

www.adrformacion.com © ADRINFOR S.L.
Héctor García González

Selectores Básicos © ADRINFOR S.L.

www.adrformacion.com © ADRINFOR S.L.
Héctor García González

www.adrformacion.com © ADRINFOR S.L.
Héctor García González

Indice

Selectores Básicos	3
Introducción	3
¿Qué es un selector?	3
Comprobar una Selección	3
Método 1. Mensaje de alerta con el número de elementos	3
Método 2. Aplicar un atributo CSS a los elementos seleccionados	4
Método 3. Mostrar por consola el resultado de la selección	5
Selectores básicos: \$("elemento"), \$("#identificador") y \$(".clase")	6
Selector por tipo de elemento - \$("elemento")	6
Selector por identificador de elemento - \$("#identificador")	8
Selector por clase - \$(".clase")	10
Combinar selectores	12
Combinaciones básicas	12
Combinación de selector de tipo de elemento y de selector de clase	13
Combinación de selector por identificador y de selector de clase	13
Combinación de varios selectores de clase	14
Selectores múltiples - \$("selector1, selector2, selectorN")	15
Selectores Jerárquicos	17
Selector Hijo \$("div > table")	17
Selector Descendiente \$("div table")	19
Selector Contiguo \$("div + table")	21
Selector Próximo Hermano \$("div ~ table")	23
Guardar Selecciones	26
Uso de la función jQuery() con un contexto	26
Ejercicios	29
Ejercicio 1: Selectores Básicos	29
Lo necesario para comenzar	29
Ejercicio 2: Selectores Jerárquicos	29
Lo necesario para comenzar	30
Recursos	31
Enlaces de Interés	31

Selectores Básicos

Introducción

Una de las mayores diferencias entre los principiantes y los usuarios avanzados de jQuery es el dominio de los selectores.

¿Qué es un selector?

Un selector es básicamente una cadena de texto con la que indicar a la librería las características del conjunto de elementos sobre los que realizar diferentes métodos o acciones por medio de las utilidades incluidas en la librería.

Domina los selectores y dominarás el mundo... de jQuery!

La sintáxis utilizada por jQuery en sus selectores es muy similar a la sintáxis utilizada en las hojas de estilo CSS. Los selectores soportados por jQuery incluyen los definidos en el estándar CSS3, además de otros muchos propios de la librería.



Si has trabajado con hojas de estilo CSS, los selectores básicos de jQuery no te supondrán un dolor de cabeza. Practica con ellos y domínalos. A medida que te sientas mas cómodo con el uso de selectores, reducirás el tiempo necesario para encontrar el selector correcto. Además, todos los comandos jQuery requieren de esta selección, y que nuestro código funcione correctamente dependerá directamente de que los selectores sean lo suficientemente concisos para actuar únicamente con los elementos del DOM deseados.

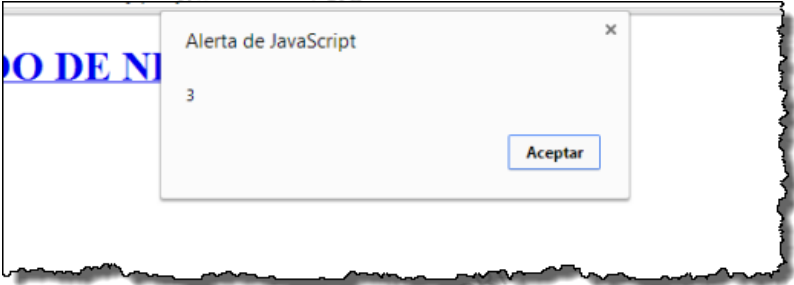
Comprobar una Selección

Cuando dispongamos de un selector el siguiente paso lógico es verificar el funcionamiento del mismo. Al utilizar la función `$(selector)`, jQuery devolverá como resultado un objeto de tipo jQuery, que contendrá los elementos del DOM que cumplan con las especificaciones indicadas por el selector.

Existen multitud de métodos con los que poder comprobar que el selector indicado es correcto. Vamos a ver tres métodos de ejemplo con los que analizar si el selector indicado en la llamada a la función localiza los elementos deseados.

