite un atributo denominado **multiple** que permite escribir en el campo varios emails separados por coma.

El tipo **text (o search)** admite un atributo denominado **dir** para indicar idiomas escritos de derecha a izquierda (**rtl**) frente a los escritos de izquierda a derecha (**rtl**).

El input type="password" simplemente oculta lo que se teclea y muestra asteriscos o alguna forma para ocultar el texto a la vista.

Inputs para opciones

Los inputs no siempre son campos para teclear datos también existen inputs en los que dar una respuesta tipo si/no, activado/desactivado, es decir, donde solo hay dos valores posibles. Son los inputs **checkbox** (casilla de verificación) y **radio button** (opción). El control checkbox permite una entrada solo de un conjunto de valores,

mientras que radio button se usa para entradas excluyentes entre sí³⁷. Los argumentos exclusivos de estos inputs son:

Atributos específicos de checkbox y radio		
required		Este campo debe ser rellenado antes de enviar el formulario
checked	checked	Para campos tipo checkbox o radio , indica que la opción está seleccionada por defecto.

Son controles típicos para respuestas como sexo (M/F), trabaja (si/no), y donde se puedan elegir varias respuestas. Por ejemplo, en una compra puedes elegir una de tres formas de pago (**radio button**), pero puedes comprar el artículo en uno o varios formatos (**checkbox**)

Opciones para la compra

```
<form action="/editor/verdatos.php">
<p>Ha comprado el libro &quot;Los asquerosos&quot;.</p>
<p>Forma de pago</p>
<p>
<label>
<input type="radio" name="Pago" value="visa" id="Formato_0"
required>
```

Visa

```
</label>
<label>
<input type="radio" name="Pago" value="paypal" id="Formato_1">
Paypal</label>
<label>
<input type="radio" name="Pago" value="transferencia"
id="Formato_1">
Transferencia</label>
</p>
```

Formatos en que lo desea recibir

```
<p>
<label>
<input type="checkbox" name="Formato_0" value="Ebook"
id="formato_0">
```

Ebook

```
</label>
```



```
<label>
<input type="checkbox" name="Formato_1" value="Papel"
id="formato_1">
Papel </label>
```

³⁷ Estos controles se pueden activar o desactivar también pulsando sobre la etiqueta label, mejorando la usabilidad del control

```

<br>
<label>
<input type="checkbox" name="Formato_2"
value="Audiolibro" id="formato_2">
Audiolibro</label>
<br>
</p>
<p>
<input type="submit" value="enviar">

```

Ejemplo formulario e.html

Opciones para la compra

Ha comprado el libro "Los asquerosos".

Forma de pago

☐ Visa ☐ Paypal ☐ Transferencia

Formatos en que lo desea recibir

☐ Ebook
☐ Papel
☒ Audiolibro

enviar

Los inputs de tipo **radio** deben agrupar las opciones posibles para que al elegir una se desactiven las otras. Esto se logra usando el mismo atributo name para todas las opciones posibles. En el ejemplo se usa Pago (name="pago").

El input de tipo **checkbox** no necesita esa unificación de nombres, sin embargo, es recomendable usar nombres que hagan ver que son valores que se refieren al mismo dato, para conseguir un código más claro y para el programa de gestión del formulario.

Enviar archivos

Mediante un formulario un usuario puede enviar archivos al servidor.³⁸ Para eso se usa un tipo especial de input, el **type=file**. Para lograr que el archivo se envíe es necesario que el formulario use el método POST y el enctype (tipo de cifrado) en **multipart/form-data**. De esta manera el programa de gestión recibirá el archivo y lo podrá almacenar en el servidor o hacer lo que esté programado.

Este campo utiliza el explorador del sistema operativo para que acceder al archivo que quieras subir. Si se usa la opción múltiple puedes seleccionar varios archivos usando la tecla CTRL y el ratón (en Windows).

Los atributos particulares para este tipo de control son:

Atributos específicos para file		
accept	<i>Extensión</i> audio/* video/* image/* <i>tipo_de_medio</i>	Indica el tipo de archivos que se esperan para subir. Actúa como un filtrado de la lista de archivos que el explorador mostrará a la hora de seleccionar los que se quieran subir. Es importante señalar que el navegador no comprueba el tipo de archivo enviado . Por cuestión de seguridad el programa que procesa el formulario debe verificar que el archivo enviado se corresponde con la extensión y que es del tipo que se espera. Con el nombre de archivo también se envía información sobre el tamaño y el tipo
required		Este campo debe ser rellenado antes de enviar el formulario

³⁸Al permitir la subida de archivos al servidor se debe vigilar muy bien el tipo de archivos que se envían

multiple	Permite seleccionar varios ficheros. En Windows se usa la tecla CTRL con el ratón
----------	---

```
<form action="/editor/verdatos.php" method="post">
<label>Seleccionar el archivo para subir: <br><br>
<input type="file" name="documento" accept=".doc">
</label><br><br>
<input type="submit" name="enviar" value=" Subir ">
</form>
```

(Un formulario puede usarse para subir archivos)

El filtro **accept** tan solo **filtra los archivos mostrados** en el explorador que se abre para seleccionar el que quieras subir. El programa **debe verificar que se trata del tipo adecuado**. Las extensiones son poco o nada fiables para asegurar de que se trata de un archivo tipo texto o imagen.

Otros tipos

Solo quedan dos tipos de campos input: **hidden** y **color**.

El input de tipo **hidden** es un campo oculto, que no se ve en el formulario. No tiene ningún atributo específico y se utiliza para pasar valores de control al programa que procesa el formulario, valores que el usuario no necesita ver ni puede modificar. No hay que dejarse engañar por lo de campo oculto (hidden), cualquier usuario que acceda a las herramientas del desarrollador o a la vista de código verá su contenido, por tanto, no debe usarse para códigos de acceso o cualquier información sensible.

