

Edición 2013



Tema 1. Introducción a jQuery. Selectores.
Alejandro Amat Reina

ÍNDICE

1. OBJETIVOS	2
2. INTRODUCCIÓN	3
3. COMENZANDO A TRABAJAR CON JQUERY	6
DESCARGAR LA LIBRERÍA.....	6
INCLUIR LA LIBRERÍA JQUERY EN NUESTRA PÁGINA WEB	6
4. CONCEPTOS BÁSICOS	9
EL ALIAS DE LA LIBRERÍA '\$'	9
CONFLICTOS CON OTRAS LIBRERÍAS	9
DOCUMENT.READY.....	10
5. SELECTORES	12
SELECTORES CSS	12
SELECCIONAR ELEMENTOS UTILIZANDO EL SELECTOR UNIVERSAL	12
SELECCIONAR ELEMENTOS UTILIZANDO EL NOMBRE DE LA ETIQUETA HTML	13
SELECCIONAR ELEMENTOS UTILIZANDO EL ID.....	13
SELECCIONAR ELEMENTOS POR CLASE	14
COMBINAR SELECTORES	15
SELECCIONAR ELEMENTOS QUE SEAN DESCENDIENTES DIRECTOS DE OTROS ELEMENTOS	15
SELECCIONAR ELEMENTOS QUE SON DESCENDIENTES DE OTROS ELEMENTOS.....	16
AGRUPAR SELECTORES.....	17
SELECTOR ADYACENTE	17
FILTROS JQUERY.....	18
TABLAS TIPO CEBRA.....	19
PRIMER Y ÚLTIMO ELEMENTO DE UNA LISTA	20
ELEMENTOS QUE CONTIENEN UN ELEMENTO ESPECÍFICO.....	21
SELECCIONAR ELEMENTOS VACÍOS.....	22
SELECCIONAR ELEMENTOS SEGÚN EL TEXTO QUE CONTENGAN	22
SELECTOR DE ATRIBUTOS	23
EJEMPLOS DE USO DE SELECTORES DE ATRIBUTOS.....	24
6. FIREBUG	26
INSTALACIÓN DE FIREBUG	26
USO BÁSICO DE FIREBUG	27
LA SOLAPA HTML	27
LA SOLAPA CSS.....	28
7. METODOLOGÍA DEL CURSO	30
8. EJERCICIOS	33
EJERCICIO 1.....	33
EJERCICIO 2.....	33
EJERCICIO 3.....	33
EJERCICIO 4.....	33
RESULTADO DE LOS EJERCICIOS.....	34
9. ÍNDICE DE EJEMPLOS Y TABLAS	35
EJEMPLOS	35
TABLAS	35

1. Objetivos

En este primer tema se pretenden conseguir los siguientes objetivos:

- ✚ Comprender la metodología del curso.
- ✚ Incluir la librería jQuery en nuestras páginas Web.
- ✚ Conocer el concepto de selector.
- ✚ Realizar selecciones de elementos existentes en la página Web.

2. Introducción

Existen muchas tecnologías que se pueden emplear para programar los clientes Web, pero las consideradas como estándar de facto son:

- ✚ El lenguaje html, para la introducción de los contenidos.
- ✚ Las hojas de estilo en cascada (css), para gestionar la visualización de esos contenidos.
- ✚ El lenguaje javascript, para introducir interactividad.

Con el tiempo, los sitios Web han crecido en contenidos y funcionalidades y con la aparición de la Web 2.0, se hizo necesaria la incorporación de nuevas tecnologías que simplificaran la tarea de desarrollar estos sitios Web interactivos. En este contexto, aparecen librerías o frameworks javascript que nos simplificarán las tareas más habituales de la programación Web en la parte cliente. Una de estas librerías (quizá la más extendida y utilizada en la actualidad) es la librería jQuery.

jQuery es una librería javascript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos html, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con ajax a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC. Es software libre y de código abierto, posee una doble licencia (Licencia MIT y Licencia Pública General de GNU v2), permitiendo su uso en proyectos libres y privados. Al igual que otras librerías, ofrece una serie de funcionalidades basadas en javascript, que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca, se logran grandes resultados en menos tiempo y espacio.

La documentación de jQuery (<http://docs.jquery.com/>) es muy completa en inglés e incluye muchos ejemplos. También existen algunos recursos útiles en español para aprender su funcionamiento básico.

En este curso trataremos de mostrar una visión general de la librería y veremos las características principales de la misma.

Las facilidades que nos ofrece jQuery son, entre otras, las siguientes:

- ✚ Acceder a elementos en un documento de forma sencilla: Sin una librería javascript, se tienen que escribir muchas líneas de código para recorrer el árbol DOM.

- ✚ Modificar la apariencia de una página Web: jQuery puede cambiar las clases o propiedades de estilo CSS individual aplicadas a una parte del documento, incluso después de que se haya mostrado la página.
- ✚ Alterar el contenido de un documento: El texto se puede cambiar, las imágenes se pueden insertar o cambiar, las listas se pueden reordenar, o toda la estructura del HTML se puede volver a escribir y ampliar.
- ✚ Responder a la interacción de un usuario: jQuery ofrece una forma elegante de interceptar una amplia variedad de eventos, sin la necesidad de saturar el propio código html con manejadores de evento.
- ✚ Animar cambios realizados a un documento: Un diseñador debe también proporcionar feedback visual al usuario. La biblioteca jQuery facilita esto al proporcionar una tabla de efectos, como desvanecerse, así como un conjunto de herramientas para diseñar nuevos efectos.
- ✚ Recuperar información de un servidor sin refrescar una página: jQuery simplifica el uso de las peticiones Ajax para obtener contenidos del servidor sin necesidad de recargar la página.
- ✚ Resolver los conflictos entre navegadores: Históricamente uno de los principales inconvenientes que nos encontrábamos al trabajar con javascript y css, eran las incompatibilidades que tienen estos lenguajes entre distintos navegadores. Cuando utilizamos jQuery, estas incompatibilidades se resuelven a nivel interno, por lo tanto, en la mayoría de los casos, podemos olvidarnos de ellas.
- ✚ Siempre trabajar con conjuntos (iteración implícita): Cuando le decimos a jQuery, "encuentra todos los elementos con la clase `.colapsable` y ocúltalos", no hay necesidad de pasar un bucle por cada elemento devuelto. En su lugar, los métodos jQuery se diseñan para trabajar automáticamente sobre conjuntos de objetos en lugar de elementos individuales.
- ✚ Permitir múltiples acciones en una línea (encadenamiento o chaining): Para evitar el uso excesivo de variables temporales, jQuery emplea un patrón de programación denominado encadenamiento para la mayoría de sus métodos. Esto significa, que el resultado de la mayoría de operaciones en un objeto, es el objeto en sí mismo, listo para la siguiente acción que se le va a aplicar.

Es fácil olvidar que jQuery es una librería javascript. Después de trabajar con la librería una cantidad de tiempo considerable, jQuery nos hace pensar que utilizamos un lenguaje propio, pero debemos tener claro que no nos ofrece ninguna nueva funcionalidad. Lo que nos va a permitir jQuery, es hacer, de forma sencilla, cosas que nos resultarían muy complicadas de programar, si no utilizáramos una librería.