Método 1. Mensaje de alerta con el número de elementos

Este es el método más simple de todos. Consiste en mostrar un mensaje de alerta con el número de elementos localizados por el selector, utilizando la propiedad length del objeto jQuery.

Mensaje de alerta con el número de elementos	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar una alerta con el numero de elementos devueltos por la seleccio n alert(\$("div").length); }); </script></pre>
Resultado	

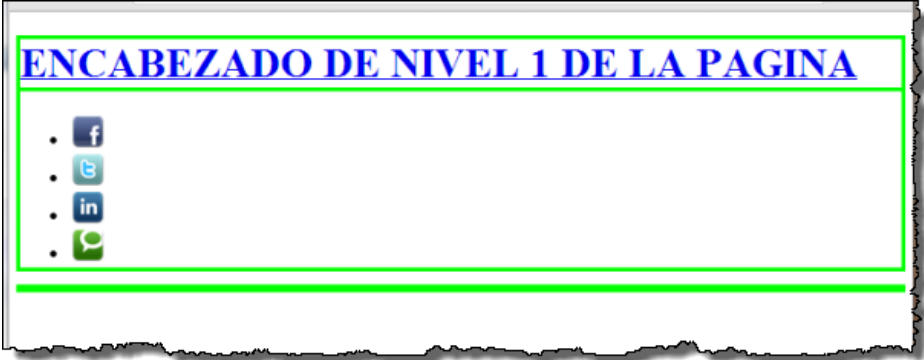
Del mismo modo, si necesitamos definir una condición que dependa de si un selector ha devuelto elementos o no, podríamos hacerlo analizando esta propiedad:

```
if($("p").length > 0){
  //acciones a realizar
}
```

Método 2. Aplicar un atributo CSS a los elementos seleccionados

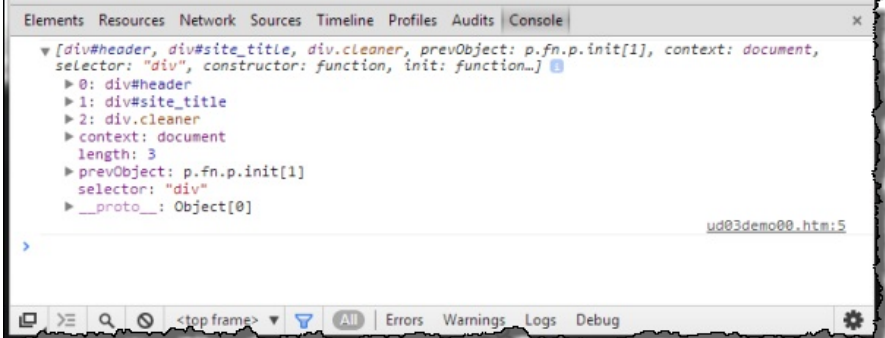
En este segundo método de comprobación de la selección, utilizamos el método `css` de jQuery para establecer una propiedad que modifique el aspecto visual de los elementos devueltos por el selector.

De este modo, podremos establecer una propiedad CSS a todos los elementos que aplique un fuerte contraste visual y así resaltar estos elementos del resto de elementos del DOM.

Aplicar un atributo CSS	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //aplicar una propiedad css a los elementos devueltos por la seleccion \$("div").css('outline','#5EFF00ss solid'); }); </script></pre>
Resultado	

Método 3. Mostrar por consola el resultado de la selección

Por último, este método será el que más información nos ofrecerá sobre el resultado de la selección. Este método consiste en enviar a la consola del navegador el resultado de la selección y, por medio de las utilidades de desarrollo ofrecidas por los navegadores, poder analizar el resultado obtenido.

Mostrar por consola el resultado de la selección	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola del navegador los elementos devueltos por la seleccion console.log(\$("div")); }); </script></pre>
Resultado de la selección	



Depurar nuestro código

Selectores básicos: \$("elemento"), \$("#identificador") y \$(".clase")

Como en cualquier aspecto de la vida, no vamos a comenzar a construir una casa por el tejado. Requerimos de unos cimientos estables, y, en los selectores, podríamos considerar los selectores básicos como la cimentación de nuestro conocimiento.

