

Primeros pasos con HTML5

En esta segunda lección veremos:

- Una introducción a HTML5
- Qué son y cómo funcionan las etiquetas
- Qué es y cómo se hace la anidación de etiquetas
- La estructura básica de una página
- Algunas etiquetas básicas

El objetivo es empezar a familiarizarnos con HTML5 y prepararnos para la siguiente lección en la que configuraremos nuestro Google Chrome para poder empezar a escribir código.

Si crees que ya dominas esta materia puedes probar a hacer el [ejercicio tipo test de esta lección](#). Si sacas un 100% de aciertos puedes pasar a la siguiente, sino te recomiendo que no te la saltes.

Introducción a HTML5

HTML ([Hypertext Markup Language](#)) es un [lenguaje de marcado](#) (*que no es lo mismo que un [lenguaje de programación](#)*) que sirve para definir la estructura y la semántica de nuestra página web (luego veremos que significa esto).

HTML fue creado y es mantenido por una organización sin ánimo de lucro llamada [W3C](#). El W3C es un consorcio formado por [más de 400 empresas](#) (entre ellas las que desarrollan los principales navegadores como Google, Microsoft, Mozilla, Apple...), etc.

Desde el consorcio trabajan continuamente en definir cómo debe evolucionar este lenguaje y otros estándares que conforman *la web*. Posteriormente los fabricantes de navegadores preparan los mismos intentando conseguir que un código funcione igual en todos los navegadores. Aunque desafortunadamente no siempre es así, cada vez es una realidad más cercana.

Por tanto, [a lo largo de los años las versiones de HTML han evolucionado](#): HTML 2.0 (1995), HTML 4.0 (1997), XHTML (2000), **HTML5 (2014)**, etc. con el objetivo de adaptarse a los nuevos tiempos y así dar soporte a nuevas necesidades (estandarización de los sistemas de audio, vídeo, etc).

Snippets interactivos

Vamos a ver una breve introducción al funcionamiento de la interfaz:

The screenshot shows the 'Snippets HTML' interface. It features a navigation bar at the top with buttons for '< Anterior' (1), 'Lección: 1.- Ejemplos (2)' (3), 'Snippets: 1.- Estructura básica' (2), and 'Siguiendo >' (5). Below the navigation bar, there are two main sections: 'Código HTML:' (3) and 'Resultado:' (4). The 'Código HTML:' section contains a code editor with the following code:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Estructura básica HTML5</title>
</head>
<body>
  Hola mundo
</body>
</html>
```

 (7). The 'Resultado:' section shows the rendered output: 'Hola mundo' (4). At the bottom of the 'Resultado:' section, there is a button that says 'EDITAR - CODEPEN.IO' (6). A link '¿Tienes alguna duda?' (7) is located between the code editor and the rendered output. A link 'Abrir en nueva pestaña' (5) is located at the bottom right of the interface. The footer of the interface contains the text: 'Código disponible en Github | Autor: Raúl Jiménez Ortega | Licencia Creative Commons: CC-BY-NC-SA 4.0 International.' (6).

1. Permite navegar entre las diferentes lecciones
2. Permite navegar entre los diferentes snippets
3. Ejemplo de código HTML (con la sintaxis resaltada)
4. Resultado del código (3) en embebido en la página
5. Nos permite abrir la previsualización a pantalla completa
6. Nos permite usar el editor web [Codepen.io](https://codepen.io), un editor bastante popular para experimentar con el código.
7. Enlace a los **issues** de Github donde podremos publicar cualquier duda o problema sobre los ejemplos.

Puedes acceder a esta interfaz a través de la siguiente URL:

[http://libro.cursohtml5desdecero.com/snippets/html/](http://librocursohtml5desdecero.com/snippets/html/)

Etiquetas

En el último estándar de HTML (HTML5) existen [más de 100 elementos](#) que nos permitirán crear etiquetas. Como comentaba al inicio del curso no los veremos todos, de hecho no es habitual encontrar a nadie que los conozca todos, ni siquiera los que llevamos tantos años haciendo webs, lo importante es saber dónde buscarlos y saber cómo usarlos.

Por esto vamos a empezar por entender qué aspecto tienen. Lo primero es saber que las etiquetas sólo pueden ser de dos tipos:

1) **Las que tienen una apertura y un cierre** como en el siguiente caso:

```
<elemento atributo="valor">Contenido</elemento>
```

Por ejemplo:

```
<a href="http://www.google.com">Google</a>
```

En este caso decimos que: *"tenemos un elemento **a** donde el valor del atributo **href** es <http://www.google.com>, y que su contenido es **Google**".* No hace falta que te preocupes aún por el significado, luego haremos incapié en esto.

Si nos fijamos las etiquetas **siempre** están contenidas entre los símbolos `<` `>`, y el cierre sólo incluye el nombre del elemento precedido de una barra lateral `/` (p.e. `</elemento>`).

2) Por otro lado están los **elementos auto-contenidos** (los que no se cierran explícitamente):

```
<elemento atributo="valor">
```

Por ejemplo:

```

```

Es importante destacar que el *atributo* y el *valor* son opcionales.

Vamos a ver algunos ejemplos de nombres de etiquetas:

- **elemento:** *html, head, meta, title, body, img...*
- **atributo:** *charset, src, alt, ...*

- **valor:** *UTF-8*, *"url"* (la URL a un recurso), *"texto"*, ...

A mi siempre me gusta tener una "**chuleta**" (o cheatsheet) a mano que resuma todos los elementos y algunos de los atributos que aceptan. Pero por ahora no hace falta que te distraigas con esto, lo importante es que entiendas el formato.

Anidación de etiquetas

También es importante saber que unas etiquetas pueden contener a otras (una o varias), o como se suele decir "que se pueden anidar".

Por ejemplo:

```
<head>
  <title>Título de la página</title>
  <meta name="author" content="Raúl Jiménez Ortega - @hhkaos">
</head>
```

En este caso vemos que la etiqueta **head** tiene como contenido a otra etiqueta **title**. En este caso se dice que:

- La etiqueta *head* es el **padre** de la etiqueta *title* y *meta*.
- Y que la etiqueta *title* y *meta* son **hijas** de la etiqueta *head*.
- La etiqueta *title* y *meta* son **hermanas**.

Si nos fijamos, además, la etiqueta anidada (*title*) está en una nueva línea y con un nivel de [indentación](#)/sangrado mayor. Esto es así por una [convención](#) mundial a la que se ha llegado para que los programadores escribamos código de una manera similar, tanto para hacernos más fácil y comprensible el código cuando este crezca, como para cuando tengamos que compartirlo con otros programadores.

Orden de apertura y cierre

Cuando anidamos etiquetas unas dentro de otras es muy importante cerrarlas *en orden*. Esto quiere decir que la primera etiqueta en cerrarse tiene que ser la última que se abrió, así por ejemplo este ejemplo sería **incorrecto**:

```
<p>El orden es <strong>muy importante</p></strong>
```

La forma **correcta** de hacerlo sería:

```
<p>El orden es <strong>muy importante</strong></p>
```

Recordemos que hay etiquetas que no es necesario cerrarlas por lo que esto sería **correcto**:

```
<p>  
  El orden es <strong>muy importante</strong>.<br>  
  Así introducíamos un salto de línea.  
</p>
```

Otros aspectos

Me gustaría puntualizar otros dos detalles:

1. No todas las etiquetas son anidables entre sí; por ejemplo: una etiqueta **body** no puede contener una etiqueta **head**. Pero no te preocupes de momento por esto, en otra lección veremos cómo podemos saber qué etiquetas son anidables entre sí.
2. Es importante indentar bien el código porque en muchos casos nos encontraremos muchos niveles de anidación, etiquetas que contienen etiquetas que a su vez contienen etiquetas, etc. ya que **no existe un límite máximo de anidamiento**.

Estructura básica de una página

Primero me gustaría hacer una pequeña aclaración sobre terminología que voy a usar, diferenciaremos:

- **Página web:** como una página individual dentro de un sitio web (p.e: rauljimenez.info/contacto)
- **Sitio web (o web):** como el conjunto de todas las páginas en las que podemos navegar dentro de un mismo dominio (p.e: rauljimenez.info es un sitio web que incluye: rauljimenez.info/experiencia, rauljimenez.info/contacto, etc).

Dicho esto, toda *página web* que hagamos en HTML5 debe tener una estructura inicial similar a la siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Título de la página</title>
</head>
<body></body>
</html>
```

A continuación explicamos la función que cumple cada etiqueta:

- `<!DOCTYPE html>` : indicar al navegador que el código HTML en el que está escrita la página es en la versión 5, osea que es HTML5. [+info](#)
- `<html lang="es">... </html>` : indica la raíz del documento y **todas** las etiquetas deben estar incluidas dentro. Además se especifica el idioma en el que está escrita, es = Español ([+idiomas](#)).
- `<head> ... </head>` : se usa para envolver otras etiquetas que ofrecen información principalmente a: el navegador, a los buscadores y a otras páginas (como pueden ser redes sociales, etc). La información especificada dentro del *head* no se muestra *dentro*¹ de la página web que ve el usuario.
- `<meta charset="UTF-8">` : indica al navegador qué tipo de caracteres contiene la página. Con el atributo charset indicamos cuál de [todos los juegos de caracteres disponibles](#) usamos. Con el valor **UTF-8** podremos crear contenido en la mayoría de los sistemas de escritura: español, inglés, francés, etc.
- `<title> ... </title>` : indica el título de nuestra página. Este se muestra en: la pestaña del navegador, el enlace que indexan los buscadores, etc.
- `<body> ... </body>` : contiene todo el contenido visible por el usuario *dentro* de nuestra

página.

Observa que la etiqueta *html* contiene dos hijas: *head* y *body*, esto ya no es obligatorio en HTML5 ya que se puede omitir las etiquetas **html**, **body** y **head**, pero por convención es recomendable usarlas.

Aquí puedes ver el **ejemplo interactivo**: [Lección 1 - Snippet 1](#)

Aclaraciones:

1. Cuando digo **dentro** me refiero al contenido de la página, lo que no incluye la pestaña del navegador ni la barra de direcciones.