3. Comenzando a trabajar con jQuery

Descargar la librería

La librería jQuery es un archivo javascript, al cual, podemos acceder de dos formas:

- 🚦 Descargando el fichero jQuery.js y alojarlo localmente en nuestro sitio Web. Para ello, iremos a la página <http://www.jquery.com>. En esta página encontraremos dos versiones de la misma librería: una versión para producción que está comprimida y optimizada para obtener la mayor velocidad de descarga posible y una versión de desarrollo que podremos depurar en caso de ser necesario. Una vez descargada la versión correspondiente de la librería, la asociaremos a nuestros documentos html cómo se verá en el siguiente punto.
- 🚦 Usar una versión alojada en un CDN (content delivery network), por ejemplo, la versión alojada en el CDN de Google o en la propia web de jQuery. En este caso, indicaremos la url correspondiente a la versión de la librería que queremos utilizar. En los siguientes enlaces podemos ver las librerías jQuery alojadas en google y en la Web de jQuery:

- <https://developers.google.com/speed/libraries/devguide#jquery>
- <http://code.jquery.com/>

Lo más recomendable es utilizar una versión local mientras estamos en desarrollo. En la versión de producción puede ser interesante utilizar una versión alojada en un sitio externo, sobre todo, si el ancho de banda del que dispone nuestro servidor, es limitado.

Incluir la librería jQuery en nuestra página Web

Para incluir la librería jQuery en nuestra página, por ejemplo, utilizando el CDN de jQuery, debemos introducir el siguiente código dentro del <head>:

```
<script type="text/javascript" src="http://code.jquery.com/jquery-1.9.0.min.js">
</script>
```

Debemos asegurarnos de dos cosas:

- 🚦 Antes de incluir la librería jQuery, debemos haber incluido todos los ficheros de hojas de estilo que vayamos a utilizar. De esta forma, nos aseguraremos

de que cuando el DOM haya terminado de cargarse, ya estén todos los estilos aplicados.

- ✚ Cualquier fichero javascript que vaya a utilizar la librería jQuery, debe incluirse en la página después de esta librería.

Según lo anterior, el orden de los archivos vinculados a nuestra página debe ser el siguiente: primero vincularemos las hojas de estilo, a continuación, vincularemos la librería jQuery, y finalmente, vincularemos todos los archivos javascript que vayamos a utilizar.

Siempre deberíamos incluir un doctype en nuestros documentos HTML. Si no incluimos un doctype podemos encontrarnos con muchos inconvenientes, como por ejemplo, que no podremos validar el documento para verificar si tenemos algún error de sintaxis. Además, jQuery puede no funcionar correctamente si no existe un doctype en la página.

Para simplificar, a lo largo de este curso, utilizaremos el doctype de html 5:

```
<!DOCTYPE html>
```

El siguiente ejemplo, es un documento html básico en el que podemos ver como se incluye la librería jQuery, se vinculan antes las hojas de estilos y después el resto de ficheros javascript. Además se utiliza el doctype de html 5:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo incluir librería jQuery en local</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link href="css/global.css" rel="stylesheet"/>
    <script src="js/jquery.js" type="text/javascript"></script>
    <script src="js/otrosJavascript.js" type="text/javascript">
    </script>
  </head>
  <body>
  </body>
</html>
```

Ejemplo 1: [documentoBasico1.html](#)

Si en lugar de utilizar la librería alojada localmente, queremos utilizar la versión alojada en la web de jQuery, el documento básico quedaría de la siguiente forma:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo incluir librería jQuery de CDN</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link href="css/global.css" rel="stylesheet"/>
    <script src="http://code.jquery.com/jquery-1.9.0.min.js" type="text/javascript">
    </script>
```



```
<script src="js/otrosJavascript.js" type="text/javascript"> </script>
</head>
<body>
</body>
</html>
```

Ejemplo 2: [documentoBasico2.html](#)

Aunque no es una buena práctica de programación, para simplificar la lectura y comprensión de los ejemplos, se ha añadido el código jQuery de los mismos, dentro de una etiqueta <script> del <head>. En cualquier caso, en los ejercicios que el alumno tiene que resolver, deberá insertar todo el código jQuery en un fichero javascript, que será vinculado al documento html.

4. Conceptos básicos

Una de las principales facilidades que nos ofrece la librería jQuery es la manipulación de los elementos del DOM. Podemos buscar elementos, añadir nuevos y modificar o eliminar elementos existentes, con mucho menos código que si utilizáramos javascript nativo.

El alias de la librería '\$'

Para acceder a cualquier elemento de la librería utilizamos el símbolo '\$'. Este símbolo es un alias de la librería jQuery en sí misma, de forma que, todas las instrucciones que accedan a la librería jQuery comenzarán con este símbolo. El siguiente código, es un ejemplo de una sentencia de selección de elementos en jQuery:

```
$(selector)
```

Existen diferentes métodos para seleccionar elementos del DOM. Los veremos más adelante.

Conflictos con otras librerías

Pueden ocurrir conflictos entre jQuery y otras librerías que también utilizan el mismo símbolo como alias, por ejemplo, la librería Prototype. Para solucionar estos conflictos tenemos dos opciones:

- ✚ Añadir la función `noConflict` al final de la librería jQuery. Esto prevendrá el conflicto con otras librerías que se enlacen más adelante en el mismo documento.

```
$.noConflict();
```

- ✚ Eliminar todas las referencias al símbolo '\$' y sustituirlas por la palabra jQuery, que será otro alias, pero en este caso no tendrá conflictos. Por ejemplo, en lugar de escribir este código:

```
$(document).ready(...
```

Escribiremos este otro:

```
jQuery(document).ready(...
```

- ✚ También podemos crear nuestro propio alias de la librería de la siguiente forma:

```
var $miAlias = jQuery;  
$miAlias(document).ready(...
```

document.ready

Antes de comenzar a manipular elementos del DOM, debemos asegurarnos de que todos han sido cargados. Para ello jQuery dispone de un método de evento conocido como `document.ready`.

Este método se ejecutará después de que todos los elementos del DOM de la página hayan sido cargados. Veamos un ejemplo:

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Ejemplo document.ready</title>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
    <script  
      src="http://code.jquery.com/jquery-1.9.0.min.js"  
      type="text/javascript"></script>  
    <script type="text/javascript">  
      $(document).ready(function()  
      {  
        alert("El DOM está cargado y preparado");  
      });  
    </script>  
  </head>  
<body>  
</body>  
</html>
```

Ejemplo 3: [documentReady.html](#)

En el ejemplo, lo que hacemos en primer lugar es seleccionar el documento:

`$(documento)`

A continuación, asociamos con el evento `ready` del documento, la función que queremos que se ejecute cuando el DOM se haya cargado. En este caso, hemos utilizado una función anónima, que definimos dentro de la propia asociación del evento.

La función `window.load` de javascript es parecida al `document.ready` de jQuery, pero esta última tiene una serie de ventajas:

- 🚦 La función `window.load` se ejecuta cuando la página está completamente cargada, por lo tanto, espera a que los objetos gráficos y demás elementos externos estén descargados. `document.ready`, por su parte, sólo espera a que el DOM esté cargado, pero no espera a que los elementos externos se hayan descargado.

- ✚ Uno de los grandes problemas de la función `window.load` de javascript, es que si le asignamos dos funciones distintas en dos partes de código diferentes, la segunda sobrescribirá a la primera. En jQuery la función `document.ready` añade funciones al evento, pero no sustituye las anteriores, por lo tanto, podemos invocarla las veces que necesitemos y todas las funciones que añadamos, se ejecutarán cuando la página esté lista.

Este evento `.ready()` sólo puede asociarse al objeto `document`, por lo tanto, es muy habitual utilizar un código equivalente simplificado, en el que se omiten las palabras `document.ready`. El siguiente ejemplo es equivalente al anterior:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo document.ready simplificado</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script
      src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        alert("El DOM está cargado y preparado");
      });
    </script>
  </head>
</body>
</body>
</html>
```

Ejemplo 4: [documentReadySimplificado.html](#)

5. Selectores

Los selectores son los pilares básicos de jQuery. Cualquier cosa que queramos hacer que necesite manipular el DOM, necesitará la incorporación de un selector. Esto es así, porque necesitamos elegir qué elementos del DOM queremos manipular.

jQuery trabaja con selectores utilizados comúnmente en CSS o en XPath, por lo que, deberíamos estar familiarizados con ellos. Entender el funcionamiento de los selectores, es fundamental para sacarle el mayor partido a la librería.