Cuando seleccionamos elementos con jQuery nuestro objetivo es ser tan específicos como sea necesario: buscamos encontrar un selector lo suficientemente conciso para devolver todo aquel elemento que queremos modificar.

Selector por tipo de elemento - \$("elemento")

El selector más básico (aunque no por ello menos importante) es el selector por tipo de objeto. Este selector permitirá seleccionar elementos del DOM cuyo tipo de elemento coincida con el indicado. Para definir un selector por tipo de elemento, simplemente pasaremos a la función \$ una cadena (entre comillas) con el tipo de elemento HTML que deseamos seleccionar.



Por ejemplo, podríamos definir estos dos selectores:

```
$("div")
```

```
$("span")
```

El primero de los selectores, seleccionará todos los elementos DIV de un documento. De la misma forma, el segundo selector devolverá todos los elementos de tipo SPAN del documento.

Podríamos definir diferentes selectores, uno para cada tipo de elemento HTML existente en nuestro documento:


```
$("p") // elementos de tipo párrafo <p></p>
```

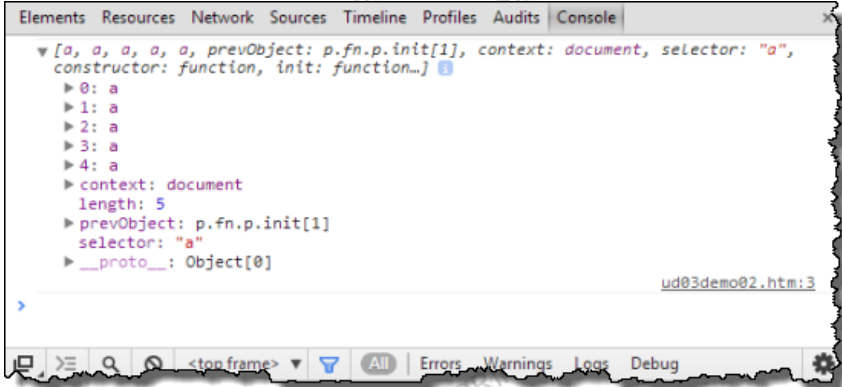
```
$("h1") // elementos de tipo H1 <h1></h1>
```

```
$("input") // elementos input de formularios <input />
```

Vamos a crear varios selectores, y ver el conjunto de elementos que se obtendrían para un fragmento de HTML. Supongamos que nuestra página tiene el siguiente código dentro de la etiqueta "body":

```
<div id="header">
<div id="site_title">
  <h1><a href="#"><strong>ENCABEZADO</strong> DE NIVEL 1<span> DE LA PAGINA</span></a></h1>
</div>
<ul id="social_box">
  <li class="social_item"><a href="#"></a></li>
  <li class="social_item"><a href="#"></a></li>
  <li class="social_item"><a href="#"></a></li>
  <li class="social_item"><a href="#"></a></li>
</ul>
<div class="cleaner"></div>
</div>
```

Selector \$("div")	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("div")); }); </script></pre>
Resultado de la selección	<p>Al acceder a la consola del navegador, observamos que el comando <code>console.log(\$("div"));</code> localiza tres elementos:</p> <ul style="list-style-type: none"> • El div <code><div id="header"></code> • El div <code><div id="site_title"></code> • El div <code><div class="cleaner" ></code> 

Selector \$("a")	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("a")); }); </script></pre>
Resultado de la selección	<p>Al acceder a la consola del navegador, observamos que el comando <code>console.log(\$("a"));</code> localiza cinco elementos:</p> <ul style="list-style-type: none"> • El a contenido el el encabezado h1 • Los cuatro a existentes en la lista no numerada 



Selector por tipo de elemento

Selector por identificador de elemento - \$("#identificador")

Uno de los atributos de los que puede disponer cualquier elemento HTML es el atributo "identificador" (id). Éste es un atributo que, de ser asignado a una etiqueta HTML, debe ser único: no debe existir ningún otro elemento dentro del documento con el mismo identificador.