Etiquetas básicas

Para terminar esta lección vamos a aprender el significado de ocho de las etiquetas que con más frecuencia tendremos que usar cuando creemos páginas web:

- `<p></p>` : representa un párrafo ([+info](#)).
- `
` : representa un salto de línea ([+info](#)).
- `<h1></h1>` : esta etiqueta se utiliza para representar el encabezado de una página, como si fuera el índice de un libro. Puede variar desde 1 hasta 6 para diferenciar subniveles ([+info](#)).
- `` : representa una lista de elementos, donde el orden de los elementos no es importante - esto quiere decir que el cambio del orden no modifica el significado. ([+info](#)).
- `` : representa una lista de elementos, donde el orden de los elementos sí es importante - esto quiere decir que el cambio del orden modifica el significado. ([+info](#)).
- `` : representa un elemento de la lista y su padre siempre tiene que ser una etiqueta *ol* o *ul*. ([+info](#)).
- `` : representa algo muy importante, serio (para avisos o precauciones) o urgente (que debe ser leído antes). ([+info](#)).
- `` : sirve para enfatizar en el contenido. ([+info](#)).
- `<!-- -->` : se utiliza para añadir comentarios dentro del código que el usuario no podrá ver. Por ejemplo para añadir notas de tareas pendientes, aclaraciones que nos ayuden a nosotros o a otras personas a entender el código, etc. ([+info](#)).

Puedes consultar los ejemplos en la [lección 2 - Snippet 1-5](#)).

Truco: Para que recuerdes mejor qué significa cada elemento, las etiquetas piensa en los acrónimos en inglés:

- h1 = **h**eading**1**; h2 = **h**eading **2**; ...
- p = **p**aragraph
- br = **b**reak line
- ul = **u**nordered list
- ol = **o**rdered list
- li = **l**ist **i**tem
- em = **e**mphasis

El siguiente ejemplo muestra una página web que combina todas ellas:

```
<!DOCTYPE html>
<!-- TODO: añadir la etiqueta lang -->
<html>
<head>
  <meta charset="UTF-8">
  <title>Ejemplo con etiquetas básicas</title>
</head>
<body>
  <h1>Etiquetas HTML</h1>
  <p>
    Este ejemplo muestra cómo combinar algunas de las etiquetas más básicas de HTML.
  <br>
    Recuerda que <strong>es importante entender la diferencias entre ellas</strong>
  .
  </p>

  <h2>Etiqueta ul+li</h2>
  <p>
    Si listamos personas nominadas a los Oscars, dado que el orden no altera el significado, debemos usar <em>ul</em>.
  </p>
  <ul>
    <li>David Verdaguer</li>
    <li>Jesús Castro</li>
    <li>Israel Elejalde</li>
    <li>Dani Rovira</li>
  </ul>

  <h2>Etiqueta ol+li</h2>
  <p>
    En el caso de que estemos listando elementos donde el orden es importante, como por ejemplo la clasificación de un mundial de fútbol, debemos usar <em>ol</em>.
  </p>
  <ol>
    <li>España</li>
    <li>Países Bajos</li>
    <li>Alemania</li>
    <li>Uruguay</li>
  </ol>
</body>
</html>
```

Esto generaría una página como la siguiente:

Etiquetas HTML

Este ejemplo muestra cómo combinar algunas de las etiquetas más básicas de HTML5. Recuerda que **es importante entender la diferencias entre ellas**.

Etiqueta `ul+li`

Si listamos personas nominadas a los Oscars, dado que el orden no altera el significado, debemos usar `ul`.

- David Verdaguer
- Jesús Castro
- Israel Elejalde
- Dani Rovira

Etiqueta `ol+li`

En el caso de que estemos listando elementos donde el orden es importante, como por ejemplo la clasificación de un mundial de fútbol, debemos usar `ol`.

1. España
2. Países Bajos
3. Alemania
4. Uruguay

Si quieres puedes ver el ejemplo en vivo aquí: [Lección 1 - Snippet 2](#)

Es importante destacar que aunque el navegador le añade un estilo (CSS) por defecto a las etiquetas, por ejemplo:

- `h1` y `h2` una fuente mayor y negrita
- `strong` en negrita
- `ul` y `ol` con un margen a la izquierda y un punto o número respectivamente
- `em` en cursiva

Esto no es responsabilidad del HTML, esto lo podremos personalizar en un futuro con CSS. Así que insisto, recuerda que HTML **sólo sirve para dotar de estructura y semántica al contenido**.

Este valor semántico es **muy importante** entre otras cosas para por ejemplo:

- Que los buscadores (que no son más que programas automatizados) puedan "entender" el contenido de nuestra página y así poder detectar de qué estamos hablando y qué es importante.
- Para que otras herramientas como por ejemplo los navegadores para invidentes (p.e.

[WebbIE](#)) que lo que hacen es leer el contenido al usuario u otros [navegadores basados en texto](#).

Ejercicio tipo test

[Ejercicio tipo test de autoevaluación - Lección 2](#)

Recuerda que puedes repetirlo tantas veces como quieras.

Dudas

Si hay algo que no te haya quedado claro puedes preguntar cualquier duda en los [issues del proyecto en Github](#).

Recursos

Aquí te dejo dos tipos de recursos. Los avanzados míralos sólo si ya tenías un conocimiento previo de programación web (XHTML, HTML4, etc) o... si no le tienes miedo a nada ;D:

- Básicos:
 - [Chuleta de etiquetas HTML5](#)
 - [Artículo: ¿Qué puede ocurrir si realizamos mal la anidación de etiquetas?](#)
- Avanzados:
 - [Organización sin ánimo de lucro responsable de gestionar los dominios a nivel mundial: ICANN - Wikipedia](#)
 - [The Web Hypertext Application Technology Working Group](#)

Chrome DevTools

En esta lección vamos a empezar a trabajar con las herramientas de Google Chrome para desarrolladores ([Chrome DevTools](#))¹.

Como en las lecciones anteriores, si crees que ya dominas esta materia puedes probar a hacer el [ejercicio tipo test de esta lección](#). Si sacas un 100% de aciertos puedes pasar a la siguiente, sino te recomiendo que no te la saltes.

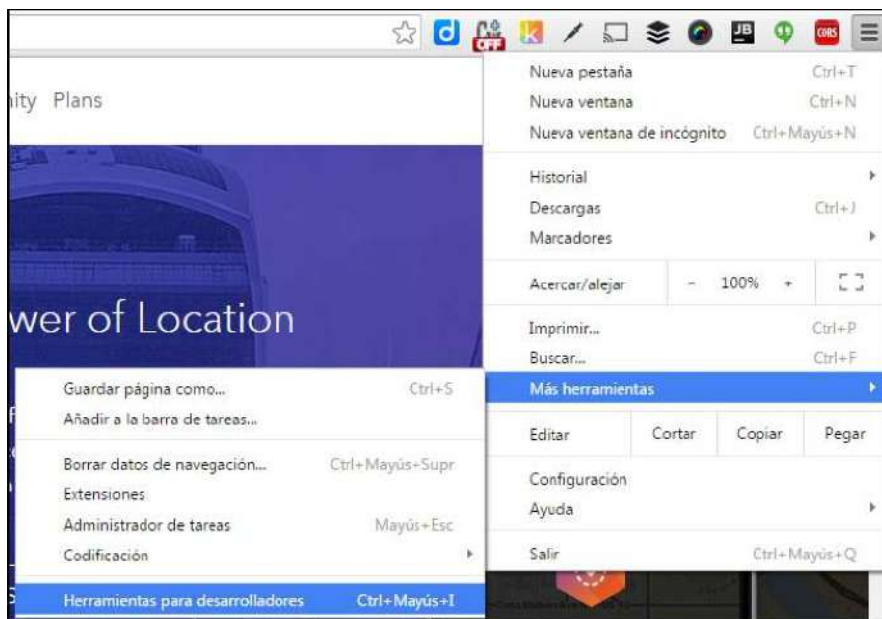
Pestañas

De momento sólo vamos a ver 3 grupos de herramientas que se encuentran organizadas por pestañas:

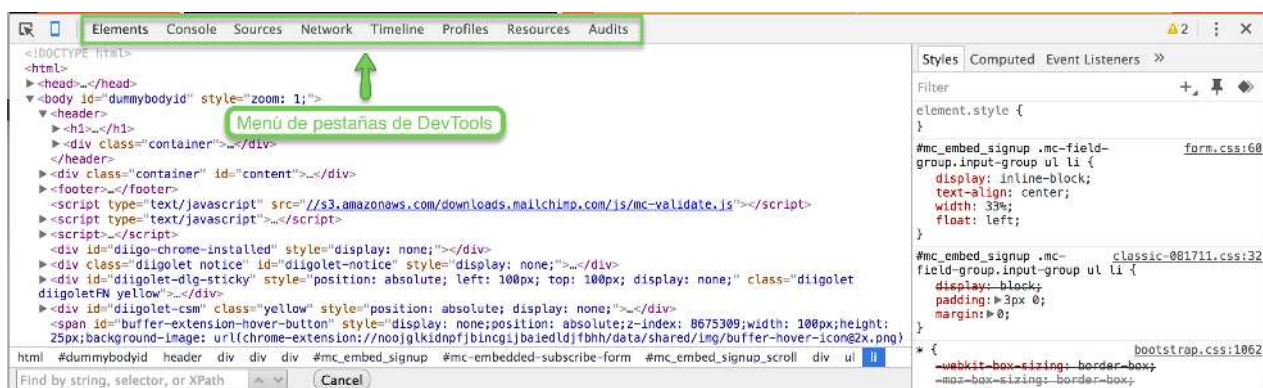
- **Red** (*Network*): esta pestaña nos permite ver los *recursos* que recupera nuestro navegador usando peticiones HTTP mientras cargamos y usamos la página.
- **Elementos** (*Elements*): nos permite ver y modificar el código² que representa la página que estamos viendo.
- **Fuentes** (*Sources*): nos permite navegar por todos los ficheros (HTML, CSS y JavaScript) que utiliza la página que estamos viendo.