Un selector es una expresión de cadena que **seleccionará un elemento o conjunto de elementos del DOM** para que podamos trabajar con ellos. Lo declararemos siempre después del alias '\$' y entre paréntesis. Una vez seleccionados, los elementos se convierten en objetos jQuery, con lo que, podremos manipular sus eventos, aplicar efectos, modificar su contenido, sus estilos, etc.

Selectores CSS

En esta sección vamos a ver los selectores de CSS que se pueden aplicar para jQuery. En los ejemplos, utilizaremos la función `css` para cambiar los estilos de los elementos seleccionados. Esta función recibirá dos parámetros: el primero será la propiedad CSS que queremos modificar, el segundo será el valor que queremos darle a dicha propiedad. Por ejemplo:

```
$("#div").css("border", "1px solid #333");
```

El selector es `$('#div')` y seleccionará todos los `<div>` que existan en el documento. Posteriormente, el método `.css()`, cambiará el estilo del borde de los `<div>` seleccionados.

Seleccionar elementos utilizando el selector universal

Se utiliza para seleccionar todos los elementos de la página. El siguiente ejemplo añadirá un borde a todos los elementos de la página:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo selector universal</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
```

```
<script type="text/javascript">
  $(function()
  {
    $("*").css("border", "1px solid #333");
  });
</script>
</head>
<body>
  <h1>Título</h1>
  <p>párrafo de texto</p>
</body>
</html>
```

Ejemplo 5: [selectorUniversal.html](#)

Seleccionar elementos utilizando el nombre de la etiqueta HTML

Selecciona todos los elementos de la página cuya etiqueta html coincide con el valor del selector. Internamente, se utiliza la función javascript `getElementsByTagName()`.

En el siguiente ejemplo, le cambiaremos el tipo de fuente a todos los elementos `<h1>` de la página:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo selector etiqueta</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $("h1").css("font-family", "Courier");
      });
    </script>
  </head>
  <body>
    <h1>Título</h1>
    <p>párrafo de texto</p>
  </body>
</html>
```

Ejemplo 6: [selectorEtiqueta.html](#)

Seleccionar elementos utilizando el ID

El selector de ID permite seleccionar un elemento de la página a través del valor de su atributo id. Este tipo de selectores sólo seleccionan un elemento de la página, ya que, el valor del atributo id no se puede repetir en dos elementos diferentes de una misma página.

Se utiliza el símbolo '#' como prefijo del id para que jQuery identifique un selector de este tipo. Internamente se utiliza la función javascript `getElementById()`. En el siguiente ejemplo, se oculta el elemento cuyo id es `#parrafo`:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo selector id</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $("#parrafo").css("display", "none");
      });
    </script>
  </head>
  <body>
    <h1>Título</h1>
    <p id="parrafo">párrafo de texto</p>
  </body>
</html>
```

Ejemplo 7: [selectorId.html](#)

Seleccionar elementos por clase

Selecciona todos aquellos elementos de la página que tengan aplicado el estilo de clase indicado en el selector. Al igual que en css, para indicar un selector de clase tendremos que poner un '.' como prefijo. Internamente se utiliza la función javascript `getElementsByClassName()`.

En el ejemplo, vamos a cambiar varias propiedades css en una sola instrucción, por lo que, tendremos que añadir las llaves '{' y '}' en la llamada:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo selector clase</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $(".titulo").css(
          {"padding": "5px", "border": "1px solid #ccc", "width": "150px"}
        );
      });
    </script>
  </head>
  <body>
    <h1 class="titulo">Título</h1>
    <p id="parrafo">párrafo de texto</p>
  </body>
</html>
```

Ejemplo 8: [selectorClase.html](#)

De esta forma, el método `.css()` funcionará como si estuviéramos definiendo la regla de estilo en el documento css.

Combinar selectores

En ocasiones tenemos elementos que tienen más de un estilo de clase aplicado. Nos puede interesar seleccionar sólo aquellos elementos que tienen aplicados varios estilos de clase. Esto lo haremos indicando en el selector las clases que nos interesan, una a continuación de la otra y con su correspondiente `'.'` delante (no separaremos las clases con espacios). Del mismo modo, nos puede interesar seleccionar aquellos elementos que son de un tipo (etiqueta) determinado y pertenecen a una clase.

En el siguiente ejemplo, se ocultarán aquellos elementos que sean de la clase `libro` e inactivo y se cambiará el alto de línea para los `` que sean de la clase `libro`:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo selectores combinados</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $(".libro.inactivo").css("display","none");
        $("li.libro").css("line-height", "3em");
      });
    </script>
  </head>
  <body>
    <h1>Lista de libros</h1>
    <ul>
      <li class="libro">La isla del tesoro</li>
      <li class="libro">Los miserables</li>
      <li class="libro inactivo">El traidor</li>
      <li class="libro">Canción de cuna</li>
    </ul>
  </body>
</html>
```

Ejemplo 9: [selectoresCombinados.html](#)

Seleccionar elementos que sean descendientes directos de otros elementos

Podemos seleccionar un elemento que es hijo directo de otro elemento. Lo indicaremos mediante el signo `'>'`. Veamos el siguiente ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo selectores hijos directos</title>
```



```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="http://code.jquery.com/jquery-1.9.0.min.js"
        type="text/javascript"></script>
<script type="text/javascript">
    $(function()
    {
        $("p > span").css("color", "red");
    });
</script>
</head>
<body>
    <p>El texto <span>rojo</span> lo marco con span </p>
    <p><strong><span>Esto no es rojo</span></strong></p>
</body>
</html>
```

Ejemplo 10: [selectoresHijosDirectos.html](#)

En el ejemplo anterior, el selector `p > span` se interpreta como "cualquier elemento `` que sea hijo directo de un elemento `<p>`", por lo que el primer elemento `` cumple la condición del selector. Sin embargo, el segundo elemento `` no la cumple porque es descendiente, pero no es hijo directo de un elemento `<p>`.

Seleccionar elementos que son descendientes de otros elementos

Hemos visto en el ejemplo anterior, que sólo se seleccionaban los descendientes directos de un elemento, pero ¿qué ocurre si nos interesa seleccionar todos los descendientes, y no sólo los directos? En ese caso, utilizaremos el selector descendiente.