Los selectores por identificador estarán compuestos por el símbolo # seguido del identificador del elemento que deseamos seleccionar.

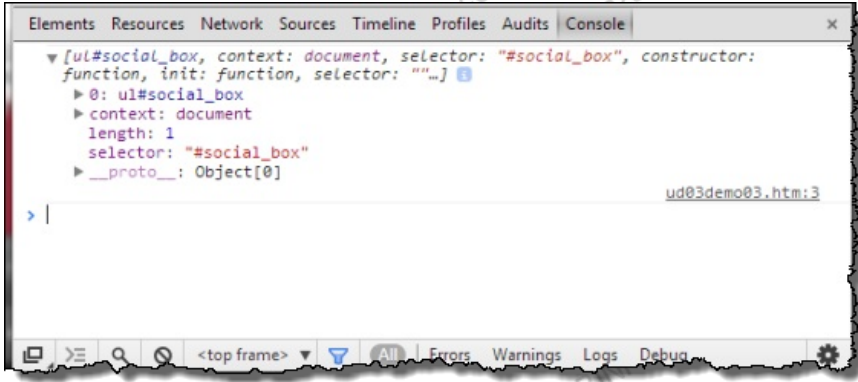


Vamos a continuar utilizando el mismo fragmento de código HTML de la página:

```
<div id="header">
<div id="site_title">
  <h1><a href="#"><strong>ENCABEZADO</strong> DE NIVEL 1<span> DE LA PAGINA</span></a></h1>
</div>
<ul id="social_box">
  <li class="social_item"><a href="#"></a></li>
  <li class="social_item"><a href="#"></a></li>
  <li class="social_item"><a href="#"></a></li>
  <li class="social_item"><a href="#"></a></li>
</ul>
<div class="cleaner"></div>
</div>
```

En este fragmento de HTML, localizamos tres elementos con identificador asignado: el "div" con el identificador header, el "div" con identificador site_title y el "ul" con identificador social_box.

Para seleccionar el ul con identificador social_box utilizaremos el selector \$("#social_box"):

Selector \$(" #social_box")	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$(" #social_box")); }); </script></pre>
Resultado de la selección	<p>Este selector, dado que busca los elementos cuyo atributo ID coincida con el texto indicado, solamente localizará un elemento y la propiedad length del objeto devuelto tendrá el valor 1 siempre y cuando el selector sea correcto.</p> <p>Al acceder a la consola del navegador, observamos que el comando <code>console.log(\$(" #social_box"));</code> localiza un único elemento:</p> <ul style="list-style-type: none"> El ul <code><ul id="social_box"></code> 

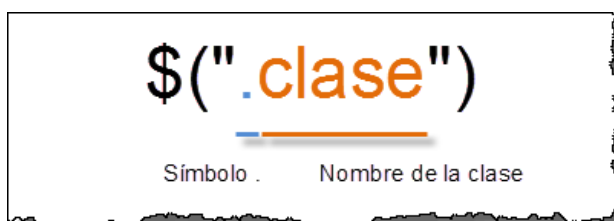
También podríamos seleccionar el div que contiene el encabezado h1 por medio del selector `$("#site_title")`.

Selector por clase - \$(".class")

Hasta ahora, hemos utilizado en nuestros selectores dos propiedades de las entidades HTML: una propiedad muy básica como es el tipo de elemento, y otra propiedad muy específica, el identificador.

Llegados a este punto, vamos a introducir otro selector con el que poder afinar ligeramente nuestra selección: los selectores por clase.

La estructura de un selector de clase se forma por el símbolo "." seguido del nombre de la clase.



Las clases son otros atributos de los elementos HTML. Una clase puede ser utilizada por varios elementos y, a su vez, un elemento puede tener asignada varias clases.