La barra de herramientas la podemos abrir en cualquier página que estemos viendo. Para abrir esta barra podemos hacerlo mediante un atajo de teclado o mediante el menú de herramientas del Chrome:

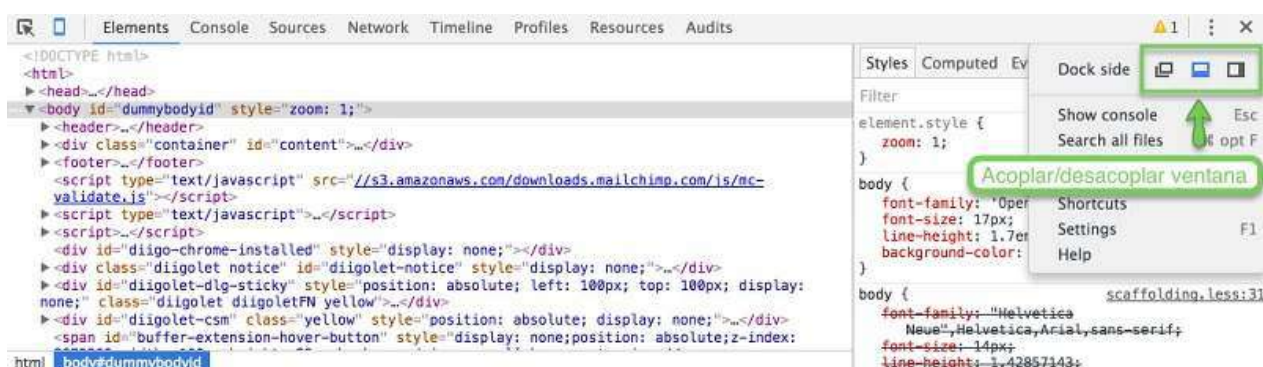
- **Atajo de teclado** (recomendado):
 - En Windows pulsando: F2 o Control + Shift + I
 - En Mac pulsando: Cmd + Opt + I
- **Pulsando el botón de menú**: "Botón de menú" -> "Más herramientas" -> "Herramientas para desarrolladores".
Como podemos ver en la siguiente imagen.



Una vez hecho esto nos aparecerá la barra de herramientas:



La barra podemos ajustarla a la derecha, abajo o desacoplarla en una nueva ventana como vemos a continuación:



Vayamos ahora analizando las pestañas.

Aclaraciones:

1. Puedes ampliar toda la información que veremos en este capítulo en la [página de documentación para desarrolladores de Google Chrome](#), en el curso [Discover DevTools](#) o

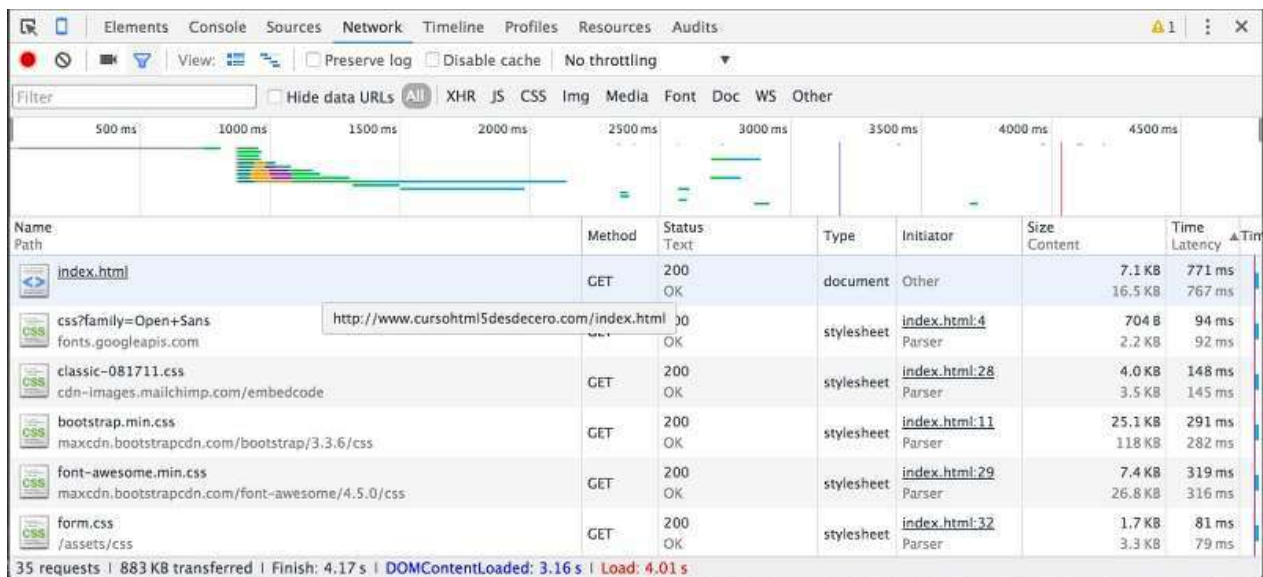
para los más avanzados en el curso [Website Performance Optimization](#)

2. El del DOM (que luego veremos que es) y el CSS, aunque de momento no nos entretendremos en esta parte.

Pestaña network

Como decíamos, esta pestaña nos permite ver los recursos que solicita el navegador de un servidor usando peticiones HTTP. También nos permite ver los detalles de las mismas: tipo del mensaje (GET/POST), código de respuesta (200, 404, ...), etc¹.

Si abres las DevTools después de haber cargado la página, la pestaña "Network" te saldrá vacía, si es así, refresca la página y verás como te aparece algo parecido² a esto:

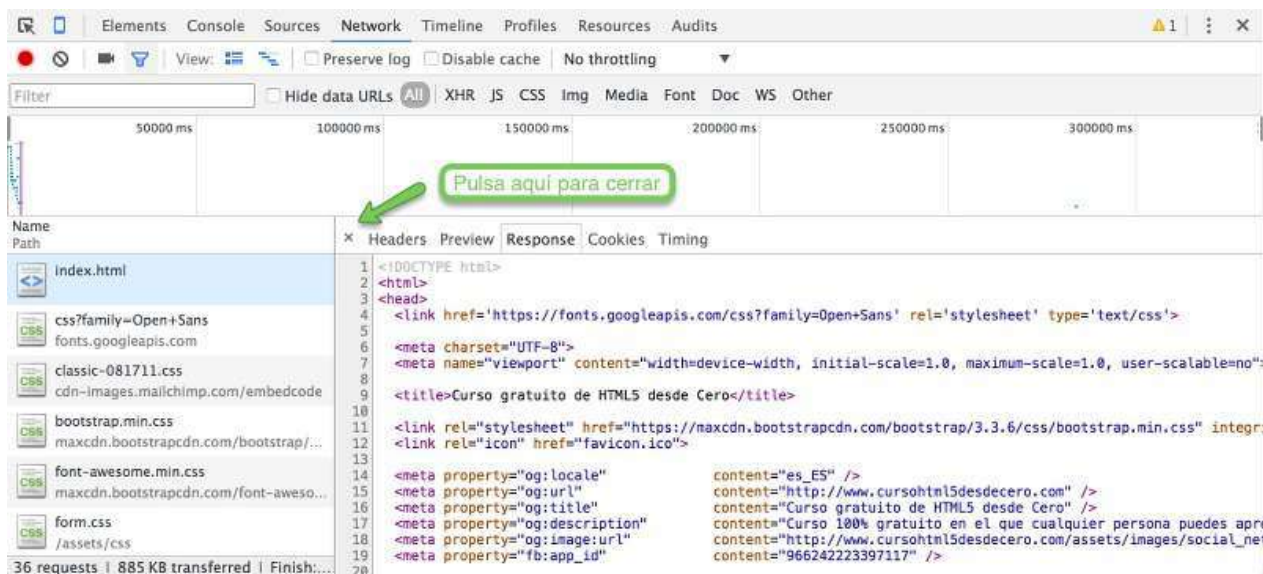


Cada fila representa una petición HTTP, y si te fijas, dejando el cursor encima de una petición te mostrará la URL completa del recurso y si pulsamos el botón derecho tendremos varias opciones, entre ellas la de abrir el recurso en una nueva ventana, eliminar todos los ficheros de la memoria caché, etc.

Nota:

Podemos vaciar la memoria caché³ de dos formas distintas, una es pulsando el icono de refrescar con el botón derecho y luego "Empty Cache and Hard Reload", y otra es pulsando el botón derecho sobre cualquiera de las peticiones HTTP y pulsar "Clear browser cache"

Si haces clic en cualquiera de las peticiones podrás ver los contenidos del recurso y algunos detalles que no nos preocuparán en este curso.



Para cerrar el detalle de la petición puedes pulsar en el aspa.

Además de filtrar las peticiones también puedes reordenarlas pulsando en el título de cada campo: **Name**, **Method**, **Status**, etc.

Veamos ahora **algunas**⁴ de las cosas que podemos hacer en esta pestaña. Si te fijas, las opciones en esta imagen no coinciden exactamente con las de la imagen anterior (y posiblemente tampoco con las tuyas), esto se debe a que este "pantallazo" es de una versión anterior del navegador (no importa), veamos que significan:



- **Preserve records on navigation**: por defecto aparece el botón en rojo, esto significa que cada vez que cambiemos de página se eliminarán las peticiones y se añadirán las nuevas. En cambio, si lo desactivamos se dejarán de registrar nuevas peticiones.
- **Preserve log**: si marcas esta opción, el efecto será justo el contrario, nunca se

borrarán las peticiones HTTP, ni cambiando de página ni de dominio (se irán añadiendo una tras otra).

- **Clear records:** este botón nos permite limpiar la información de las peticiones.
- **Filter:** nos permite filtrar las peticiones, se buscarán *URLs* que contengan el texto introducido.
- **Hide/show filter buttons:** para ocultar/mostrar los botones para filtrar.
- **Filter buttons:** estos botones nos permite ver sólo las peticiones HTTP que ha recuperado un tipo de recurso. *De momento* no lo usaremos mucho.
- **Summary view:** podemos ver cuántas peticiones HTTP se han necesitado para cargar la página (requests), cuando ocupa la suma de todos los recursos recuperados (XXX transferred), el tiempo exacto que ha tardado en descargar los recursos (ms = milisegundos), y en la siguiente lección veremos qué es el DOM y qué significa el DOMContentLoaded.
- **No throttling** (se ve en la imagen anterior): esto permite simular que tu conexión a Internet es más lenta⁵. Lo usaremos más adelante cuando queramos ver si nuestra página carga lo suficientemente rápido usando un dispositivo conectado por 3G.

Por último y antes de pasar a la siguiente lección, te animo a que dediques un par de minutos a jugar con esta pestaña, y si te surge alguna duda... no olvides preguntarla en los [issues de Github](#).