Veamos el ejemplo anterior, pero utilizando el selector descendiente, en lugar del selector de hijos:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Ejemplo selectores descendientes</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <script src="http://code.jquery.com/jquery-1.9.0.min.js"
                type="text/javascript"></script>
        <script type="text/javascript">
            $(function()
            {
                $("p span").css("color", "red");
            });
        </script>
    </head>
    <body>
        <p>El texto <span>rojo</span> lo marco con span </p>
        <p><strong><span>Esto no es rojo</span></strong></p>
    </body>
</html>
```

Ejemplo 11: [selectorDescendientes.html](#)

Con el selector descendente se aplica a todos los elementos `` que se encuentran dentro de elementos `<p>`. Por lo tanto, en este caso, al contrario que en el anterior, los estilos del selector se aplican a los dos ``.

Agrupar selectores

En ocasiones nos puede interesar seleccionar múltiples tipos de elementos, los cuales podrían ser una combinación de clases, ids, etiquetas, etc. Podemos añadir múltiples elementos al selector creando una lista de los mismos, para ello, los separaremos por comas.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo selectores agrupados</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $("h1, .parrafo, #destacado, p > span").css("color", "red");
      });
    </script>
  </head>
  <body>
    <h1>Título principal</h1>
    <p class="parrafo">texto destacado</p>
    <p>El texto <span>rojo</span> lo marco con span </p>
    <p><strong id="destacado">Esto sí es rojo</strong></p>
  </body>
</html>
```

Ejemplo 12: [selectoresAgrupados.html](#)

En el ejemplo, todas las etiquetas `<h1>`, todos los elementos que sean de la clase `.parrafo`, el elemento con id `#destacado` y todos los `` que sean descendentes directos de una etiqueta `<p>`, tendrán la fuente de color rojo.

Selector adyacente

El selector adyacente utiliza el signo ``+`` y su sintaxis es: `elemento1 + elemento2`. La explicación del comportamiento de este selector no es sencilla, ya que selecciona todos los elementos de tipo `elemento2`, que cumplan las dos siguientes condiciones:

- ✚ `elemento1` y `elemento2` deben ser hermanos, por lo que su elemento padre debe ser el mismo.

- ✚ elemento2 debe aparecer inmediatamente después de elemento1 en el código HTML de la página.

Observemos el siguiente ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo selector adyacente</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $("h1+h2").css("color", "red");
      });
    </script>
  </head>
  <body>
    <h1>Titulo1</h1>
    <h2>Subtítulo</h2>
    <p>texto del subtítulo</p>
    <h2>Otro subtítulo</h2>
  </body>
</html>
```

Ejemplo 13: [selectorAdyacente.html](#)

Los estilos del selector `h1+h2` se aplican al primer elemento `<h2>` de la página, pero no al segundo `<h2>`, ya que:

- ✚ El elemento padre de `<h1>` es `<body>`, el mismo padre que el de los dos elementos `<h2>`. Así, los dos elementos `<h2>` cumplen la primera condición del selector adyacente.
- ✚ El primer elemento `<h2>` aparece en el código html justo después del elemento `<h1>`, por lo que, este elemento `<h2>` también cumple la segunda condición del selector adyacente.
- ✚ Por el contrario, el segundo elemento `<h2>` no aparece justo después del elemento `<h1>`, por lo que, no cumple la segunda condición del selector adyacente y, por tanto, no se le aplican los estilos de `h1+h2`.

Filtros jQuery

Los filtros jQuery nos permiten refinar los elementos que estamos seleccionando. Por ejemplo, nos puede interesar seleccionar las filas pares o impares de una tabla, el primer elemento de una lista, etc. Para poder hacer esto utilizando selectores css, tendríamos que aplicar estilos de clase a los elementos que queremos seleccionar, lo cual puede ser muy tedioso, si la página la generamos

dinámicamente. Utilizando filtros jQuery, estas selecciones las haremos de forma muy cómoda.

Un filtro se define con dos puntos ':' y a continuación la palabra que define el tipo de filtro que queremos aplicar. Por ejemplo:

:filter

A continuación, veremos una tabla con los filtros disponibles en jQuery. Algunos de estos filtros están disponibles a partir de la aparición de la pseudo-clase correspondiente en el CSS 3. El inconveniente es que la versión 3 de CSS, sólo está disponible en los navegadores más modernos (para ver el porcentaje de compatibilidad con CSS 3 del navegador que utilizamos, podemos ir a la página <http://css3test.com>). En la tabla se indica si el filtro es de CSS 3.

Filtro	Funcionamiento	CSS3
:even y :odd	Selecciona los ítems pares o impares en base a su índice	
:header	Selecciona elementos que son etiquetas h1, h2, h3, etc.	
:not	Selecciona elementos que no coinciden con el selector especificado	SI
:eq(índice)	Selecciona los elementos que coinciden con el índice	
:gt(índice)	Selecciona elementos mayores que el índice	
:lt(índice)	Selecciona elementos menores que el índice	
:first-child, :last-child	Selecciona los elementos hijos primero o último	SI
:only-child	Selecciona los que sean hijos únicos	SI
:nth-child(índice)	Selecciona el hijo correspondiente al índice indicado	SI
:has(p)	Selecciona los elementos que contienen otro elemento	
:contains('texto')	Selecciona los elementos que contengan el texto especificado	
:empty	Selecciona los elementos que estén vacíos	SI
:parent	Selecciona el elemento padre	
:hidden	Selecciona los elementos ocultos	
:visible	Selecciona los elementos visibles	
:animated	Selecciona los elementos que estén en proceso de animación	

Tabla 1: Filtros jQuery

Para entender mejor el funcionamiento, veremos a continuación algunos ejemplos de uso de filtros.

Tablas tipo cebra

Para facilitar la lectura de los datos de una tabla, es muy habitual añadir un color de fondo diferente a las filas pares e impares (observar la tabla anterior). Para hacer esto con jQuery utilizaremos los filtros :even y :odd.

```
<!DOCTYPE html>
<html>
  <head>
```

```

<title>Filtros odd y even</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
<script type="text/javascript">
  $(function()
  {
    $('tr:even').css('background', '#dedede');
    $('tr:odd').css('background', '#ffffff');
  });
</script>
</head>
<body>
<table>
<tr>
  <th>Producto</th><th>Descripción</th><th>Precio</th>
</tr>
<tr>
  <td>Estuche de pinturas</td>
  <td>Pinturas de varios colores</td>
  <td>18,99 </td>
</tr>
<tr>
  <td>Compás</td>
  <td>De diferentes tamaños</td>
  <td>16,99 </td>
</tr>
<tr>
  <td>Folios</td>
  <td>Paquete de 500 folios</td>
  <td>5,99 </td>
</tr>
<tr>
  <td>Cartulina</td>
  <td>Tamaño din A2</td>
  <td>2,99 </td>
</tr>
</table>
</body>
</html>

```

Ejemplo 14: [filtrosOddEven.html](#)

Primer y último elemento de una lista

Si queremos seleccionar el primer y/o último elemento de un conjunto de elementos del DOM, seleccionados mediante un selector, podemos utilizar los filtros `:first` y `:last`.