Ejemplo:

```
<table>
<tr class="impar principal"></tr>
<tr class="par"></tr>
<tr class="impar"></tr>
<tr class="par"></tr>
<tr class="impar final"></tr>
<tr class="par final"></tr>
</table>
```

Generalmente, este atributo - al igual que los anteriores - es utilizado por las hojas de estilo CSS, para aplicar unas reglas de diseño a los elementos de una página. En jQuery vamos a aprovechar estos atributos para hacer una selección de elementos de DOM que compartan una misma clase.

En el fragmento HTML anterior, podríamos utilizar los siguientes selectores:

`$(".impar")`

`$(".par")`

`$(".final")`

El primero de los selectores, filtraría aquellos elementos que tienen asignada la clase "impar". El segundo, agruparía los elementos sobre los que se ha aplicado la clase "par" y, por último, el tercer selector filtrará aquellos que dispongan de la clase "final".

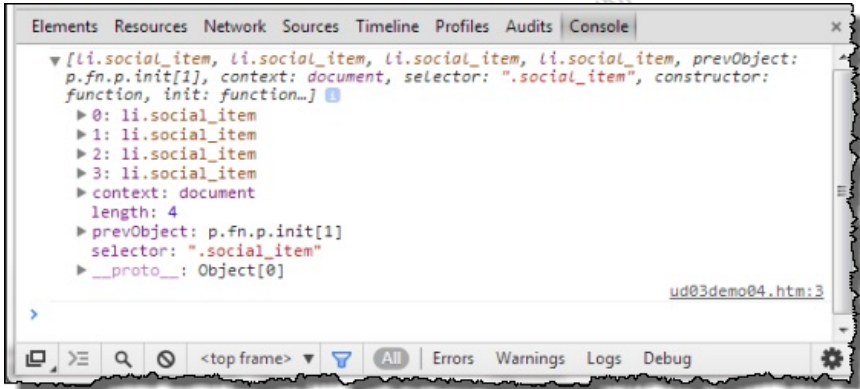
Volvamos una vez más al fragmento de código HTML que hemos utilizado en otras ocasiones:

```
<div id="header">
<div id="site_title">
  <h1><a href="#"><strong>ENCABEZADO</strong> DE NIVEL 1<span> DE LA PAGINA</span></a></h1>
</div>
<ul id="social_box">
  <li class="social_item"><a href="#"></a></li>
  <li class="social_item"><a href="#"></a></li>
  <li class="social_item"><a href="#"></a></li>
  <li class="social_item"><a href="#"></a></li>
</ul>
<div class="cleaner"></div>
</div>
```

Como se puede observar, cinco elementos de la página tienen asignado el atributo `class` :

- Cuatro elementos `li` con la clase `social_item`
- Un elemento `div` con la clase `cleaner`

Vamos a construir un selector de clase para seleccionar los cuatro elementos `li` que disponen de la clase `social_item`:

Selector \$("social_item")	
Código de ejemplo	<pre> <script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("social_item")); }); </script> </pre>
Resultado de la selección	<p>El selector indicado localizará entre los elementos del DOM aquellos que tengan asignada la clase social_item. Para nuestro fragmento de código, el objeto jQuery resultante incluirá los cuatro elementos li que tienen aplicada la clase social_item.</p> <p>Como resultado de la ejecución del comando console.log(\$("social_item")); , la consola del navegador nos mostrará el siguiente resultado:</p> 



Selectores Básicos



Selectores

Combinar selectores

Combinaciones básicas

Los tres tipos de selectores vistos hasta ahora nos permiten hacer selecciones con diferentes tipos de precisión:

- extremadamente específicas (selector por identificador)
- algo específicas (selector por clase)
- muy poco específicas (selector por elemento)

A su vez, los selectores pueden combinarse entre ellos para conseguir así mayor precisión a la hora de definir los elementos del DOM que seleccionarán.

Aunque no existe una obligación implícita en el orden de combinar selectores, aplicaremos, por convenio, la siguiente ordenación:

1. Selector por tipo de elemento
2. Selector por identificador
3. Selector por clase
4. Selector por atributo
5. Filtrados

Así, un selector "completo" sería `div#identificador.clase[atributo]:filtro()`. Este ejemplo es un poco "exagerado", pero sirve para hacernos una idea de la estructura de éstos.