Aclaraciones:

1. Es la página de [desarrolladores de Chrome](#) puedes consultar que significa cada columna, aunque *no es trivial y no te recomiendo que lo hagas si estás empezando*.
2. Las peticiones serán distintas en cada página, por lo tanto con casi total seguridad tus peticiones serán distintas a las de la imagen.
3. Todos los navegadores incluyen una memoria caché temporal para optimizar el tiempo de carga de la página, de este modo el navegador puede reducir el número de peticiones HTTP ([más info](#)).
4. Si ya tienes experiencia con Chrome DevTools y quieres, encontrarás dónde ampliar conocimientos en la sección recursos de esta lección.
5. El throttling no funciona cuando estamos cargando un fichero sin utilizar un servidor web.

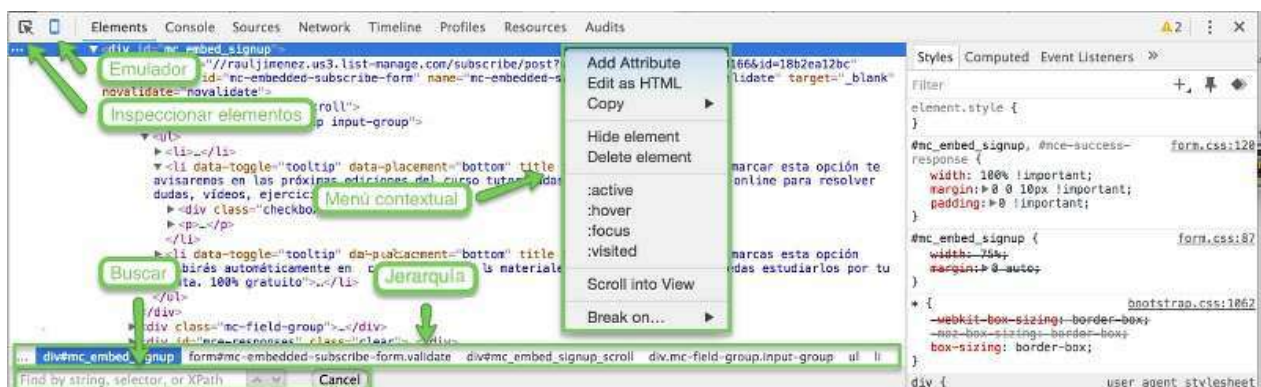
Pestaña elements

La pestaña "**Elements**" representa lo que llamaremos el "**DOM**"¹ (*Document Object Model*), que no es más que lo que representa la página que estamos viendo.

El DOM lo construye el navegador a partir del código HTML que recibe tras hacer la petición HTTP inicial. Además, el navegador intenta arreglar cualquier error que se encuentre en el código, por ejemplo: si se nos olvida cerrar una etiqueta, si anidamos etiquetas que no son anidables, etc. Por ese motivo y porque el DOM se puede modificar², este se parece pero no suele ser exactamente igual al código HTML recibido en la petición HTTP.

Además de poder ver el DOM, podemos editarlo, buscar texto en él y hasta reordenar las etiquetas arrastrándolas con el ratón.

En la siguiente imagen vemos las diferentes partes de esta pestaña:



- **Emulador:** esta opción nos permitirá simular que estamos usando un móvil o tablet y hacer *throttling* (simular otro tipo de conexión) al igual que hemos visto en la pestaña "**Network**".
- **Inspeccionar elementos:** activando esta opción podrás hacer clic sobre cualquier parte de la página y el inspector señalará en el DOM el código que representa el elemento seleccionado.
- **Menú contextual:** pulsando el botón derecho sobre el código veremos varias opciones:
 - **Add attribute:** permite añadir un atributo, por ejemplo: `charset="UTF-8"` (atajo de teclado: *Enter*).
 - **Edit as HTML:** nos permite añadir, editar o quitar cualquier cosa (atajo de teclado: F2 tanto en Windows como en Mac)
 - **Copy:** nos permite copiar el código (*outerHTML*), el selector (ya veremos lo que significa cuando veamos CSS), el **XPath**³ (es un lenguaje que nos permite buscar y seleccionar elementos teniendo en cuenta la estructura jerárquica del código), cortar y copiar el elemento.
 - **Ocultar elemento:** cambia la visibilidad a "no visible" usando CSS.

- **Delete element**: permite eliminar el elemento (atajo de teclado: *Suprimir* o *Borrar*).
- **:active, :hover, :focus, :visited**: nos permite cambiar el estado del elemento (esto lo usaremos en el apartado de CSS)
- **Scroll into View...**: en caso de ser necesario se hace scroll hasta que se muestre el elemento.
- **Break on...**: nos permite establecer puntos de parada indicando que se debe detener la ejecución de cualquier código JavaScript si:
 - Se modifica alguno de sus hijos.
 - Se modifica algún atributo.
 - O si se elimina el código.
- **Buscar**: Nos permite buscar cualquier palabra dentro del DOM (atajo de teclado: *Ctrl + F* en Windows ó *Cmd + F* en Mac).
- **Jerarquía**: nos muestra todos los ancestros del elemento y nos permite seleccionarlos.

Los cambios que hagas sobre esta pestaña no se guardarán ya que no estamos modificando el fichero de código⁴ sino el DOM (y ya hemos visto la diferencia), por tanto al refrescar la página todos los cambios desaparecerán.

Para mejorar tu productividad te recomiendo que de vez en cuando consultes [los atajos de teclado del panel Elements](#), como por ejemplo:

Windows/Linux	Mac	Función
Ctrl + Z	Cmd+Z	Deshacer los cambios realizados
Ctrl + Shift + C	Cmd + Shift + C	Abrir DevTools con la herramienta para " Inspeccionar elementos " activada por defecto
←, →, ↑, ↓	←, →, ↑, ↓	Navegar por los elementos del DOM

El panel que sale a la derecha es el del código CSS que se le ha aplicado al elemento seleccionado, pero esto ya lo veremos más adelante.

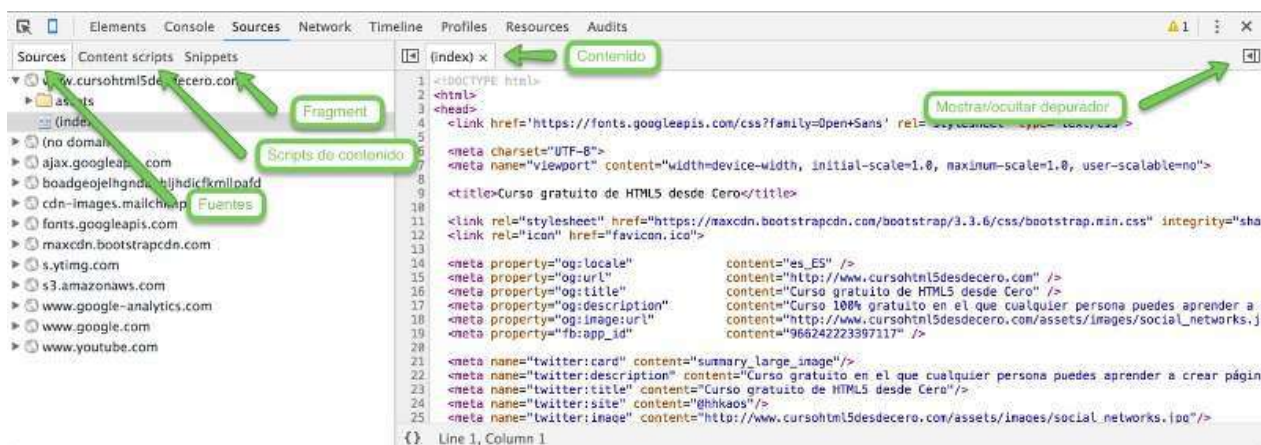
Aclaraciones:

1. Puedes encontrar la definición formal del DOM en la [página del W3C](#).
2. Usando JavaScript, o mediante una extensión del navegador por ejemplo
3. Puedes encontrar la definición formal de XPATH en la [página del W3C](#)
4. A veces escucharás "[código fuente](#)" en lugar de código, son sinónimos.

Pestaña sources

La pestaña "**Sources**" nos muestra las fuentes de contenido que se han utilizado para construir la página. Desde esta pestaña podemos escribir y modificar ficheros que estén vinculados a nuestro disco duro, pero veremos cómo hacer esto en la siguiente lección.

Empecemos por describir los distintos paneles:



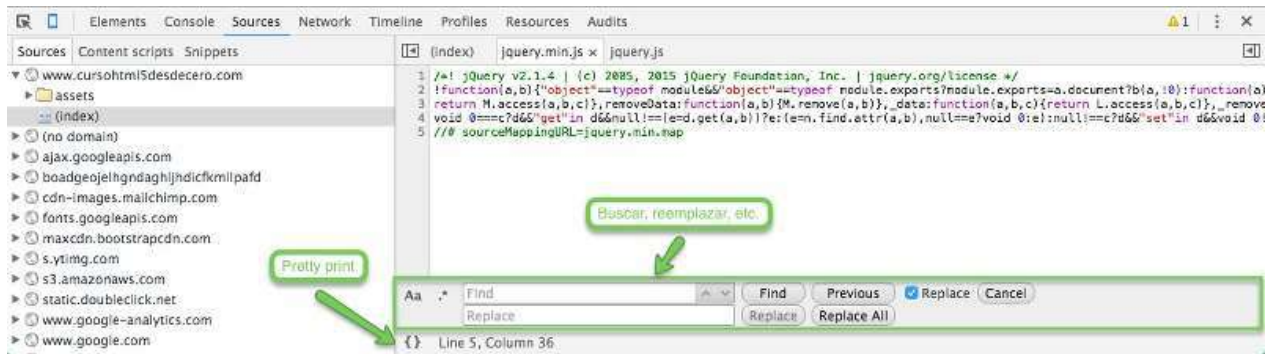
- **Sources**: aquí encontraremos por cada dominio desde el cual nuestro navegador haya obtenido recursos¹ (HTML, CSS o JavaScript) una jerarquía de ficheros. Haciendo clic en cualquiera de ellos se abrirá el código en un panel derecho.
- **Content scripts**: aquí se encuentran ficheros JavaScript simples (**scripts**) implementados desde las extensiones de Google Chrome.
- **Snippets**: esta pestaña nos permite almacenar *pequeños trozos de código*² *reutilizables* (**snippets**) escritos en JavaScript que podremos ejecutar o reutilizar (valga la redundancia) en cualquier página.
- **Depurador**: este panel nos ayudará a hacer un seguimiento paso a paso de la ejecución de nuestro código JavaScript para encontrar errores, veremos como usarlo en los capítulos de JavaScript.