El tipo **color** como se nombre indica es una utilidad para poder seleccionar códigos de color con un **color-picker**, o selector de colores. En este campo puedes seleccionar un color y el cuadro del campo se verá con ese color. No se ve el código que se envía al programa del formulario. Admite el atributo **list** con un datalist conteniendo códigos hexadecimales para listar (si no se usa un datalist el navegador ofrece una gama de colores por defecto).

```
<form action="/editor/verdatos.php" method="post">
<label for="funda">Seleccionar color: </label><br><br>
<input type="color" name="funda" id="funda" list="colores"
value="#ff0000">
<datalist id="colores">
<option value="#ff0000">
<option value="#0000ff">
<option value="#00ff00">
</datalist><br><br>
<input type="submit" name="enviar" value=" Subir ">
</form>
```

Tipos de INPUT en el formulario

A modo de resumen este es el conjunto de atributos que puede aceptar un elemento input de cualquier formulario

Nombre	Valores	Uso
accept	<i>Extensión de tipo</i> audio/* video/* image/* <i>tipo_de_medio</i>	En el tipo de campo file (campos usados para nombres de ficheros) valida el nombre ficheros. Ojo para subir archivos al servidor hace falta también un script que realice el proceso.
alt	<i>text</i>	Para tipo image da un texto alternativo a la imagen, Es similar al atributo alt de un elemento image de HTML.
autocomplete	on off	Habilita autocompletar para cuando se está escribiendo texto. Hace más cómoda la entrada de texto, pero también puede ser inseguro.
autofocus	autofocus	El campo que tiene este atributo es al que va el cursor cuando se carga la página
checked	checked	Para campos tipo checkbox o radio , indica que el campo debe estar seleccionado por defecto.
disabled	disabled	El campo INPUT no está habilitado, no puede usarse.
form	<i>form_id</i>	Si el elemento INPUT NO está entre las etiquetas <form></form>, este atributo indica a que formulario pertenece el INPUT
formaction	<i>URL</i>	Se usa para botones de enviar (tipos SUBMIT o IMAGE) Permite establecer el script que procesará los datos. Tiene preferencia sobre el action del elemento form
formenctype	application/x-www-form-urlencoded multipart/form-data text/plain	Se usa para botones de enviar (tipos SUBMIT o IMAGE) y especifica la forma en que se codifican los datos.
formmethod	get post	El formulario envía los datos de los campos como GET o como POST. El primero es como si fueran argumentos de la url, mientras

		el segundo va más escondido, y es útil cuando se mueven muchos datos.
formnovalidate		Para marcar el dato como que no debe ser comprobado al ser enviado.
formtarget	_blank _self _parent _top <i>nombre de frame</i>	Al enviar los datos el programa puede devolver al usuario a una página (gracias, error, pedidos...) Aquí se indica si debe ser la misma página (_self) una nueva (_blank), Los demás valores valen para cuando se usan frames
height	<i>pixels</i>	Para tipo image este valor define el alto de la imagen. Esta imagen es en realidad un botón de enviar.
list	<i>datalist_id</i>	Referencia del elemento <datalist> que contiene las opciones o valores predefinidos para un input. Esos inputs son listas desplegables.
max	<i>número o fecha</i>	Para un valor numérico (number) o tipo fecha (date) o rangos (range) permite establecer un valor máximo. Útil para validar datos.
maxlength	<i>número</i>	Ancho máximo del campo medido en caracteres.
min	<i>número o fecha</i>	Para un valor numérico (number) o tipo fecha (date) o rangos (range) permite establecer un valor mínimo. Útil para validar datos.
multiple		Con este atributo se admite que se puedan entrar varios valores para un mismo input. Estos valores se separan entre sí con comas.
name	<i>texto</i>	Esto se usa para dar un nombre al elemento input.
pattern	<i>regexp</i>	Expresión regular o patrón al que debe ajustarse el texto entrado. Esto es muy conocido para programadores Perl o usuarios de Unix y Linux. Es un paso más allá de los comodines típicos que usamos por ejemplo para listar * y ?
placeholder	<i>texto</i>	El valor escrito en este atributo aparecerá dentro del campo y desaparece automáticamente al entrar con el cursor. Útil para dar información al usuario sobre lo que se espera que teclee.

readonly		El campo es de solo lectura, no puede escribirse en él.
required		Este campo debe ser rellenado antes de enviar el formulario
size	<i>número</i>	Ancho en caracteres del cuadro de INPUT.
src	<i>URL</i>	Para tipo IMAGE sirve para la dirección donde se encuentra la imagen.
step	<i>number</i>	Para campo numéricos nos da el intervalo entre valores consecutivos. Por ejemplo si es 5 y los valores a teclear están entre 1 y 10 o solo serán válidos 1, 6, 11, ... Por defecto es 1, pero si quieres decimales en el input type number puedes usar step con decimales: 0.1, 0.01, 0.001 para uno, dos o tres decimales respectivamente
title	texto	El texto aquí escrito vale como indicador, para cuando un campo no acepta un valor por que la autovalidación de form lo impide.
type	button checkbox email file hidden image number password radio range reset submit text url	Posibles tipos del campo INPUT. Estos valen para los navegadores mayoritarios (Firefox, Explorer y Chrome), pero hay más posibilidades definidas en la norma: color date datetime datetime-local month search tel time week
value	<i>text</i>	Valor por defecto del campo.
width	<i>pixels</i>	Para tipo image este valor define el ancho de la imagen. Esta imagen es en realidad un botón de enviar.

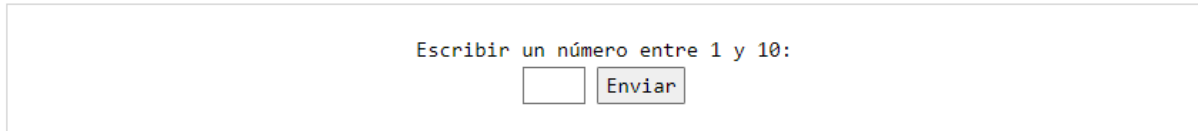
Ejemplos de uso de INPUT

En estas líneas tienes algunos ejemplos para que veas la forma de utilizar los controles INPUT en el elemento form.

Vemos el uso de **atributos para validar entradas numéricas**. Por ejemplo una entrada limitada a un valor entre 1 y 10.

```
<form action="demo.php">
Escribir un número entre 1 y 10:
<input type="number" name="nota" min="1" max="10">
<input type="submit" value="Enviar">
</form>
```

Este código crea este formulario



Formulario visualizado: "Escribir un número entre 1 y 10:" seguido de un campo de entrada de tipo "number" con límites de 1 a 10 y un botón "Enviar".