Combinación de selector de tipo de elemento y de selector de clase



Selector <code>\$("li.social_item")</code>	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("li.social_item")); }); </script></pre>
Resultado de la selección	<p>Con este selector conseguiremos obtener únicamente los elementos <code>li</code> del documento que dispongan de la clase <code>social_item</code>. De este modo, se excluirán de la selección los elementos que no sean <code>li</code>, así como los <code>li</code> que no tengan asignada la clase.</p>

Combinación de selector por identificador y de selector de clase

\$("#identificador.clase")

Símbolo # ID del elemento Símbolo . Nombre de la clase

Selector \$("#site_title.destacado")	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("#site_title.destacado")); }); </script></pre>
Resultado de la selección	<p>La combinación del selector por identificador con el selector de clase nos permitirá filtrar un elemento por su identificador siempre y cuando tenga asignada la clase indicada. Así, si el elemento con el identificador indicado no posee la clase, no será incluido como resultado de la selección.</p>

Combinación de varios selectores de clase

\$(".identificador.clase")

Símbolo . Nombre de la clase 1 Símbolo . Nombre de la clase 2

Selector \$("impar.final")	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("impar.final")); }); </script></pre>
Resultado de la selección	Otro método de combinación es el uso de varios selectores de clase. De este modo, limitaremos la selección a elementos que tengan asignadas todas las clases indicadas.



Combinar selectores

Selectores múltiples - \$("selector1, selector2, selectorN")

Los selectores múltiples no son, propiamente dicho, un tipo diferente de selectores, sino más bien un método para obtener como resultado de la selección los elementos obtenidos por cada uno de los selectores indicados en la combinación.

Para construir un selector múltiple indicaremos tantos selectores como deseemos en una misma cadena separándolos por el carácter ",". El único requisito de los selectores indicados es que se encuentren correctamente definidos.



Analicemos un selector múltiple muy básico:

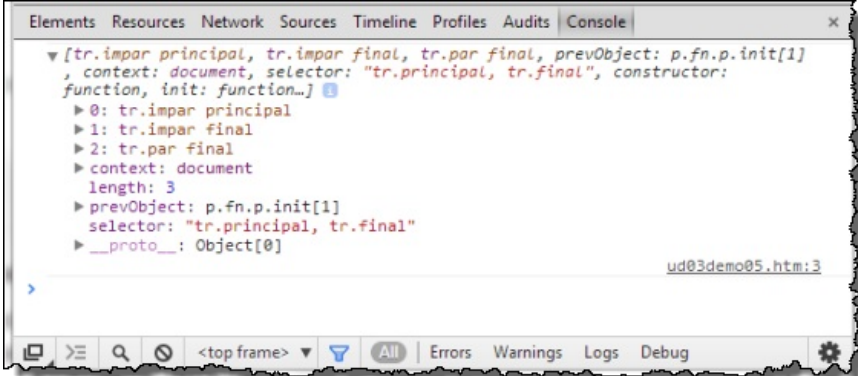
\$("div, p, span")

El selector anterior devolverá, como resultado de la selección, los elementos que cumplan con tres selectores separados por comas: Los elementos de tipo "div", los de tipo "p" y los de tipo "span" del DOM. Este selector es un ejemplo muy simple de uso de selectores múltiples. Sin embargo, la complejidad del selector múltiple podrá ser mucho mayor, dependiendo de los selectores que lo compongan.