Al igual que en la lección anterior, te recomiendo que guardes en un lugar seguro los **atajos de teclado del panel Sources** y de vez en cuando los revises para aumentar tu productividad.

Panel de contenido

El panel de contenido nos ofrece un **editor de código** que dispone adicionalmente de **otros atajos de teclado**.

Es importante saber que: a diferencia de los cambios del DOM, en la pestaña "**Elements**" para poder ver los cambios reflejados en la página que estamos viendo es necesario **guardar los cambios y refrescar la página**.



En cuanto a los atajos me gustaría destacar cinco que usaremos con **mucha frecuencia**:

Windows/Linux	Mac	Función
Ctrl + F	Cmd + F	Buscar (y adicionalmente reemplazar) texto dentro de un fichero
Ctrl + S	Cmd + S	Guardar un fichero
Ctrl + R, F5	Cmd + R	Refrescar la página
Ctrl + P	Cmd + P	Buscar ficheros por nombre
Ctrl + P + :num	Cmd + P + :num	Acceder directamente a un número de línea

Por último, la opción "**Pretty print**" veremos que es especialmente útil cuando estemos depurando [bibliotecas JavaScript minificadas](#) (*comprimidas*), aunque de momento no te preocupes por esto.

Recursos y aclaraciones:

1. Normalmente mediante peticiones HTTP aunque puede que también mediante las extensiones de Chrome.
2. En este repositorio de Github podrás encontrar una [colección de snippets](#).

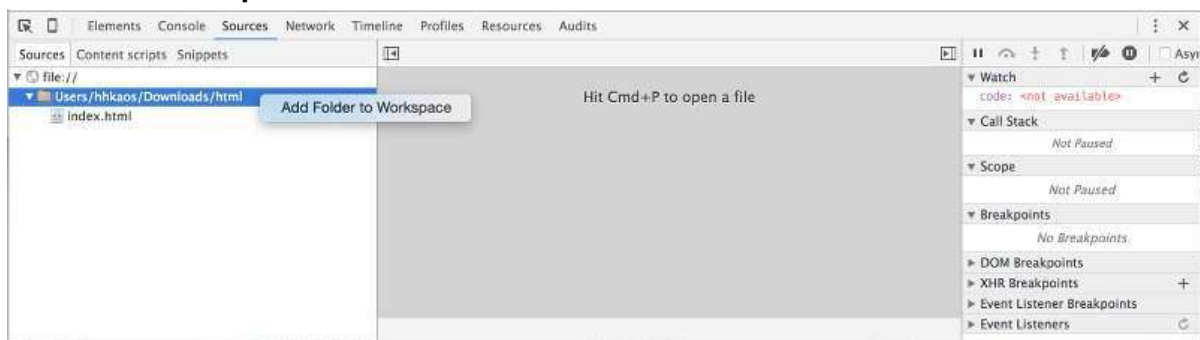
Configuración

Ahora vamos a ver cómo podemos configurar Google Chrome para poder modificar ficheros que se encuentren en nuestro disco duro.

Para hacer esto usaremos los **"Workspaces"**, estos nos permitirán realizar cambios **persistentes** en nuestro código sin tener que ejecutar otro editor de código.

Para ello vamos a seguir los siguientes pasos:

1. Creamos una carpeta (de prueba) en nuestro disco duro (por ejemplo **"html"**).
2. Creamos un fichero vacío dentro de la carpeta llamado: **"index.html"**¹
3. Abrimos el fichero con Google Chrome
4. Abrimos las DevTools y nos vamos a la pestaña **"Sources"**
5. Hacemos clic en el panel izquierdo sobre la ruta del directorio y seleccionamos **"Add Folder to Workspace"**:



6. Y por último pulsamos **"Allow"** para autorizar a DevTools a realizar cambios persistentes en el disco duro:



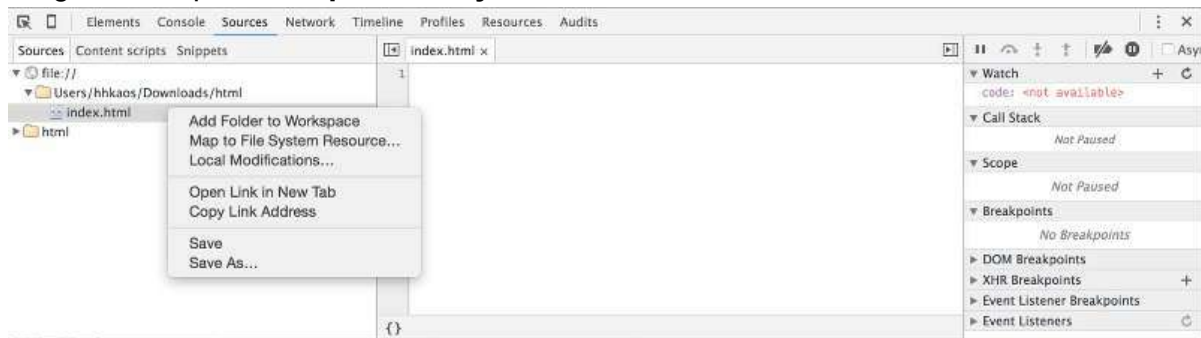
Nota:

Si nos equivocamos al añadir un directorio al Workspace podremos eliminarlo simplemente pulsando con el botón derecho sobre el directorio y seleccionando la opción **"Remove Folder from Workspace"**.

Ahora tenemos que "mapear" (vincular) el recurso que ha obtenido el navegador con el fichero del disco duro que queremos modificar (osea, con él mismo), para ello:

1. Hacemos clic con el botón derecho sobre el nombre del fichero (*index.html* que cuelga de "file:///")

2. Elegimos la opción "Map to File System Resource":



3. Seleccionamos el fichero dentro del espacio de trabajo (Workspace).

4. Y refrescamos la página.

Y ya estamos listos para empezar a programar usando las Chrome DevTools.

También puedes consultar las [limitaciones](#) de los "Workspaces", pero no te preocupes por ellas ya que no nos afectarán en este curso; por ejemplo, los cambios en la pestaña "Elements" no serán persistentes (lógico ya que lo que estamos cambiando es el *DOM*, que como vimos anteriormente es dinámico, osea que va cambiando).

Gestión de ficheros

Una vez hecho todo esto podemos añadir y eliminar ficheros directamente desde DevTools:

1. Añadir ficheros: pulsando con el botón derecho sobre la carpeta y seleccionando **"New File"**.
2. Eliminar ficheros: pulsando con el botón derecho y seleccionando **"Delete"**

Sin embargo no podemos crear y eliminar directorios, esto lo tendremos que hacer directamente desde la carpeta que hemos creado en nuestro disco duro.

Aclaraciones:

1. A pesar de que no vamos a usar aún un servidor web, lo llamaremos así para ir acostumbrándonos a este nombre. Por defecto los servidores web cuando reciben la petición de un recurso y no se indica explícitamente el nombre del recurso, busca en la carpeta un fichero con nombre "index.html", y lo devuelve en caso de encontrarlo.

Ejercicio

Ejercicio tipo test de autoevaluación - Lección 3

Una vez más, recuerda que puedes repetirlo tantas veces como quieras.

El ejercicio práctico es muy simple, tan sólo consiste en añadir el código que hemos visto en el capítulo "**HTML5: Primeros pasos > Etiquetas básicas**" (lo puedes encontrar al final de esta lección) al fichero *index.html* que hemos creado.

Luego quiero que te familiarices un poco con las pestañas que hemos visto e intentes:

- Usar los atajos de teclado en la pestaña "**Sources**".
- Buscar la petición HTTP que se envía al cargar la página en "**Network**".
- Editar el código en la pestaña "**Elements**":
 - Modifica el contenido y el HTML de la etiqueta **h1**
 - Elimina una etiqueta
 - Reordena otra etiqueta
 - Haz una búsqueda y encuentra por ejemplo: **Rovira**
 - Haz clic en la jerarquía para acceder al elemento "**ul**".
 - Ejecuta el emulador y activa la opción de **Apple iPhone 5**.

```
<!DOCTYPE html>
<!-- TODO: añadir la etiqueta lang -->
<html>
<head>
  <meta charset="UTF-8">
  <title>Ejemplo con etiquetas básicas</title>
</head>
<body>
  <h1>Etiquetas HTML</h1>
  <p>
    Este ejemplo muestra cómo combinar algunas de las etiquetas más básicas de HTML.
  <br>
    Recuerda que <strong>es importante entender la diferencias entre ellas</strong>
  </p>

  <h2>Etiqueta ul+li</h2>
  <p>
    Si listamos personas nominadas a los Oscars, dado que el orden no altera el significado, debemos usar <em>ul</em>.
  </p>
  <ul>
    <li>David Verdaguer</li>
    <li>Jesús Castro</li>
    <li>Israel Elejalde</li>
    <li>Dani Rovira</li>
  </ul>

  <h2>Etiqueta ol+li</h2>
  <p>
    En el caso de que estemos listando elementos donde el orden es importante, como por ejemplo la clasificación de un mundial de fútbol, debemos usar <em>ol</em>.
  </p>
  <ol>
    <li>España</li>
    <li>Países Bajos</li>
    <li>Alemania</li>
    <li>Uruguay</li>
  </ol>
</body>
</html>
```

Dudas

Si hay algo que no te haya quedado claro puedes preguntar cualquier duda en los [issues del proyecto en Github](#).

Recursos

Definiciones que hemos visto:

- [Definición de DOM por el W3C](#)
- [Definición de XPath por el W3C](#)

Recursos sobre Chrome y las DevTools:

- [Documentación para desarrolladores de Google Chrome](#)
- [Chrome DevTools Overview](#)
- [Atajo de teclado de Chrome DevTools](#)
- [Chuleta de atajos de teclado en Chrome](#)
- [Funcionamiento del cacheado](#)
- [Configuración del Workspace en Chrome DevTools](#)
- [Limitaciones del Workspace de Chrome DevTools](#)
- [Content scripts en Chrome DevTools](#)
- [Snippets en Chrome DevTools](#)
- [Colección de snippets para Chrome DevTools](#)

*Si es tu primera vez con DevTools **no te lo recomiendo**, pero si quieres, puedes encontrar más información sobre cómo sacarle más provecho a Google Chrome en los siguientes cursos:*

- [Discover DevTools - CodeSchool.com](#)
- [Website Performance Optimization - Udacity.com](#)
- [Browser rendering optimization - Udacity.com](#): aprende cómo funciona internamente Google Chrome y cómo optimizar el layout (**conocimientos en CSS requestidos**).