```

<!DOCTYPE html>
<html>
<head>
  <title>filtros first y last</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style>
    ul { width:200px;font-family:arial; }
    ul li a { text-decoration:none; }
  </style>
  <script src="http://code.jquery.com/jquery-1.9.0.min.js"

```

```

        type="text/javascript"></script>
<script type="text/javascript">
    $(function()
    {
        $('ul li:first').css('border-top','1px solid red');
        $('ul li:last').css('border-bottom','1px solid red');
    });
</script>
</head>
<body>
<div id='menu'>
<ul>
<li><a href='/index'>Home</a></li>
<li><a href='/about'>About Us</a></li>
<li><a href='/customer-service'>Customer Service</a></li>
<li><a href='/contact'>Contact Us</a></li>
<li><a href='/coupons'>Coupons</a></li>
</ul>
</div>
</body>
</html>

```

Ejemplo 15: [filtrosFirstLast.html](#)

Elementos que contienen un elemento específico

Podemos necesitar seleccionar elementos que contengan un elemento específico dentro de ellos. En estos casos, podemos usar el filtro `:has()`. No es necesario que el elemento hijo, sea hijo directo, bastará con que sea descendiente.

En el siguiente ejemplo, se mostrará el texto subrayado en todos aquellos `<div>` que contengan algún elemento enfatizado (etiqueta ``).

```

<!DOCTYPE html>
<html>
<head>
<title>filtro has()</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="http://code.jquery.com/jquery-1.9.0.min.js"
        type="text/javascript"></script>
<script type="text/javascript">
    $(function()
    {
        $('div:has(strong)').css('text-decoration','underline');
    });
</script>
</head>
<body>
<div>
<p>Este está <strong>resaltado</strong></p>
</div>
<div>
<p>Este no está resaltado</p>
</div>
</body>
</html>

```

Ejemplo 16: [filtroHas.html](#)

Seleccionar elementos vacíos

Para seleccionar elementos que no contengan nada dentro usaremos el filtro `:empty`.

En el siguiente ejemplo, se pondrá un borde rojo alrededor de las celdas de la tabla que estén vacías:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Filtros empty</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $('td:empty').css('border','2px dotted red');
      });
    </script>
  </head>
  <body>
    <table>
      <tr>
        <th>Producto</th><th>Descripción</th><th>Precio</th>
      </tr>
      <tr>
        <td>Estuche de pinturas</td>
        <td>Pinturas de varios colores</td>
        <td>18,99 </td>
      </tr>
      <tr>
        <td>Compás</td>
        <td>De diferentes tamaños</td>
        <td></td>
      </tr>
      <tr>
        <td>Folios</td>
        <td>Paquete de 500 folios</td>
        <td>5,99 </td>
      </tr>
      <tr>
        <td>Cartulina</td>
        <td>Tamaño din A2</td>
        <td>2,99 </td>
      </tr>
    </table>
  </body>
</html>
```

Ejemplo 17: [filtroEmpty.html](#)

Seleccionar elementos según el texto que contengan

Podemos seleccionar los elementos en función de su contenido. Para ello, usaremos el filtro `:contains`.

En el siguiente ejemplo, se seleccionarán las celdas de la tabla que contengan el texto '€':

```
<!DOCTYPE html>
<html>
  <head>
    <title>Filtro contains</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $("td:contains('€')").css('border', '2px dotted red');
      });
    </script>
  </head>
  <body>
    <table>
      <tr>
        <th>Producto</th><th>Descripción</th><th>Precio</th>
      </tr>
      <tr>
        <td>Estuche de pinturas</td>
        <td>Pinturas de varios colores</td>
        <td>18,99 €</td>
      </tr>
      <tr>
        <td>Compás</td>
        <td>De diferentes tamaños</td>
        <td>9,45 €</td>
      </tr>
      <tr>
        <td>Folios</td>
        <td>Paquete de 500 folios</td>
        <td>5,99 €</td>
      </tr>
      <tr>
        <td>Cartulina</td>
        <td>Tamaño din A2</td>
        <td>2,99 €</td>
      </tr>
    </table>
  </body>
</html>
```

Ejemplo 18: [filtroContains.html](#)

Selector de atributos

El último tipo de selectores avanzados lo forman los selectores de atributos, que permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.

A continuación, veremos una tabla donde se muestran los distintos tipos de selectores de atributo existentes:

Selector	Funcionamiento
<code>\$('[atributo]')</code>	Selecciona los elementos que tienen establecido el atributo, independientemente de su valor.
<code>\$('[atributo*="texto"]')</code>	Selecciona los elementos que tienen establecido el atributo y su valor contiene el texto indicado.
<code>\$('[atributo ="texto"]')</code>	Selecciona los elementos que tienen establecido el atributo y su valor es una serie de palabras separadas con guiones, pero que comienza con texto. Este tipo de selector sólo es útil para los atributos de tipo lang que indican el idioma del contenido del elemento.
<code>\$('[atributo~="texto"]')</code>	Selecciona los elementos que tienen establecido el atributo y uno de sus valores es el texto indicado (los valores se separan por espacios).
<code>\$('[atributo\$="texto"]')</code>	Selecciona los elementos que tienen establecido el atributo y su valor termina por el texto indicado
<code>\$('[atributo="texto"]')</code>	Selecciona los elementos que tienen establecido el atributo y su valor coincide con el texto indicado
<code>\$('[atributo^="texto"]')</code>	Selecciona los elementos que tienen establecido el atributo y su valor comienza por el texto indicado
<code>\$('[atributo!="texto"]')</code>	Selecciona los elementos que tienen establecido el atributo y su valor NO coincide con el texto indicado
<code>\$('[atributo1="texto1" [atributo2="texto2" [atributo3="texto3"]')</code>	Selecciona los elementos que tienen establecido todos los atributos indicados con los valores indicados.

Tabla 2: Selectores de atributos

Ejemplos de uso de selectores de atributos

Imaginemos que queremos seleccionar todos aquellos enlaces que van dirigidos a la página del GVA y quitarles el subrayado. Haremos lo siguiente:

```
<!DOCTYPE html>
<html>
  <head>
    <title>selector de atributos *=</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $("a[href*='gva.es']").css("text-decoration", "none");
      });
    </script>
  </head>
  <body>
    <ul>
      <li><a href="http://www.edu.gva.es">Consellería de educación</a></li>
      <li><a href="http://www.google.es">Buscador Google</a></li>
      <li><a href="http://cefired.gva.es">Moodle del cefire</a></li>
    </ul>
  </body>
</html>
```

Ejemplo 19: [selectorAtributos1.html](#)

Supongamos ahora, que tenemos en nuestra página tres entradas de un blog, cada una de ellas dentro de un `<div>`. A los `<div>` le hemos puesto los siguientes ids:

primera-entrada, segunda-entrada y tercera-entrada. Queremos seleccionar los tres <div> y aplicar una serie de estilos ¿cómo lo haremos? muy sencillo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>selector de atributos $=</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="http://code.jquery.com/jquery-1.9.0.min.js"
      type="text/javascript"></script>
    <script type="text/javascript">
      $(function()
      {
        $("div[id$='-entrada']").css(
        {
          "margin": "5px", "border": "1px solid #ccc",
          "width": "300px", "background-color" : "blue"
        });
      });
    </script>
  </head>
  <body>
    <div id="primera-entrada">
      Texto de la primera entrada
    </div>
    <div id="segunda-entrada">
      Texto de la segunda entrada
    </div>
    <div id="tercera-entrada">
      Texto de la tercera entrada
    </div>
  </body>
</html>
```

Ejemplo 20: [selectorAtributos2.html](#)

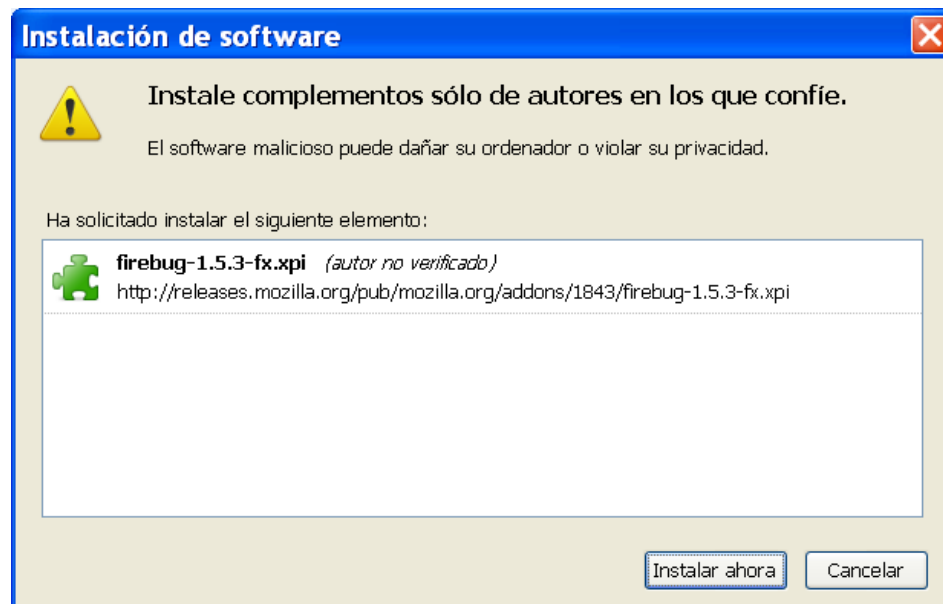
Hemos aprovechado que los tres ids terminan igual para seleccionarlos con el selector de atributo '\$='.

6. Firebug


Firebug es una extensión gratuita del navegador Mozilla Firefox. Firebug nos aporta una serie de funcionalidades muy interesantes, como por ejemplo, la posibilidad de inspeccionar los elementos de la página o, mejor todavía, depurar el código javascript de la misma. Lo único que tendremos que hacer para empezar a utilizarla es descargarla e instalarla.

Instalación de Firebug

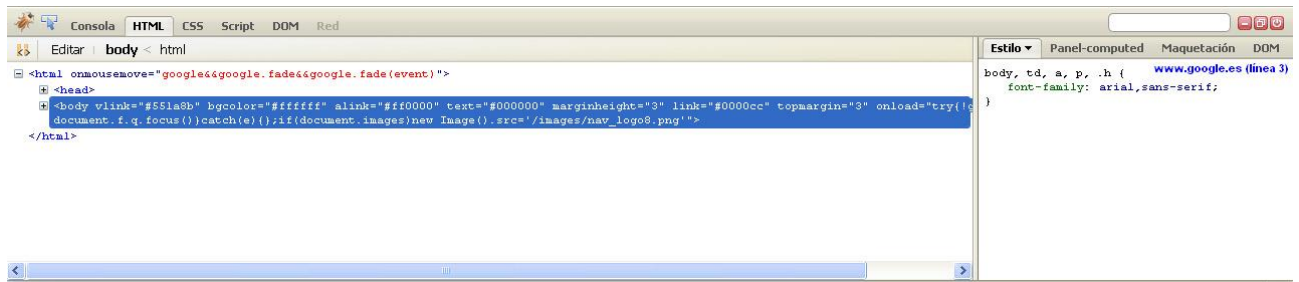
Podemos instalar el complemento directamente de la página oficial (<http://getfirebug.com/>). Para ello, sólo tenemos que abrir la dirección con el navegador Firefox y pulsar el botón Install Firebug.



En el momento en el que se crearon estos materiales la versión disponible de Firebug era la 1.5.3., pero las principales características de la herramienta siguen funcionando igual.

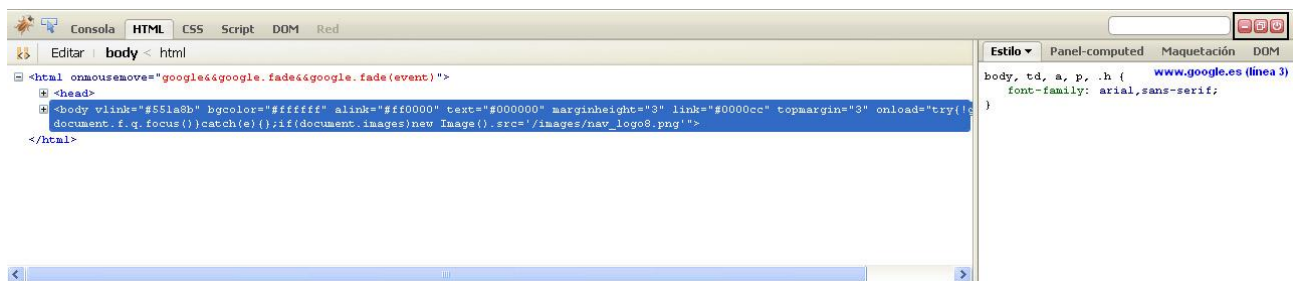
Una vez instalado el complemento nos indicará que debemos reiniciar el navegador y tras el reinicio tendremos el icono de firebug  en la esquina inferior derecha de la pantalla.



Si pulsamos sobre este icono se desplegará la ventana de Firebug:





Uso básico de Firebug

En la ventana de Firebug tenemos tres botones de color rojo (esquina superior derecha) que sirven para (de izquierda a derecha) minimizar la ventana, desanclar la ventana del navegador (pasando a tener una ventana independiente) y cerrar Firebug.



Podemos observar en la parte izquierda que la ventana está dividida en solapas: Consola, HTML, CSS, Script, DOM y Red. Cada una de estas solapas nos muestra la parte correspondiente de la página que estamos visualizando. Si nos situamos en la solapa HTML nos mostrará el código HTML del documento de forma jerárquica, con el símbolo  para desplegar los elementos y el símbolo  para replegarlos.

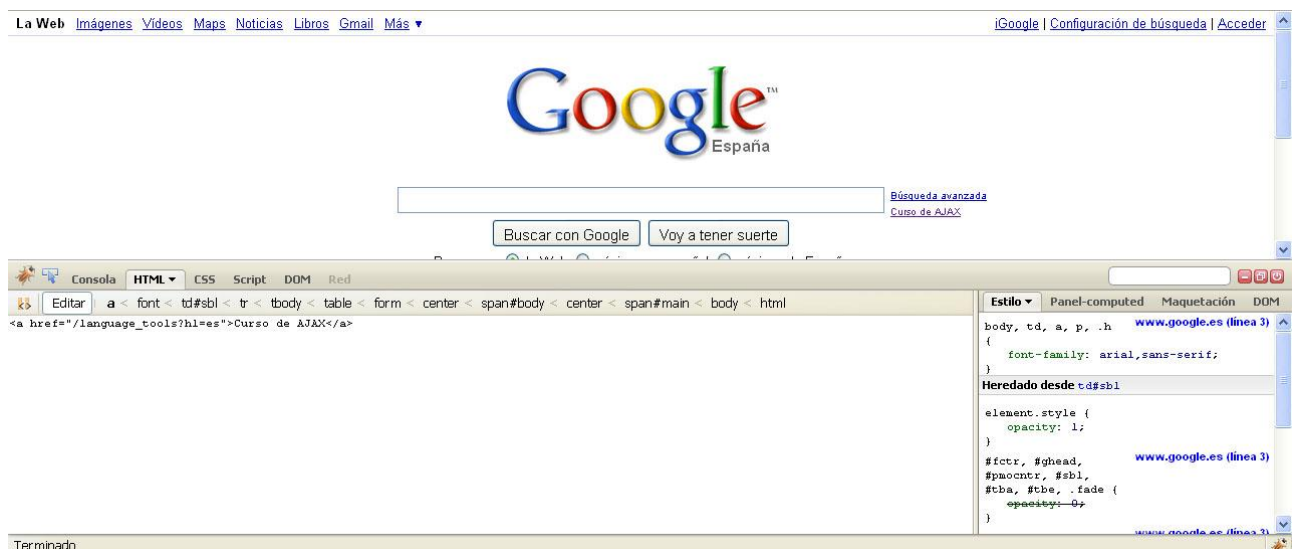
La solapa HTML