El uso de los límites máximo y mínimo crea un selector de valores enteros (la doble flecha arriba/abajo a la derecha del cuadro) para entrar el dato correcto, aunque también lo puedes teclear a mano. Si te pasas de los límites te aparecerá un aviso al enviar el formulario.

Y hablando de rangos, mira esto algo parecido, ahora entre 0 y 10

```
<form action="demo.php">
Calificación
<input type="range" name="nota" min="0" max="10">
<input type="submit" value="Enviar">
</form>
```

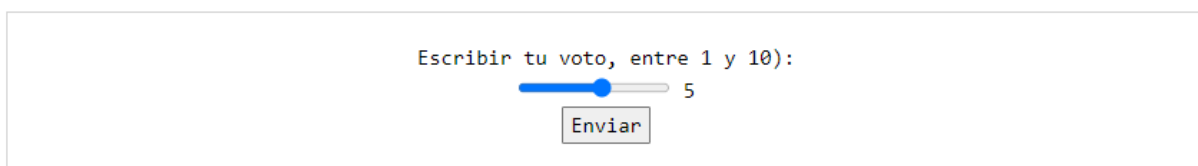


Formulario visualizado: "Calificación" seguido de un controlador de rango (slider) entre 0 y 10 y un botón "Enviar".

Como ves este sistema es más gráfico, aunque no se ven los valores, para eso se necesita un poco de JavaScript. Es útil cuando no se necesita precisión en los datos.

Mira esta modificación usando un campo output

```
<form action="demo.php" oninput="rango.value=votos.value">
Escribir tu voto, entre 1 y 10:<br>
<input type="range" name="votos" min="0" max="10" title="votos">
<output name="rango" for="nota">5</output>
<input type="submit" value="Enviar">
</form>
```



Formulario visualizado: "Escribir tu voto, entre 1 y 10):" seguido de un slider para "votos" y un campo "output" que muestra el valor "5", con un botón "Enviar".

En este otro ejemplo tiene un formulario que pide el número de libros de una compra, los libros van en cajas de 5 por tanto no puedes pedir 6 libros ni 4 ni 11. Sería algo como esto:

```
<form action="demo.php" method="post">
Número de libros (van en cajas de 5):
<input type="number" name="cantidad" min="0" max="100" step = "5">
<input type="submit">
</form>
```

Número de libros (van en cajas de 5):

En este ejemplo se usa una expresión regular para permitir solo un tipo de valores, concretamente un código que serán 3 letras seguidas de un guion y 3 números como abc-234 o frw-567

```
<form action="demo.php" method="post">
<input name="codigo" type="text" pattern="[a-z]{3}-[0-9]{3}">
<input type="submit" name="submit" id="submit" value="Enviar">
</form>
```

Este formulario se vería como sigue:

Un ejemplo de formulario donde se usan campos agrupados, algo bastante útil si quieres tomar información sobre algo con varios apartados, por ejemplo:

```
<form action="demo.php" method="post">
<fieldset>
<legend>Cliente:</legend>
Nombre:      <input name="nombre" type="text"><br>
Apellido 1º: <input name="apel1" type="text"><br>
Apellido 2º: <input name="apel2" type="text">
</fieldset>
<input type="submit" name="submit" id="submit" value="Enviar">
</form>
```

Este código crea este formulario

Cliente:

Nombre:

Apellido 1º:

Apellido 2º:

Los datos quedan agrupados en el recuadro con nombre **Cliente**, es algo visual para remarcar que los datos están relacionados entre sí. Se crea un elemento tipo bloque rodeado por un borde. Las etiquetas claves son **legend**, para darle un nombre al grupo y **fieldset** que conforma el grupo propiamente dicho. Para el proceso del formulario esta estructura no afecta en nada: los datos son enviados individualmente como si no existiera el **fieldset**.

21. Imágenes HTML

Puede que nuestro contenido necesite imágenes para llegar al usuario o puede que queramos aligerar la página o hacerla más atractiva con el uso de alguna fotografía o dibujo. En el estándar HTML 5 la inclusión de imágenes sigue siendo muy simple, basta con usar la etiqueta **** con los atributos adecuados para mostrar correctamente una imagen en la página.

Los atributos específicos de este elemento son pocos (aparte de los globales como `class` y `style`)

Atributo	Valor	Uso
<code>alt</code> ³⁹	text	Texto alternativo a la imagen. Indispensable
<code>aspect-ratio</code>	width/height	Controla la relación ancho:alto (w/h) de la imagen
<code>crossorigin</code>	código	Se usa para imágenes de otro sitio utilizable en el canvas
<code>height</code>	pixels	Alto de la imagen
<code>ismap</code>	ismap	Para imágenes usadas en mapas con soporte servidor
<code>longdesc</code>	URL	Un URL donde se da una descripción detallada de la imagen
<code>sizes</code>	pixels	Tamaños adaptaos al medio de visualización
<code>src</code>	URL	URL de la imagen. Indispensable
<code>srcset</code>	URL	Imágenes alternativas
<code>title</code>	texto	Es un atributo global. Permite que al situar el ratón sobre la imagen se muestre un texto.
<code>usemap</code>	#mapname	Hace que la imagen se usa en un mapa de lado cliente
<code>width</code>	pixels	Ancho de la imagen

Como ves HTML 5 enriquece este elemento y elimina algunos atributos, como las alineaciones y espacios alrededor, que ahora se han sustituido por reglas de estilos CSS.

Los atributos imprescindibles son **src** y **alt**. Si no indicas alto y ancho, el navegador usa el tamaño original de la imagen.

³⁹ El atributo **alt** es considerado tan importante como el **src**. Los buscadores lo valoran bastante y además es importante para mejorar la accesibilidad de la página

Si como tamaño de la imagen usas un porcentaje respecto contenedor donde se encuentra verás que puedes hacer que la imagen nunca se salga de la pantalla aunque la veas en un móvil. Pruébalo con este código

```

```

Si pruebas este código verás que al cambiar el tamaño de la ventana (moviendo la barra central) la imagen se adapta al nuevo tamaño. Si no quieres que supere su tamaño original y pierda calidad, puedes usar además el estilo **max-width**. Le pones el ancho real de la imagen y su tamaño no pasará de ese valor. El atributo **height** (la altura) no está puesto, por defecto es auto, es decir se adapta al ancho.