HTML5: Mi Curriculum Vitae

En este capítulo vamos a aprender todo lo necesario para crear el código HTML de una página con nuestro CV, y en el siguiente aprenderemos cómo podemos subir nuestro curriculum a un repositorio de Github y a ponerlo accesible para cualquier persona usando gh-pages.

Por tanto al terminar este apartado tendremos que haber creado una página HTML [similar a esta](#):

- Experiencia profesional
- Habilidades y conocimientos
- Educación
- Perfiles sociales
- Contactar

Raúl Jiménez Ortega

Me apasionan temas como son: los negocios en Internet, la analítica web, la optimización web (A/B Testing, UX, ...), el SEO, los móviles y el SoLoMo, el emprendimiento y las startups, las redes sociales aplicadas a los negocios, las tecnologías web, etc.

Pero creo que es mejor que me conozcas en mi vídeo de presentación.

Experiencia profesional

Empresa / Organización	Cargo	Localización	Desde-hasta
Erii España	Head of Developers and Startup	Madrid	Marzo 2014 - Actualidad
Upplia.com	Product manager y desarrollador frontend	Madrid	Noviembre 2013 - Marzo 2015
Universidad Europea de Madrid	Profesor del MOOC: Innovación y emprendimiento	Madrid	Noviembre 2013 - Marzo 2015
Pluqa	Product manager y frontend developer	Madrid	Diciembre 2012 - Julio 2013
Google Developers Group	Coordinador nacional	Granada	Septiembre 2012 - Enero 2013
Imagen Marketing	Socio y asesor	Granada	Mayo 2012 - Diciembre 2012
Pádelo.com	Fundador y CEO	Granada	Noviembre 2011 - Diciembre 2012
GeoRefinedMe	Fundador y CEO	Granada	Octubre 2010 - Marzo 2012
Asociación de Webmasters de Granada	Presidente	Granada	Marzo 2009 - Marzo 2011

Habilidades y conocimientos

- Desarrollo web:** HTML5, CSS3, JavaScript, Python-Django, LEMP & SASS, PHP, SEO, Node.js, MySQL, AngularJS, jQuery, Bootstrap, Dojo, responsive design, WordPress, usabilidad, CoffeeScript, etc.
- Analítica web/Métricas:** Google Analytics, Mixpanel, segmentin, etc.
- Gestión:** SCRUM, gestión de comunidades y organización de eventos, gestión de equipos, etc.
- Emprendimiento:** startups, Lean Startup, etc.
- Aplicaciones móviles:** Phonegap
- Marketing:** email marketing, redes sociales, gamificación, native marketing, lead scoring, etc.
- Sistemas de información geográfica (GIS):** ArcGIS API

Educación

Título	Entidad	Promoción
Master of Business Administration (MBA)	IESE Escuela de Negocios	2012
Curso: creación de empresas	IESE Escuela de Negocios	2009
Psicología de la publicidad, marketing y consumo	Facultad de Psicología, Universidad de Granada	2009
Master en informática (Erasmus)	Universiteit van Amsterdam	2008
Master en informática (Erasmus)	Universitetet i Bergen, Noruega	2006
Ingeniería Superior en Informática	ETSIT, Universidad de Granada	2002

Perfiles sociales

También puedes encontrarme en:

- [RaúlJiménez.info](#)
- [Github](#)
- [Twitter](#)
- [LinkedIn](#)
- [Youtube](#)
- [Slides.com](#)
- [Dribbble](#)
- [SlideShare](#)
- [Google Plus](#)
- [Flickr](#)
- [StumbleUpon](#)
- [Facebook](#)
- [CodeSchool](#)
- [Google](#)



Contactar

Para:

Resuelve este CAPTCHA para saber mi email: [h...@gmail.com](#)

Temática: Asunto: Cuerpo: Enviar

Cualquier persona puede desarrollar esta página como esta a partir de los conocimientos adquiridos en el curso HTML5 desde cero.

W3C  

Así que vamos a empezar por ver los elementos HTML que nos faltan por aprender para poder llegar a hacerla.

Etiquetas - Parte 2

Ahora vamos a ver las etiquetas básicas para trabajar con: enlaces o hipervínculos, imágenes, tablas, formularios, separadores y otras consideraciones.

Esta vez tampoco veremos todos los atributos posibles aunque añadiré enlaces a la documentación:

Enlaces o hipervínculos

Una de las características más destacables de HTML es la posibilidad de enlazar unas páginas con otras, para hacer esto utilizamos el elemento "**a**" con el atributo "**href**" (Hypertext Reference). Por ejemplo:

```
<a href="http://www.cursohtml5desdecero.com">Curso de HTML5 desde cero</a>
```

Hay 3 tipos de enlaces:

- **Absoluto:** es un enlace que incluye todas las partes de una URL como vimos en el capítulo 1:

```
scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]
```

- **Relativo:** hace referencia a un recurso que se encuentra en una posición relativa a nuestra URL, así podemos establecer rutas relativas, por ejemplo:

```
<a href="img/imagen1.jpg">enlace a una imagen</a>
```

Donde indicamos que si por ejemplo la URL actual es

<http://www.cursohtml5desdecero.com/leccion1>, la imagen se encuentra en

<http://www.cursohtml5desdecero.com/leccion1/img/imagen1.jpg>, y si queremos hacer referencia a un recurso que se encuentra en un nivel superior del "**path**" lo hacemos usando "../", por ejemplo:

```
<a href="../img/imagen1.jpg">enlace a una imagen</a>
```

Que hará referencia a la siguiente URL:

<http://www.cursohtml5desdecero.com/img/imagen1.jpg> (eliminamos **leccion1**).

- **Ancla (o anchor):** a diferencia de los dos anteriores, este enlace se utilizar para indicar un elemento dentro de la misma página que estamos viendo. Para ello tenemos que explicar un nuevo tipo de atributo que tienen todos los elementos en HTML, el **id**

(*unique identifier*), como su nombre indica es un identificador único y por tanto no podemos ponerle a dos elementos HTML el mismo **id**. Luego para añadir un hipervínculo a este elemento sólo tenemos que establecer el atributo **href** del enlace al **id** del elemento precedido de una almohadilla (#), por ejemplo:

```
<a href="#leccion1">Lección 3</a>
...
<!-- aquí vendría todo el contenido -->
<h2 id="leccion1">Lección 3</h2>
```

Esto estamos acostumbrado a verlo en la Wikipedia, por ejemplo:

https://en.wikipedia.org/wiki/Hyperlink#Hyperlinks_in_HTML

Ver: [Lección 2 - Snippet 6](#)

Imágenes

Para mostrar una imagen en una página tenemos dos formas de hacerlo, una es usando el elemento **img** y otras es mediante CSS (que veremos en el capítulo correspondiente).

Esta etiqueta sólo requiere de dos atributos obligatorios que son: **src** (de *source*) y el otro es **alt** (de *alternative*), por ejemplo:

```

```

Como podemos deducir del código anterior, el atributo **src** lo usaremos para indicar la URL (absoluta o relativa) a la imagen, y **alt** como el texto (alternativo) que mostrará el navegador en caso de no encontrar la imagen¹.

Ver: [Lección 2 - Snippet 7](#)

Tablas

Podemos crear tablas en HTML usando el elemento **table**². Para ello como mínimo tendremos que indicar las filas y las columnas usando los elementos **tr** (*table row*) y **td** (*table data*) respectivamente, así por ejemplo:

```
<table>
  <tr>
    <td>Fila 1, columna 1</td>
    <td>Fila 1, columna 2</td>
  </tr>
  <tr>
    <td>Fila 2, columna 1</td>
    <td>Fila 2, columna 2</td>
  </tr>
</table>
```

Que daría un resultado como el siguiente:

Fila 1, columna 1	Fila 1, columna 2
Fila 2, columna 1	Fila 2, columna 2

Como podemos comprobar las columnas (**td**) siempre van contenidas dentro de las filas (**tr**). En caso de querer agrupar celdas de una misma fila o columna lo haremos así:

- Misma fila: la usaremos el atributo **colspan** (*column span* = número celdas a abarcar)
- Agrupar dos celdas de una misma columna usaremos el atributo **rowspan** (*row span* = número de celdas a abarcar)

Por ejemplo:

```
<table>
  <tr>
    <td>Fila 1, columna 1</td>
    <td colspan="3"> Fila 1, columnas 2, 3 y 4</td>
  </tr>
  <tr>
    <td rowspan="2">Fila 2, columna 1 + Fila 3, columna 1</td>
    <td>Fila 2 columna 2</td>
    <td>Fila 2 columna 3</td>
    <td>Fila 2 columna 4</td>
  </tr>
  <tr>
    <td>Fila 3 columna 2</td>
    <td>Fila 3 columna 3</td>
    <td>Fila 3 columna 4</td>
  </tr>
</table>
```

Quedando algo como esto:

Fila 1, columna 1	Fila 1, columnas 2, 3 y 4		
Fila 2, columna 1 + Fila 3, columna 1	Fila 2 columna 2	Fila 2 columna 3	Fila 2 columna 4
	Fila 3 columna 2	Fila 3 columna 3	Fila 3 columna 4

Como podemos ver el atributo afecta a las celdas de las siguientes columnas/filas y el valor indica cuántas celdas debe abarcar.

Ver: [Lección 2 - Snippet 8](#)

Formularios

Vamos a hablar muy brevemente de los formularios, algunos de los elementos y de sus propiedades:

- **form**: será el elemento padre que anide todos los elementos HTML que representarán los campos de nuestro formulario, incluido el botón de enviar.
 - **action**: indica la URL a la que se enviará la petición HTTP con toda la información del formulario
 - **method**: indica si la petición HTTP será *GET* o *POST*
- **input**: permite introducir diferentes *tipos* de campo de formulario en base al valor del atributo **type**. En función del valor indicado en **type** dispondremos de unos atributos u otros (en total hay 30, pero no todos aplican a todos los casos):
 - **type** (obligatorio): este valor puede tener **muchos valores**: *text*, *email*, *checkbox*, *color*, *date*, *file*, *hidden*, etc. en función del tipo de campo que queramos, los nombres son bastante auto-explicativos.
 - **id**: este atributo es obligatorio si en el elemento **label** tiene un atributo **for**, en tal caso deberá contener un identificador único³ en la página.
 - **name**: este atributo es opcional y representa el nombre asignado al campo cuando se envíe la petición HTTP.
 - **value**: este valor es opcional pero representa el valor que se asignará al campo cuando se envíe la petición HTTP.
- **select**: nos permite crear una lista desplegable de opciones, cada opción estará contenida como hija dentro de un elemento **option**.
 - **id**: igual que el elemento **input**
 - **name**: igual que el campo **input**
- **option**: nos sirve para "encapsular" cada opción de la lista.
 - **value**: igual que el atributo **value** del campo **input**.
- **textarea**: representa un campo que nos permite introducir textos con saltos de línea incluidos, normalmente se usa cuando hay que introducir: descripciones, biografías,

etc.