A la izquierda de las solapas tenemos dos iconos  el primero despliega el menú principal de la herramienta y el segundo nos servirá para inspeccionar los elementos HTML de la página. Si pulsamos el icono  y nos vamos moviendo por la página Web, observaremos como se van seleccionando los distintos elementos sobre los que pasamos y como se va mostrando el código correspondiente al elemento seleccionado en la solapa HTML de Firebug. Si hacemos clic con el botón izquierdo del ratón sobre el elemento, seleccionaremos su código en la solapa HTML. Cuando seleccionamos un elemento nos aparecerá en la parte superior de la solapa los elementos que tenemos que recorrer dentro de la jerarquía del documento HTML para llegar al elemento seleccionado. En la parte derecha de la

ventana nos aparecerán los estilos que afectan a dicho elemento, podemos mostrar estos estilos de distintas formas: tal y como aparecen en la hoja de estilos (solapa Estilo), ordenados por categorías (solapa Panel-computed) o visualizando el modelo de caja del elemento (solapa Maquetación).



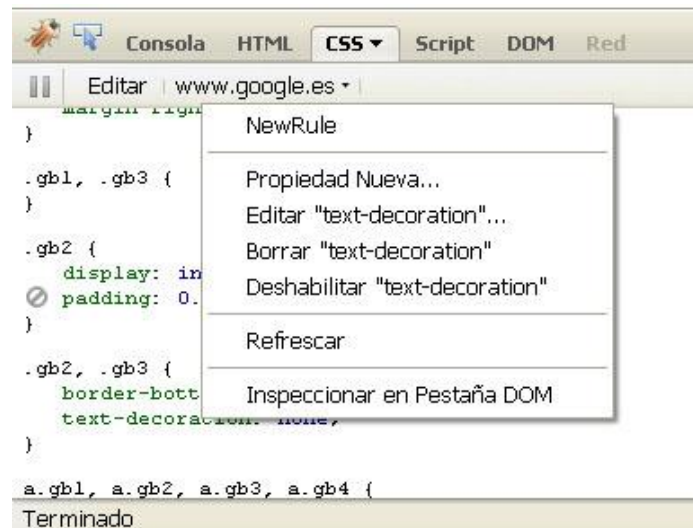
Otra funcionalidad interesante que nos proporciona Firebug es la posibilidad de cambiar el contenido de la página al vuelo mediante el botón editar de la esquina superior izquierda de la solapa Editar. Al pulsar sobre el botón editar con un elemento seleccionado podremos cambiar su contenido e iremos viendo cómo se van materializando estos cambios en la página Web.



La solapa CSS

En la solapa CSS podemos ver las propiedades de estilo CSS que se están utilizando en la página y, al igual que ocurría con el código HTML, podremos

cambiarlas al vuelo para ir probando como nos van quedando los distintos estilos que aplicamos. Para cambiar una propiedad únicamente tendremos que pulsar sobre ella y pasará al modo de edición que nos permitirá cambiar su valor. Además, si hacemos clic con el botón derecho del ratón sobre una regla CSS el menú contextual desplegado nos permitirá: añadir una nueva propiedad, editar o borrar la propiedad sobre la que hemos pulsado o añadir una nueva regla de estilo.

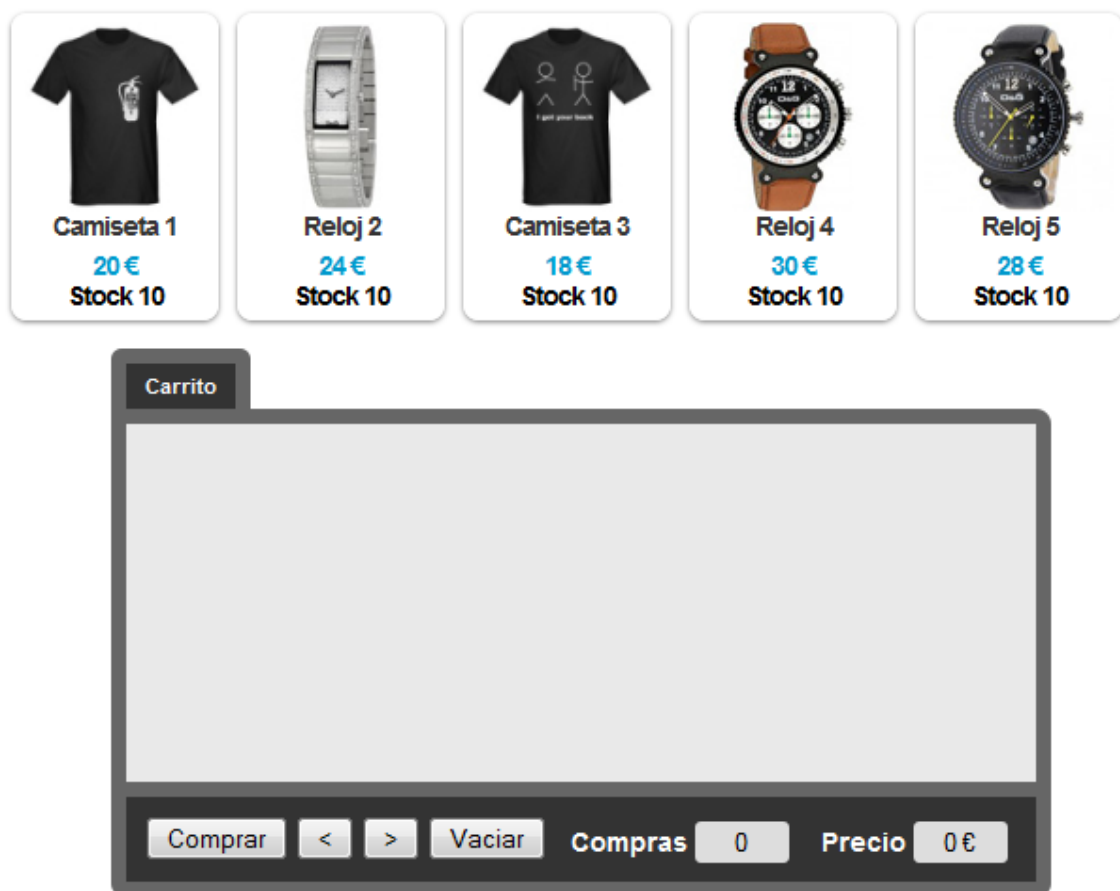


7. Metodología del curso

A lo largo del curso, realizaremos una pequeña aplicación Web utilizando la librería jQuery. Los ejercicios realizados en un tema, ampliarán los del tema anterior y servirán de base para realizar los del tema siguiente.

La aplicación Web que vamos a desarrollar es un carro de la compra, para lo cual, se proporciona un documento html (carro.html), una hoja de estilos (css/carro.css), un archivo javascript (script/carro.js) y una carpeta de imágenes (img). Estos ficheros se pueden encontrar dentro del archivo carrito.zip.

La apariencia que tiene la página es la siguiente:



Para poder realizar los ejercicios, necesitaremos conocer la estructura interna del documento carro.html.

La página está dividida en dos secciones:

- La sección de los artículos, que se encuentra en el <div> con id item_container.
- La sección del carrito, que se encuentra en el <div> con id cart_container.

Dentro de la sección de artículos, cada artículo se encuentra dentro de un `<div>` de la clase `item`, cuyo `id` será el código del artículo. Este `<div>` tendrá dentro los siguientes elementos:

- ✚ La imagen del artículo.
- ✚ El nombre del artículo, dentro de un `<label>` de la clase `title`.
- ✚ El precio del artículo, dentro de un `<label>` de la clase `price`.
- ✚ El stock disponible, dentro de un `<label>` de la clase `stock`.

El código html y el aspecto que tomarán los artículos será el siguiente:



```
<div class="item" id="i1">
  
  <label class="title">Reloj 1</label>
  <label class="price">20 €</label>
  <label class="stock">Stock 10</label>
</div>
```

Finalmente, antes de cerrar el `<div>` `item_container`, tenemos otro `div` de la clase `clear`, cuya misión es terminar con los elementos flotantes del `<div>`.

Por otra parte, el carrito contiene tres `<div>`:

- ✚ El título del carrito (`cart_title`).
- ✚ El contenedor de artículos comprados (`cart_toolbar`).
- ✚ La barra de navegación (`navigate`).

El código html es el siguiente:

```
<div id="cart_container">
  <div id="cart_title">
    <span>Carrito</span>
    <div class="clear"></div>
  </div>
  <div id="cart_toolbar">
    <div id="cart_items" class="back"></div>
  </div>
  <div id="navigate">
    <div id="nav_left">
      <button id="btn_comprar"
        title="Confirma la compra de los artículos">Comprar</button>
      <button id="btn_prev"
        title="Desplaza el carrito hacia la izquierda">&lt;</button>
      <button id="btn_next"
        title="Desplaza el carrito hacia la derecha">&gt;</button>
      <button id="btn_clear"
        title="Vacía el carrito">Vaciar</button>
    </div>
    <div id="nav_right">
      <span class="sptext">
        <label>Compras </label>
        <input id="citem" value="0" readonly="true"
          title="Número de productos comprados"/>
      </span>
      <span class="sptext">
        <label>Precio</label>
      </span>
    </div>
  </div>
</div>
```



```
        <input id="cprice" value="0 €" readonly="true"
            title="Precio total de los productos comprados"/>
    </span>
</div>
<div class="clear"></div>
</div>
</div>
```

Podemos ayudarnos del selector de elementos de Firebug, para ir viendo cómo se muestran los distintos elementos.

Los estilos necesarios se encuentran en la hoja de estilos `carro.css`. Este archivo no debemos modificarlo.

Inicialmente, no tenemos implementada ninguna funcionalidad, ya que, el fichero `carro.js` está vacío, pero a medida, que vayamos completando los temas del curso, iremos implementando las funcionalidades necesarias para que la aplicación funcione correctamente.

8. Ejercicios

En este primer tema, vamos a comenzar a trabajar con selectores, que como se ha comentado anteriormente, son fundamentales para el desarrollo de aplicaciones con jQuery.

Ejercicio 1

Añade la librería jQuery a la página `carro.html`. Debes utilizar el CDN de jQuery, tal y como, se ha visto en el ejemplo `documentoBasico2.html`.

Ejercicio 2






Añade al archivo `carro.js`, un método `document.ready` simplificado que llame a una función anónima. Puedes fijarte en el ejemplo `documentReadySimplificado.html`.

Ejercicio 3

Instala la herramienta Firebug y recorre los distintos elementos de la página `carro.html` con el inspeccionador de elementos (esto te ayudará a ir familiarizándote con la estructura de la página).

Ejercicio 4

Debes utilizar los selectores y el método `.css()` vistos en el tema, para resolver los siguientes ejercicios dentro del método `document.ready` simplificado del ejercicio anterior:

-  Cambia a gris (`#cecece`) el color de fondo (`background-color`) de los artículos (`items`).
-  Pon un borde (`border`) sólido, de color negro y 4 píxeles de grosor (`4px solid black`) al elemento con id `cart_items`.
-  Pon un borde (`border`) sólido, de color azul y 1 píxel de grosor (`1px solid blue`) a todas las imágenes de la página.
-  Subraya (`text-decoration:underline`) los `<label>` que sean hijos directos de un elemento de la clase `item`.
-  Pon la fuente de color rojo (`color:red`) a todos los botones (`<button>`) que estén dentro del `cart_container`.

- ✚ Pon la fuente de color blanco (`color:white`) a todos los `<label>` que estén adyacentes a otro `<label>` y estén dentro de un elemento de la clase `item`.
- ✚ Pon la fuente de color verde (`color:green`) a todos los elementos que contengan en el texto el símbolo '€' y a todos los `<input>` de la página.
- ✚ Pon el fondo de color amarillo a todos los `<div>` que estén vacíos.
- ✚ Pon el fondo de color rojo al primer y último elemento de la clase `item`.
- ✚ Pon el borde de color verde (`border-color:green`) a las imágenes de camisetas.

Resultado de los ejercicios

Tras resolver todos los ejercicios, la página Web debe verse de la siguiente forma:



9. Índice de ejemplos y tablas

Ejemplos

EJEMPLO 1: DOCUMENTOBASICO1.HTML.....	7
EJEMPLO 2: DOCUMENTOBASICO2.HTML.....	8
EJEMPLO 3: DOCUMENTREADY.HTML.....	10
EJEMPLO 4: DOCUMENTREADYSIMPLIFICADO.HTML.....	11
EJEMPLO 5: SELECTORUNIVERSAL.HTML	13
EJEMPLO 6: SELECTORETIQUETA.HTML	13
EJEMPLO 7: SELECTORID.HTML	14
EJEMPLO 8: SELECTORCLASE.HTML	14
EJEMPLO 9: SELECTORESCOMBINADOS.HTML.....	15
EJEMPLO 10: SELECTORESHIJOSDIRECTOS.HTML	16
EJEMPLO 11: SELECTORDESCENDIENTES.HTML.....	16
EJEMPLO 12: SELECTORESAGRUPADOS.HTML	17
EJEMPLO 13: SELECTORADYACENTE.HTML	18
EJEMPLO 14: FILTROSODDEVEN.HTML	20
EJEMPLO 15: FILTROSFIRSTLAST.HTML	21
EJEMPLO 16: FILTROHAS.HTML.....	21
EJEMPLO 17: FILTROEMPTY.HTML	22
EJEMPLO 18: FILTROCONTAINS.HTML	23
EJEMPLO 19: SELECTORATRIBUTOS1.HTML	24
EJEMPLO 20: SELECTORATRIBUTOS2.HTML	25

Tablas

TABLA 1: FILTROS JQUERY.....	19
TABLA 2: SELECTORES DE ATRIBUTOS.....	24