Para señalar el ancho y alto dispones también de la relación de aspecto (aspect-ratio) que te permite poner la relación entre ancho y alto de la imagen, así al colocar el ancho automáticamente se calcula el alto. La forma de escribirlo es bien fácil aspect-ratio: 16/9

```

```

En este ejemplo puedes comprobar que al cambiar el ancho o el alto la proporción se mantiene

Figure: Fotos con pie y título

Aunque no es exclusivo para imágenes, el elemento **figure** se adapta muy bien a mostrar imágenes con un texto al pie sin complicar el diseño con combinaciones de bloques.⁴⁰

Este elemento es un contenedor compuesto o sea que contiene otros elementos: usamos uno para una imagen y un espacio para el comentario de la imagen (el pie de foto). Puede contener elementos de HTML anidados. No deja de ser un bloque.

```
<figure>

<figcaption style="text-align:center">
Usa un book, salva un árbol </figcaption>
</figure>
```

(Imagen con pie de foto)

No es exclusivo de imágenes, en lugar de img podemos poner lo que queramos: un bloque div, una tabla, un texto, un video. El apartado **figcaption** es el que podemos usar para colocar un texto referente en este caso a nuestra imagen. Este elemento puede aparecer también arriba de la imagen si lo ponemos arriba después de la etiqueta **<figure>**. Incluso se pueden colocar dos uno arriba y otro abajo, a modo de título y pie.

```
<figure style="width:200px">
<figcaption style="text-align:center">Los árboles: la
vida</figcaption>
```

⁴⁰ No es un bloque exclusivo para imágenes, es utilizable para videos, bloques div, tablas... Y facilita poner un texto al pie

```

</figure>
```

(Imagen con título de foto en la parte superior)

Prueba con los ejemplos a poner un título y un pie de foto en el mismo cuadro figure.

Como te digo más arriba figure puede contener cualquier elemento de HTML, es aconsejable para esquemas, códigos, notas tipo post it o en general cualquier elemento al que pueda hacerse referencia desde otro lugar como un índice o tabla de contenido. Además su estructura de bloque hace que pueda cambiarse posición sin alterar el flujo de la página. Los ejemplos de código de este tutor son elementos figure.

Picture: adaptar las imágenes

Uno de los usos del elemento picture⁴¹ es seleccionar la imagen según el formato aceptado por el navegador. Este elemento es un contenedor con elementos hijos

- **source**: imágenes optativas en función del formato y/o el tamaño de visor disponible
 - **type**: tipo mime para esta imagen. Vale para tipos de imágenes.
 - **media, srcset**: imágenes según condición de medios.
- **img**: imagen por defecto si no existe una imagen adecuada entre los elementos source

Básicamente se trata de indicar imágenes alternativas para diferentes formatos. Por ejemplo

```
<picture>
<source srcset="/imgs/user.svg" type="image/svg+xml">

</picture>
```

(Picture para adaptar el tipo MIME)

En este ejemplo el navegador mostrará la imagen user.svg si soporta el formato **SVG** de imágenes. Si no lo soporta mostrará la imagen por defecto indicada en el elemento ****. Como ves se trata de declarar un tipo MIME y la imagen correspondiente. Para comprobar como funciona, en la ventana de prueba cambia el type por cualquier tipo o texto (type="image/algo") y verás como muestra el logo.

Imágenes adaptativas con HTML

El diseño web responsive o adaptativo no es más que conseguir que la página web sea visible y contenga las mismas funcionalidades independientemente del dispositivo utilizado para acceder a ella.

Desde el punto de vista del diseño de la página la gran diferencia entre los diferentes dispositivos es el tamaño y la resolución del visor. Una pantalla de 25 pulgadas no es lo mismo que otra de 7. Sin embargo, las páginas han de ser igualmente utilizables en ambos entornos.

Al cambiar de entorno de visualización los elementos de la página se deben redistribuir y los tamaños de las imágenes deben adaptarse. Para esto se puede

⁴¹ Picture sigue el esquema de las etiquetas **video** y **audio**

usar JavaScript y CSS, pero también es posible hacerlo con las nuevas características que ofrece HTML5. Se trata de los atributos **srcset**, **media** y **size**.

Estos atributos permiten ajustar el tamaño de imagen al tamaño del visor⁴², sería una imagen responsive, pero también permite cambiar de imagen para que se ajuste mejor al contexto. Por ejemplo, si reducimos el tamaño de una imagen podemos perder detalles, puede ser mejor usar una versión recortada.

Para ver como funcionan con el editor de esta web (el que se usa con el botón probar código) mejor usa el navegador Firefox, Chrome no muestra los ejemplos adecuadamente. Para comprobar puedes modificar el tamaño de la mitad de la página donde se ven los resultados o puedes usar el inspector de código del navegador para probar distintos tamaños de pantalla.

Los tamaños de un elemento img como cualquier otro puede ser modificado dinámicamente mediante CSS. En esta página vemos métodos HTML, sin CSS

Imágenes responsive: srcset

El atributo **srcset** permite usar imágenes alternativas en función de la pantalla que el usuario esté usando. Simplemente le indicamos una url de imagen y el tamaño real de la imagen.

```

```

(Uso del atributo srcset con resoluciones)

¿Como se interpreta esto? Fácil, foto1x.jpg es la imagen más pequeña con 400 pixels de ancho, foto2x.jpg es de tamaño medio 667pixels, foto3x.jpg tiene un tamaño mayor, 996 pixels. Pero ojo la unidad usada es **w** no los habituales pixels.⁴³ Esta unidad se refiere a esta los pixels de definición del visor. El navegador adaptará el tamaño mostrado al tamaño de la pantalla. Si el navegador no puede hacer esto porque no reconoce este atributo pues usa la imagen por defecto señalada en **src**

- [Ventana de 400px de ancho y una foto pequeña](#)
- [Ventana de 720px de ancho y una foto mediana](#)
- [Ventana de 1048px de ancho y una foto grande](#)

Pero es más práctico usar las alternativas con valor del ancho de pantalla. En este caso en lugar de valor de resolución usamos tamaños reales de la imagen usando como unidad de medida w, valor relativo al ancho de la ventana.