- **id**: igual que el elemento **input** y **select**.
- **name**: igual que el campo **input** y **select**.
- **label**: se usa para especificar la etiqueta (o nombre) del campo del formulario.
 - **for**: tiene que tener el mismo valor que el atributo **id** del campo (input, select o textarea) al que hace referencia la etiqueta.
- **button**: representa un botón y el texto del botón está representado por su contenido.
 - **type**: define el comportamiento del botón cuando está activado y puede contener tres valores: *submit*, *reset*, *button*

Existen muchos otros atributos que no veremos dado que no les daremos uso en este curso ya que para sacarle el máximo provecho sería necesario tener conocimientos en programación.

Por último añadir que el elemento **input** no requiere una etiqueta de cierre (o lo que es lo mismo, que está autocontenido).

Aquí tenemos un ejemplo de formulario:

```
<form action="miScript.php" method="GET">
  <label for="to">Para:</label>
  <input id="to" type="email">

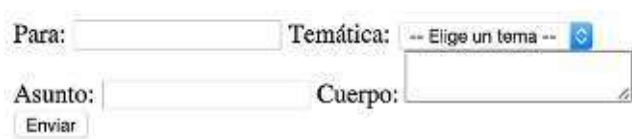
  <label for="topic">Temática: </label>
  <select id="topic" name="topic">
    <option>-- Elige un tema --</option>
    <option value="proposal">Propuesta</option>
    <option value="report">Reporte</option>
    <option value="other">Otro</option>
  </select>

  <label for="subject">Asunto: </label>
  <input id="subject" name="subject" type="text">

  <label for="body">Cuerpo:</label>
  <textarea id="body" name="body"></textarea>

  <button type="submit">Enviar</button>
</form>
```

Que nos dará como resultado algo así:



Como ves los estilos por defecto serán muy poco atractivos, pero no te preocupes, ya aprenderemos a solucionar esto usando CSS.

Por último comentar que *en muchos de los elementos*⁴ podemos añadir (opcionalmente) otros atributos como:

- **required** a un elemento para que el navegador se encargue de validar que este campo está relleno
- **readonly** si queremos que sea de sólo lectura
- **placeholder** si queremos que aparezca un texto de ayuda para rellenar el campo
- **value** para introducir un valor por defecto en el campo

Por ejemplo:

```
<label for="to">Para:</label>
<input id="to" type="email" placeholder="tu@correo.com" required>

<label for="subject">Asunto: </label>
<input id="subject" type="text" value="Formulario de contacto" readonly>
```

Ver: [Lección 2 - Snippet 9](#)

Separadores

Existe un elemento que nos permite añadir un separador (una línea horizontal), este elemento es **hr**.

Ver: [Lección 2 - Snippet 10](#)

Otras consideraciones

Para terminar este capítulo hay una última cosa que me gustaría comentar:

- En HTML se ignoran todos los espacios a partir del primero, por lo tanto nunca podremos (*ni se debe*) alinear usando espacios.
- Las entidades HTML (*HTML entities*) se usan para pintar palabras, caracteres o símbolos reservados o que puede que no tengas en tu teclado como por ejemplo: <, >, ©, &, €, etc.
Existen [1446 entidades](#) reservadas que puedes consultar en la página del W3C. (ver: [Lección 2 - Snippet 11](#))

Para representar la entidades HTML se usa el siguiente formato:

```
&código_de_la_entidad;
```

Veamos un ejemplo para entender mejor por qué existen las entidades HTML y cómo se usan.

Imaginemos que estamos escribiendo un libro como este y necesitamos hablar sobre la etiqueta `<hr>` :

```
<p>La etiqueta <hr> se utiliza para ...</p>
```

En este caso cuando el navegador esté interpretando el código HTML encontrará "**<hr>**" y en lugar de mostrar la cadena de texto (que es lo que queremos) nos mostrará un separador horizontal.

Para evitar este problema usaríamos el siguiente código HTML:

```
<p>La etiqueta &lt;hr> se utiliza para ...</p>
```

En este caso hemos modificado el símbolo "menor que" (**Lower Than**) por la entidad HTML **lt** y "mayor que" (**Greater Than**) por **gt**, así el navegador podrá representarlo sin ningún problema.

En alguna ocasión puede que navegues por una página con una codificación (encoding) que no soporte los acentos agudos (á, é, í, ó, ú), en ese caso usarán las entidades HTML (`´`, `é`, `í`, `ó`, `Ú`) para representarlos. Por cierto: "acute" en inglés significa "agudo".

Aclaraciones:

1. Puede que no se encuentre la imagen porque alguien la borre del servidor o porque nos equivoquemos al introducir la URL.
2. Las tablas sólo deben usarse para mostrar datos que tengan sentido en una tabla y nunca para maquetar.
3. Con esto nos referimos a un nombre (o cadena de texto) que no contenga ningún otro elemento, por ejemplo no puede haber dos elementos con **id="email"**.
4. En la documentación del W3C podemos ver qué atributos admite cada elemento: [input](#), [textarea](#), [select](#), etc.

Anidación - Parte 2

En el primer capítulo de HTML vimos que las etiquetas se pueden anidar, pero comentamos que no todas las etiquetas son anidables entre sí, por ejemplo **esto sería incorrecto**:

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

Ya que una etiqueta **body** no puede contener a otra **head**, pero... ¿cómo podemos saber qué etiquetas son anidables?.

Con la práctica aprenderá algunas anidaciones que están prohibidas y desarrollará una capacidad de razonar algunas anidaciones obvias, pero al principio podrás servirte de tres recursos principalmente:

- La pestaña **Elements** del navegador. Como decíamos el navegador es un programa bastante complejo, y entre otras cosas que se encarga es de construir el DOM. Si durante el proceso de construcción del DOM el navegador se encuentra una anidación incorrecta intentará resolverla. Por eso si inspeccionamos el DOM de nuestra página, podremos ver si el propio navegador ha encontrado etiquetas mal anidadas y nos ha modificado el código.
- El validador de código del W3C que veremos cómo usar en el siguiente apartado.
- Pero nuestro principal recurso debe ser [la especificación de HTML5 del W3C](#), ahora veremos cómo usarla.

Usar la especificación del W3C

En la descripción de todo elemento HTML nos encontraremos un apartado que se llama **Content model** que especifica qué tipo de etiquetas se pueden anidar, por ejemplo: **Metadata content**, **Flow content**, **Sectioning content**, **Heading content**, **Phrasing content**, ...

Por ejemplo el elemento **"li"** que usábamos para especificar un elemento de una lista (List Item) indica que su "Content model" es **"Flow content"**, si hacemos clic en el enlace verás que esto significa que el elemento soporta casi todos los elementos: a, area, article, b, blockquote, br, div, em, footer, form, h1, h2, h3, h4, h5, h6, header, hr, i, iframe, img, input, etc.

Sin embargo si vas a la especificación del elemento "**p**" verás que su modelo es de "[Phrasing content](#)", que admite muchas etiquetas pero por ejemplo no admite: "ul", "ol", "hr", etc.

Esta es la mejor forma de saber qué etiquetas son anidables y cuales no.

Validación

Que el código se muestre en nuestro navegador web como queríamos no implica necesariamente que lo hayamos escrito bien. En muchas ocasiones el navegador es capaz de detectar errores humanos y corregirlos de manera automática para que el usuario vea bien la página, pero esto no es siempre así. Si queremos asegurarnos de que nuestra página está correctamente escrita podemos usar [el Validador de HTML del W3C](#), que además en caso de encontrar errores nos dará pistas sobre cómo resolverlos.

Si abres el enlace verás que tienes 3 formas de validar código:

1. **Validate URI:** que te permite introducir la URL de una página para comprobar su código. Como nosotros aún no hemos alojado nuestra página en ningún servidor web no usaremos esta opción (aún).
2. **Validate by File Upload:** que nos permite subir un fichero HTML
3. **Validate by Direct Input:** que nos permite introducir el código dentro de un elemento **textarea**.

Usaremos las opciones 2 y 3 hasta que en la siguiente lección aprendamos cómo alojar nuestra página en un servidor web accesible desde Internet.

Aunque el navegador sea capaz de solucionar algunos de nuestros errores, es importante crear código válido porque:

- Aunque el navegador sea capaz de resolver un problema, no todos los navegadores son iguales, y por tanto puede que algunos no lo resuelvan o la solución que aplique provoque un efecto no deseado.
- Hace tu página más accesible como veremos a continuación.

Accesibilidad

Vamos a ver este apartado muy por encima, pero no quería dejarlo completamente de lado.

Existe una iniciativa que tiene como objetivo hacer la web más accesible, especialmente para personas cualquier tipo de discapacidad: visual (completa o parcial), auditiva, cognitiva, etc. Esta iniciativa se llama: **Web Accessibility Initiative - Accessible Rich Internet Applications (WAI-ARIA)** y están creando un estándar que [actualmente se encuentra en la versión 1.1](#).

Cualquiera puede aplicar algunas prácticas de accesibilidad con poco esfuerzo, como por ejemplo usar los [landmarks](#), que no son más que atributos que añadimos a las etiquetas para indicar las partes más relevantes de nuestra página como: el menú de navegación, el área de contenido principal o contenido complementario¹.

Si hablas inglés te recomiendo ver [esta charla de Leonie Watson](#), una mujer ciega explicando la importancia de la accesibilidad.

Por último, si estos argumentos no son suficientes, me gustaría añadir que haciendo una página accesible hacemos que esta esté mejor posicionada por los buscadores, dado que a las arañas² de los motores de búsqueda a fin de cuentas tienen una forma de navegar³ por nuestra página igual que las personas con problemas de accesibilidad.