Con esto las imágenes se adaptarán perfectamente al ancho del dispositivo, la va escalando. Existe una variante que adapta la imagen a la definición del visor, algo que depende del dispositivo físico. en lugar de anchos en unidades w se indica para cada imagen la densidad de bits del dispositivo 1x, 2x o 3x. A mayor definición

⁴² Los tamaños de un elemento img como cualquier otro puede ser modificado dinámicamente mediante CSS. En esta página vemos métodos HTML, sin CSS

⁴³ Las reglas CSS permiten igualmente controlar el tamaño de la imagen ajustándola a su contenedor utilizando anchos relativos (porcentajes)

mayor puede ser la imagen usada. Estas cifras indican el **número de pixels de la pantalla física por cada pixel de tamaño CSS**.

Las reglas CSS permiten igualmente controlar el tamaño de la imagen ajustándola a su contenedor utilizando anchos relativos (porcentajes).

Imágenes responsive: sizes

Podemos tener aún más control sobre el tamaño mostrado de la imagen mediante este otro atributo, **size**⁴⁴. En este caso además del **srcset** indicamos la imagen que debe elegir el navegador según el tamaño del visor (condición de medios).

```

```

(Uso del atributo srcset con dimensiones)

Para cada imagen posible existe un atributo size con dos partes:

1. Una **condición** entre paréntesis que comprueba el tamaño de ventana
2. La **anchura** disponible para mostrar la imagen

Al usar los atributos srcset y sizes el navegador actúa como sigue

1. **Comprueba** el tamaño de la ventana
2. Verifica la **primera condición** de medios que se cumple
3. Determina el **ancho disponible** para la imagen
4. Elige la imagen con tamaño más próximo de las declaradas en **srcset**

En el ejemplo propuesto si la anchura de la pantalla es 400px o menos (max-width: 400px) se usa foto1px, si esa condición no se cumple pasa a la siguiente: si la pantalla tiene un ancho hasta 720px se usa la imagen más próxima a 720px y si está por encima de 720px se usa la imagen foto.jpg.

La última condición no sería necesaria, pero la dejo a fin de que se vea como funcionan las condiciones.

Este ejemplo no se ve bien con página de probar códigos, deberías crearte una página web y abrirla con ventanas de explorador de diferentes tamaños para ver el efecto. Para verlo ahora:

- [Ventana de 400px de ancho y una foto pequeña](#)
- [Ventana de 720px de ancho y una foto mediana](#)
- [Ventana de 1048px de ancho y una foto grande](#)

Si no ves el efecto puede ser por problemas de cache, lo mejor es probarlo en páginas independientes.

⁴⁴ El uso de sizes no es indispensable para el escalado. Realmente srcset es suficiente

Imágenes responsive: Picture

Aún existe una tercera vía para conseguir que nuestras imágenes se ajusten a la pantalla. Se trata del elemento **<picture>**, es parecido a una abreviatura de los atributos **srcset** y **sizes**⁴⁵. Este es el que mejor funciona como podrás comprobar si lo ejecutas en el editor.

Se trata de un contenedor con elementos hijos

- **source**: imágenes optativas en función del formato y/o el tamaño de visor disponible
 - **type**: tipo mime para esta imagen. Vale para [tipos de imágenes](#).
 - **media**, **srcset**: imágenes según condición de medios
- **img**: imagen por defecto si no existe una imagen adecuada entre los elementos **source**

Este contenedor no deja que el navegador decida que imagen usar, se la indicamos nosotros en el código con los atributos **srcset** y **media** explicados en el apartado anterior. Además este contenedor permite también adapta el formato de la imagen al formato admitido por el navegador. Incluye un elemento **img** que actuará como imagen por defecto si no se puede cumplir con el atributo **type** ni con el **srcset - size**.

```
<picture>
<source media="(max-width: 400px)"
srcset="/tutorhtml5/imgs/foto1x.jpg">
<source media="(max-width: 720px)"
srcset="/tutorhtml5/imgs/foto2x.jpg">
<source media="(min-width: 721px)"
srcset="/tutorhtml5/imgs/foto3x.jpg">

</picture>
```

(Uso del elemento PICTURE para imagen responsive)

Utiliza una imagen de tamaño fijo en función del ancho de la ventana. En el ejemplo

- si el ancho es igual o menor de 400px usa la imagen foto1x.jpg
- si el ancho es mayor de 400px (**por la primera regla**) e igual o menor a 720px usa la imagen foto2x.jpg
- si el ancho es igual o mayor que 721 usa la imagen foto3x.jpg

Volvemos a la importancia del **orden de las condiciones de media**. Si la condición **max-width: 720px** estuviera en primer lugar al acceder a la página en un dispositivo de 300p de ancho de pantalla se mostraría la foto2x.jpg, porque se cumpliría que el ancho no alcanza los 720px.

Para ver en funcionamiento este método puedes usar estos enlaces que abrirán ventanas de distintos tamaños

- [Ventana de 400px de ancho y una foto pequeña](#)
- [Ventana de 720px de ancho y una foto mediana](#)
- [Ventana de 1048px de ancho y una foto grande](#)

⁴⁵ El uso de **sizes** y **picture** es adecuado para mostrar imágenes de un tamaño fijo pero adaptado al dispositivo

Aparentemente son métodos iguales, pero hay una diferencia sutil: el uso de `srcset-sizes` deja que el navegador intervenga en la imagen usada y adapte mejor los tamaños, en `Picture` es el diseñador el que decide todo. Si se trata de cambiar una imagen por otra el elemento `picture` es más indicado, si predomina una cuestión de resolución y tamaños puede ser mejor usar `srcset-sizes`.

Imágenes y enlaces

En la página dedicada a los enlaces se explica que estos se escriben mediante la etiqueta `<a>`. Entre estas marcas se sitúa el elemento que señala donde pulsar con el ratón para ir al sitio definido en el atributo `href`. Este contenido puede ser un texto, como *pulse aquí*, o una imagen.⁴⁶ A diferencia de la versión anterior de HTML, las imágenes que son a su vez enlaces no se destacan con un borde azul. Esta imagen se crea de forma normal no necesita nada en especial: o sea, etiqueta `img` con los atributos necesarios, como mínimo el `src` y `alt`. Incluso pueden usarse texto e imagen como cuerpo del enlace. Si se desea se le puede dar formato con estilos CSS

```
<a href="genero-terror.htm">

</a>
```

Se ve así:



Mapas de imagen lado cliente

Los enlaces en las páginas web pueden estar contruidos en base a un texto o a una imagen⁴⁷, como habrás visto en el apartado de enlaces. En el caso de usar imágenes HTML nos ofrece una interesante posibilidad: hacer que cada parte de la imagen contenga una URL diferente, es decir, apunte a un lugar diferente. Así se crea un mapa de enlaces.

Este sería el código del ejemplo

```

<map name="personajes">

  <area shape="circle" coords="126,52,42"
    href="../mapa/alegria.htm"
    alt="Alegría" title="Alegría">
```

⁴⁶ Las imágenes son perfectas para crear botones personalizados

⁴⁷ Los mapas de imagen son una forma muy visual de mostrar enlaces en una infografía por ejemplo

```

<area shape="rect" coords="146,116,223,214"
href="../mapa/tristeza.htm"
alt="Tristeza" title="Tristeza">

<area shape="circle" coords="213,68,47"
href="../mapa/asco.htm"
alt="Asco" title="Asco">

<area shape="rect" coords="81,103,142,206"
href="../mapa/furia.htm"
alt="Furia" title="Furia">

<area shape="poly" coords="26,32,22,84,44,150,66,168,85,81,57,33"
href="../mapa/miedo.htm"
alt="Miedo" title="Miedo">

</map>

```

(Mapa de enlaces con una imagen)

Como ves se trataría de una imagen normal, pero que lleva un atributo **usemap** cuyo valor es el nombre de un mapa definido con el elemento **map**. En este elemento se describen tres tipos de figuras geométricas definidas con el atributo **shape**, **circle** para círculo, **rectangle** para rectángulo y **poly** para una figura irregular.



Cada figura se define por un sistema de **coordenadas cuyo origen está en la esquina superior izquierda de la imagen**. Para definir cada área se usa:

- **rectángulo**: coordenadas de sus esquinas superior derecha e inferior derecha.
- **círculo**: coordenadas del centro y el radio del círculo
- **polígono**: coordenadas de cada punto que forma el polígono.

Finalmente cada forma lleva un atributo **href** para el enlace al que debe ir cuando se pulsa dentro de ella. Por supuesto tratándose de imágenes es obligado utilizar el atributo **alt** y recomendable utilizar **title** como ayuda al usuario.

Mapa de imagen de servidor

Este tipo de mapas usa un programa alojado en el servidor para procesar la pulsación sobre la imagen. Al pulsar se llama al programa que recibe las coordenadas del punto donde se ha pulsado, dependiendo de las coordenadas hará

una cosa u otra. Es necesario crear ese programa, lo que se puede hacer en perl, php, o cualquier otro lenguaje del servidor.

Un enlace basado en una imagen que apunta a una página php que recibe como argumentos el lugar donde se pulsa tendría el código:

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

El programa en php, que tan solo nos dice donde se ha pulsado, sería

```
# Extraer argumentos
list($x, $y) = preg_split("/", "/", $_SERVER['QUERY_STRING']);
print "Content-type:text/html\r\n\r\n";
print "<html>";
print "<head>";
print "<title>ISMAP</title>";
print "</head>";
print "<body>";
print "<h2>Pulsado en el punto.: X: $x, Y: $y </h2>";
print "</body>";
print "</html>";
```

(Esquema de un mapa de lado servidor en PHP)

En otros lenguajes tendría otro aspecto pero lo principal es leer los argumentos (coordenadas donde se pulsó al llamarlo) que van en la url, el programa debe extraer esos valores, determinar en qué área se encuentra y abrir la correspondiente página, o realizar la acción que se programe.

22. Vídeo en HTML 5

Hasta ahora si querías poner un video en tu página web lo normal era tirar de extensiones externas (plugins) como Flash o Java, muy dependiente de las prestaciones del navegador. Con el estándar HTML 5 esto ya no es necesario. Este lenguaje de páginas web ya incluye la posibilidad de colocar video sin necesidad de dependencias externas y de manera muy fácil. Incluso resuelve el tema de la diversidad de formatos.

La cosa es tan fácil como usar la nueva etiqueta **<video>**, que define un contenedor dentro del cual indicamos el origen del video que queremos mostrar y su formato.⁴⁸ Mira este ejemplo de código:

```
<video width="320" height="240" src="movie.mp4" controls>
Este navegador no soporta la etiqueta video de html5
</video>
```

Así de sencillo, ya tendrías un video colocado en tu web. Admite más atributos: para autoejecutarse, ejecutarse en bucle, mostrar controles... Si el navegador no entiende la etiqueta **<video>** mostrará la frase escrita. Los atributos que soporta esta etiqueta son

⁴⁸ La inclusión de videos en las páginas es cada vez más frecuente, dada las velocidades que ofrece actualmente la conexión a internet

Nombre	Valores	Uso
autoplay		Para que el video se ejecute automáticamente (ignorado en los navegadores modernos)
controls		Muestra controles de ejecución, pausa, sonido.
height width	pixels	alto (height) y ancho (width) del video
loop		Reproducción en bucle.
muted		Reproducir el video sin sonido
poster	URL	Dirección de la imagen a mostrar mientras el video está descargando o en parada
preload	auto metada none	Le video se carga cuando la página se carga Solo carga metadatos el cargar la página El video NO se carga con la página (lo hace al final)
src	URL	Dirección del video que se va a reproducir.

Con el video se pueden cargar los llamados metadata, que no es más que información que acompaña al video como, por ejemplo: título, autor, álbum..., depende de cada caso

Si el explorador no reconoce la etiqueta video usada se mostrará lo que aparezca entre las etiquetas de apertura y cierre, puede ser un simple texto de aviso o un enlace por ejemplo para descargar el video o enviarlo a una página de explicaciones o una imagen...

No obstante, aún existen más elementos que pueden usarse con los videos en la página web, por ejemplo para indicar el formato o formatos... Lo vemos en la siguiente página.