Aclaraciones:

1. Pequeño vídeo explicativo en inglés sobre [cómo usar los landmarks](#).
2. Las "[arañas](#)" ([spiders](#)), [bots](#) o [web crawlers](#), son es el nombre convencional que le damos a los programas que se dedican a "rastrear" por Internet y que usan entre otros los grandes buscadores para buscar nuevo contenido y cualificarlo para después devolverlo en sus resultados de búsqueda.
3. Sitio web explicando [cómo funciona Googlebot](#) (la araña de Google).

Convenciones

Antes de terminar me gustaría explicarte algunas de las principales convenciones o buenas prácticas que deberemos de tener en cuenta a la hora de escribir código HTML:

- Los nombres de los elementos HTML y sus atributos se deben escribir en minúsculas

```
<!-- MAL -->
<p>
  <IMG SRC="html5.gif" ALT="Logo HTML5">
</p>
<!-- BIEN -->
<p>
  
</p>
```

- Los valores de los atributos en HTML deben ir entre comillas dobles:

```
<!-- MAL -->
<img src='html5.gif' alt='Logo HTML5'>
<!-- BIEN -->

```

- La indentación se debe hacer con 2 espacios (prácticamente todos los editores de código permite configurar este valor).

```
<p>
  
</p>
```

- No introducir espacios antes o después del signo "igual":

```
<!-- MAL -->
<img src = "html5.gif" alt = "Logo HTML5">
<!-- BIEN -->

```

- Usar UTF-8 como encoding.
- No cerrar elementos autocontenidos, por ejemplo usa `
` en lugar de `
`
- Evita el uso de estilos en línea (atributo `style` lo veremos en el siguiente apartado)
- Evita el uso de entidades HTML siempre que sea posible (salvo por ejemplo para `<` y `&`)

```
<!-- MAL -->
<h1>P&acute;gina sobre &lt; HTML5 &amp; CSS3</h1>
<!-- BIEN -->
<h1>Página sobre &lt; HTML5 &amp; CSS3</h1>
```

- Especifica el atributo **lang** en el elemento **html**:

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

- Especifica siempre el atributo **for** cuando añadas un elemento **label**

```
<label for="field-email">email</label>
<input type="email" id="field-email" name="email" value="" />
```

- Internet Explore soporta el uso de una etiqueta de compatibilidad **meta** indicando cómo tratar el código, usar siempre que se pueda:

```
<meta http-equiv="X-UA-Compatible" content="IE=Edge">
```

Esta recopilación ha sido extraída de algunas guías de estilo.

- [HTML\(5\) Style Guide and Coding Conventions](#)
- [HTML coding standards - CKAN](#)
- [Google HTML/CSS Style Guide](#)
- [Code Guide by @mdo](#)

Otras convenciones

Nombres de ficheros

Es recomendable seguir las siguientes convenciones:

- Establecer los nombres de los ficheros en minúsculas, Windows no hace distinción entre mayúsculas y minúsculas pero otros sistemas sí, y esto puede provocar que una ruta funcione en un sistema operativo pero no en otro. Por ejemplo si tenemos un fichero llamado **Logo_HTML5.jpg** y una página que hace referencia a él con `` . Esto funcionará en Windows pero en un sistema basado en Unix (Linux o Mac) no funcionará.
- Dale un nombre que represente lo que contiene, esto no sólo por usabilidad sino por posicionamiento (SEO¹)
- En lugar de espacio usa un guión "-".
- Nunca uses acentos ni caracteres especiales: ñ, ", ", etc.

Extensiones de ficheros

Es recomendable que cada tipo de fichero tenga una extensión:

- HTML: ".html"
- JPEG: ".jpg"
- GIF: ".gif"
- PNG: ".png"

Estructura de directorios

Conforme crezca tu proyecto agradecerás tener una estructura lógica que te ayude a organizar todos los ficheros. Basándome en [esta respuesta en Stackoverflow](#)² te recomiendo seguir esta estructura³:

```
resources/  
  css/  
    main.css  
  images/  
    logo-html5.jpg  
  js/  
vendors/  
  jquery/  
    jquery.min.js  
index.html
```

Donde:

- **resources**: es un directorio que incluye los **elementos que tú has creado** y que contiene tantos directorios como tipos de recursos (css -> estilos, images -> imágenes, js -> JavaScript).
- **vendors**: para almacenar recursos creados por terceros
- Y en el fichero raíz colocar los ficheros HTML que necesites.

Aclaraciones:

1. SEO es el acrónimo de Search Engine Optimization, o lo que viene a ser lo mismo: [Optimización en Motores de Búsqueda](#). Por ejemplo, estableciendo correctamente los nombres de nuestra imágenes, es más probable que aparezcamos en buenas posiciones en [Google Images](#).
2. Stackoverflow es uno de los sitios de referencia donde podrás encontrar muchas dudas de programación, lo que lo hace realmente interesante es [el sistema de valoraciones](#) que permite discernir las "buenas" de las "malas" respuestas.

3. No todos los proyectos se deben organizar igual, en muchas ocasiones dependerá de las tecnologías que estés usando, pero para este curso esta estructura será suficientemente buena.

Errores frecuentes

Este es un listado de alguno de los errores más frecuentes en HTML:

- No poner el encoding UTF-8 hará que algunos caracteres se muestren de manera extraña
- Poner dos identificadores iguales (suele pasar al copiar y pegar código). Esto nos dará problemas de validación y al intentar acceder al elemento usando JavaScript
- Introducir & en las URLs; en su lugar se debe usar &
- Anidamiento incorrecto, ya sea por no cerrar tags en el orden correcto como por anidar elementos de bloque en elementos en línea, por ejemplo: `<h2>Título</h2>`
- Utilizar los elementos `` o `<i>` para darle estilo
- Usar múltiples consecutivamente en lugar de usar estilos

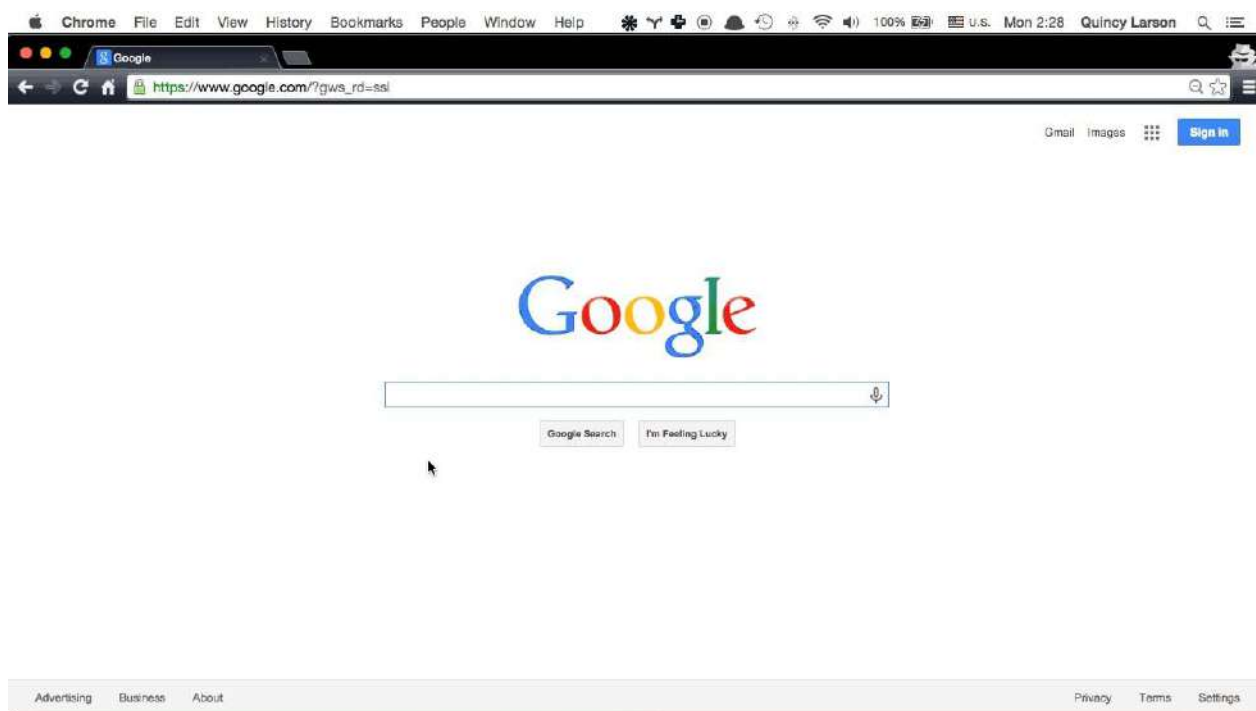
Estos son sólo algunos errores frecuentes, pero en ningún libro, manual, tutorial o curso encontrarás todos los errores que te pueden suceder, por eso es importante que aprendas a buscar las soluciones a los problemas que te vayan surgiendo, mis consejos:

- Lee **atentamente** el error
- Usa Google para buscar tu error (a ser posible busca en inglés)
- Intenta reducir la frase a las palabras clave como el lenguaje, el número de error, ...

Cuando encuentres alguna página donde parezca haber una respuesta, fíjate en:

- Que la fecha de la respuesta sea relativamente nueva (no más de 2 años)
- Busca en la documentación oficial
- Si estás en Stackoverflow revisa el número de valoraciones positivas de la respuesta

Fíjate en la siguiente animación:



Aquí se sigue el proceso recomendado salvo que se ha seleccionado una respuesta que tiene una sola valoración.

Cómo pedir ayuda

Lo más normal es que cualquier error que te ocurra cuando estés empezando ya lo haya preguntado otra persona, insiste en la búsqueda y si después de un buen rato (~30min) no encuentras el error pregunta en cualquier foro y comenta lo que te ocurre (en inglés), pero unos consejos:

- Especifica que ya has buscado antes el error
- Explica cómo lo has buscado (*por si te pueden dar algún consejo para mejorar tu forma de buscar*)
- Y finalmente haz la pregunta dando el máximo número de detalles

Es importante que cuando preguntes algo el resto vea que te has molestado en investigar previamente y también que te esfuerzas en explicar lo que te pasa, sino es probable que alguien te de una mala respuesta porque piense que no te has molestado en solucionar el problema por ti mismo.

Ejercicio

Recursos

- [Can I Use?](#)
- Mozilla CDN