Los navegadores actuales no hacen caso de la etiqueta **autoplay**, salvo que llevan además **muted**, es decir, solo se acepta **autoplay** pero en silencio. Si le agregas controls el usuario podrá habilitar el sonido.

Video y formatos

El nuevo elemento para colocar videos en una página web permite especificar diferentes formatos de reproducción, esto nos permite asegurarnos la compatibilidad con distintos exploradores.

Esto se logra mediante el componente **<source>** (un elemento dentro del contenedor de videos) La cosa es tan fácil como usar la nueva etiqueta **<video>**, que define un contenedor dentro del cual definimos el origen del video que queremos mostrar y su formato⁴⁹. Mira este ejemplo de código:

⁴⁹ Puedes usar archivos alternativos de videos en función del formato aceptado por el navegador

```
<video width="320" height="240" autoplay>
<source src="movie.mp4" type="video/mp4">
<source src="movie.ogg" type="video/ogg">
<source src="movie.webm" type="video/webm">
Este explorador no reconoce la etiqueta video.
</video>
```

Ejecuta este ejemplo en el editor y modifícalo con los atributos vistos arriba como controls, para aprender con funciona el video en el HTML moderno.

Así de sencillo sobre todo si lo comparamos con el método usado en las versiones anteriores de html, que además exigían el uso de reproductores externos.

Nombre	Valores	Uso
src	url	la dirección web del video a mostrar
type	video/ogg video/mp4 video/webm	Formato de video usado, o sea, el tipo MIME de datos.

Los formatos como avi o flv (de flash) no son admitidos, la única solución es convertirlos en alguno de los formatos señalados aquí. Para hacer esto existen herramientas gratuitas como Freemake, AviDemux, o VLC.

Textos en los videos con HTML 5

Aún tenemos un elemento más dentro del contenedor **<video>**, se trata del **track**⁵⁰ una forma de colocar subtítulos o cualquier otro texto en el visionado del video.

```
<video width="320" height="240" autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
<source src="movie.webm" type="video/webm">
<track kind="subtitles" src="movie-es.vtt" lang="es" label="Español"
default>
<track kind="subtitles" src="movie-en.vtt" lang="en" label="Inglés">
```

Este explorador no reconoce la etiqueta video, puede descargar el video en

```
<a href="movie.mp4" >Descargar video</a>
</video>
```

(HTML5 facilita enormemente la inserción de videos)

En este pequeño ejemplo simplemente hemos señalado que existe una pista para subtítulos que está colocada en el archivo de texto **movie.vtt**. Estos archivos de texto deben tener un formato adecuado que puede ser WEBVTT o TTML (ambos estándares publicados por W3C). Para subtítulos contienen los textos y los instantes

⁵⁰ El atributo track tiene bastante más complejidad, es útil para los doblajes de videos o para subtitular. Mejora la accesibilidad de la página

en que deben aparecer y desaparecer de pantalla. Funciona bien con Chrome y con Mozilla y Edge.

Un archivo vtt se debe guardar con **formato UTF-8**, por los acentos y demás particularidades del idioma:

Se señalan los tiempos en formato hh:mm:ss.ddd (horas, minutos, segundos con dos dígitos y milésimas de segundo, con tres dígitos) y van seguidos del texto que se verá en ese intervalo.

```
WEBVTT 00:00:02.000 -->
00:00:05.000
Bonito paisaje
La campiña 00:00:05.001 -->
00:00:10.000
Y aquí el pajarito
Cantando feliz
```

Los atributos que admite este elemento hijo de video son:

Nombre	Valores	Uso
default		Para señalar la pista usada por defecto
kind	captions	Tipo de dato mostrado:
	chapters	Título
	Descripciones	Capítulos
	metadata	Descripciones
	subtitles	Metadatos para scripts
label	Texto	Subtítulos
src	URL	El título de la pista, para el menú CC de los controles de video, que permite elegir una u otra.
srclang	idioma	El lugar donde se encuentra el archivo de texto con el contenido a mostrar
		Código de idioma (es, en, de, it...)

El uso de track es bastante más extenso de lo aquí indicado pero con estas breves notas es suficiente para comenzar a trabajar con él.

Si activas la opción controls, el usuario podrá cambiar el idioma de los subtítulos mediante el botón de configuración del video.

El estilo del subtítulo es definible usando el pseudo elemento **CSS ::cue** con el podemos modificar los estilos habituales de texto como puedan ser color, tamaño, transparencia, fondo, sombreado e interlineado entre otros.

23. Audio en HTML 5

La cosa es tan fácil como usar la nueva etiqueta **<audio>**,⁵¹ que define un contenedor dentro del cual definimos el origen del video que queremos mostrar y su formato. Mira este ejemplo de código:

```
<audio src="musica.mp3" controls>  
Este navegador no soporta la etiqueta audio video de html5  
</audio>
```

Así de sencillo, ya tendrías un reproductor de audio colocado en tu web.

Admite más atributos: para autoejecutarse, ejecutarse en bucle, mostrar controles.

Muy parecido al elemento video. Si el navegador no puede aceptar la etiqueta audio mostrará la frase escrita.

Los atributos que soporta este elemento multimedia en HTML 5 son:

Nombre	Valores	Uso
autoplay		Para que el sonido suene automáticamente
controls		Muestra controles de ejecución, pausa, sonido.
loop		Reproducción en bucle.
muted		Para no reproducir el sonido
preload	auto metada none	El sonido se carga cuando la página se carga Solo carga metadatos el cargar la página El sonido NO se carga con la página (lo hace al final)
src	URL	Dirección del archivo de sonido que se va a reproducir.
metadata		Datos para los scripts

También en audio existen los metadata, es decir, información extra que acompaña al archivo de sonido como por ejemplo: título, autor, álbum... depende de cada caso.

Si el explorador no reconoce la etiqueta **audio** o ninguno de los formatos de sonio se mostrará el texto que aparezca entre las etiquetas **<audio>**, puede ser un simple texto de aviso o un enlace por ejemplo para descargar el archivo o enviarlo a una página de explicaciones.

Audio y formatos en HTML 5

Al igual que con el elemento videos, el audio puede presentarse en varios formatos y podemos especificarlo en la página para asegurar la compatibilidad con diferentes exploradores.

⁵¹ Las políticas de experiencia de usuario recomiendan no usar audio automático en las páginas web. Solo debe iniciarse cuando lo indique el usuario

Pueden existir varios elementos **<source>**, el elemento hijo de audio, que permiten indicar el archivo de sonido que se debe reproducir para cada formato. Cada explorador reproducirá el archivo compatible. Mira este ejemplo de código:

```
<audio autoplay>
  <source src="musica.mp3" type="audio/mpeg">
  <source src="musica.ogg" type="audio/ogg">
  <source src="musica.wav" type="audio/wav">
Este explorador no reconoce la etiqueta video.
</video>
```

Así de sencillo, son estos tres formatos aseguramos que cualquier explorador reproducirá nuestro archivo de audio. He aquí un resumen de los atributos que permite el elemento **source**

Nombre	Valores	Uso
src	url	la dirección web del archivo de audio
type	audio/ogg audio/mpeg audio/wav	Formato de audio usado, o sea, el tipo MIME de datos. El mpeg es para mp3

Al igual que ocurre con los videos, estos tipos MIME deben estar declarados en el servidor. Una forma es mediante un archivo .htaccess cuyo contenido sería de la forma

AddType audio/mpeg .mpeg

Aunque habitualmente los servidores tienen esta asociación en su configuración.

Un código sería por ejemplo:

```
<audio controls>
<source src="imgs/Baby.mp3" type="audio/mp3">
Este explorador no reconoce el elemento audio
</audio>
```

24. Colocar scripts en la página web con HTML 5

Un **script** es un programa insertado dentro del documento html y que es **interpretado y ejecutado por el navegador del usuario**. Permiten crear **páginas dinámicas**: modificar el comportamiento normal del navegador, validar formularios, realizar efectos visuales, animaciones, crear elementos de la página, etc.

Estos programas **se ejecutan en el ordenador del usuario**, bien sea directamente, cuando el navegador lee el documento HTML para mostrar la página, o cuando se produce un suceso determinado (**eventos**), como puede ser el pulsar sobre un

enlace o mover el ratón o cargar una imagen o modificar el tamaño de la ventana del navegador.

Son importantes por que permiten crear **páginas dinámicas e interactivas**. al ejecutarse en la máquina del usuario y no en el servidor tienen ciertas limitaciones como manejar bases de datos o crear archivos permanentes entre otros. Para tareas que funcionan en el servidor se usan programas escritos en lenguajes como PHP.

El lenguaje usado para crear *scripts* es el **JavaScript**⁵², actualmente normalizado mediante el estándar ECMAScript.

JavaScript

Tras aprender html y CSS, JavaScript⁵³ es la tercera pata en la que se basa el desarrollo web, por eso es necesario mostrar la utilidad de los scripts en las páginas web.

Haremos surgir una ventana que nos muestre el tradicional mensaje "*hola, mundo*". Así podremos ver los elementos principales del lenguaje. El siguiente código es una página Web completa con un botón que, al pulsarlo, muestra el mensaje.

```
<!doctype html>
<html>
<head>
<script>
function Saludo() { alert("¡Hola, mundo!"); }
</script>
</head>
<body>
<button onClick="Saludo()">
Púlsame </button>
</body>
</html>
```

Ahora vamos a ver, paso por paso, que significa cada uno de los *elementos extraños* que tiene en la página del ejemplo:

```
<script>
</script>
```

Esta etiqueta encierra el programa escrito en JavaScript, el **código del programa**. Puedes poner cuantos quieras a lo largo del documento y en el lugar que consideres necesario. Si un navegador no entiende la etiqueta **<script>** escribirá el código como si fuera parte de la página, por eso a veces se encierra entre comentarios `<!-- código del script -->`.

Eso sería un script integrado en el código HTML. Vale para casos muy sencillos.

Lo recomendable es separar código de programa de código HTML. Cada cosa en un sitio.

⁵² Además de Javascript Microsoft intentó usar Visual Basic como base para un lenguaje de script en la web. No funcionó y creó su propio Javascript, conocido como JScript

⁵³ Programar en Javascript es muy fácil, es un lenguaje muy flexible y nada exigente con la sintaxis. Se puede decir que es un lenguaje orientado a objetos, esto te sonará si has usado Java, C++ o Php

El código de programa se puede guardar en un archivo externo y se incorpora a la página web añadiendo a al elemento **<script>** el atributo **src="url_file.js"**. Con esto le decimos al navegador que inserte en la página web el contenido del archivo cuya dirección es **url_file.js**:

```
function Saludo(){ alert("¡Hola, mundo!"); }
```

El sitio donde colocar este **script** (con **src**) es el **head**, cuando el navegador ve esta línea se detiene la carga de la página hasta que termina de cargar el archivo con el **script**. Y esto no es bueno. Puede solucionarse con atributos extra como **async** (la carga del **script** y la página se hacen en paralelo) o **defer** (la carga de página y **script** es en paralelo y el **script** no se ejecuta hasta que la página está totalmente cargada).

O también se pueden ponerse al final del documento, justo antes de la etiqueta de cierre de **body**. Con esto nos aseguramos que el archivo **script** no bloquea la carga de la página y que no se ejecuta hasta que el documento se ha cargado.

Esta es nuestra primera **función** en JavaScript. En el código de la misma vemos que pone **alert** (es un método o acción que puede ejecutar el navegador donde se vea tu página web. Concretamente este método muestra un mensaje en la pantalla, ¿qué mensaje? el que tú le digas, en este caso *¡Hola Mundo!*

```
<button onClick="Saludo()">Púlsame</button>
```

Dentro del elemento botón vemos una cosa nueva: **onClick**. Se llama **evento** y funciona como una marca que le dice al navegador que haga algo cuando se haga click con el ratón. Cuando el usuario pulsa el botón, el evento **onClick** se pone en marcha y ejecuta el código que tenga entre comillas, en este caso la llamada a la función **Saludo()**, que tendremos que haber definido con anterioridad. Y al ejecutarse **saludo()** mostrará en pantalla el mensaje.

Púlsame

Este ejemplo es una pequeñísima muestra de como funcionan los scripts en JavaScript.