Veamos un ejemplo de uso de selector múltiple frente al siguiente fragmento de HTML:

```
<table>
<tr class="impar principal"></tr>
<tr class="par"></tr>
<tr class="impar"></tr>
<tr class="par"></tr>
<tr class="impar final"></tr>
<tr class="par final"></tr>
</table>
```

Vamos a utilizar el selector `$("tr.principal, tr.final")`

Selector <code>\$("tr.principal, tr.final")</code>	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("tr.principal, tr.final")); }); </script></pre>
Resultado de la selección	<p>Como resultado de lanzar este comando, jQuery devolverá una selección en la que se combinarán los elementos de los dos selectores que forman la selección múltiple:</p> <ul style="list-style-type: none"> • El primer selector (<code>tr.principal</code>) seleccionará los elementos de tipo <code>tr</code> con la clase <code>principal</code> asignada • El segundo selector (<code>tr.final</code>) seleccionará los elementos de tipo <code>tr</code> con la clase <code>final</code> asignada <p>El resultado, mostrado en la consola del navegador, incluirá dos elementos: el primer <code>tr</code> de la tabla, que tenía asignada la clase <code>principal</code>, y el último, ya que tiene asignada la clase <code>final</code>.</p> 

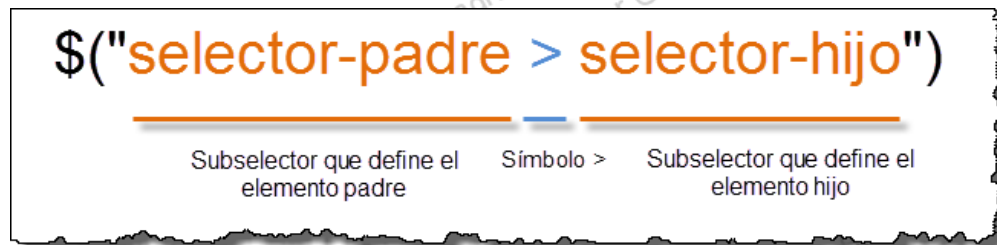
Selectores Jerárquicos

En ocasiones, es necesario acceder a ciertos elementos en relación al contexto en el que se encuentran o que tienen una relación con otros elementos del DOM. Los elementos del DOM son interpretados en forma de árbol, estableciendo una relación jerárquica entre ellos como elementos padre e hijos.

Con los selectores jerárquicos podremos definir condiciones que examinarán esta estructura, indicando la posición que deberán tener los elementos a seleccionar con relación a otros elementos del DOM.

Selector Hijo \$("div > table")

El Selector Hijo (o Child Selector) nos permitirá definir una selección que obtendrá como resultado los elementos que cumplan con el fragmento del selector situado a la derecha del símbolo ">" y cuyo elemento padre directo en el DOM cumpla, en su caso, con el fragmento del selector situado a la izquierda del símbolo anteriormente mencionado.





Selector Hijo I

Veamos un ejemplo sobre este fragmento de código HTML.

```
<div id="todo">
  <div class="parte" id="hijo1">
    <table id="tabla1"><tr><td></td></tr></table>
  </div>
  <div class="parte" id="hijo2">
    <div id="subhijo1">
      <table id="tabla2"><tr><td></td></tr></table>
    </div>
  </div>
  <div class="parte" id="hijo3">
    <span id="subhijo2">
      <table id="tabla3"><tr><td></td></tr></table>
    </span>
  </div>
</div>
```

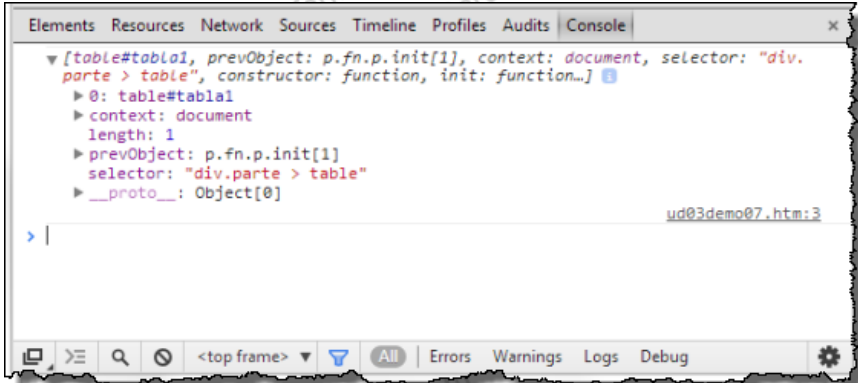
Supongamos que en este fragmento aplicamos el selector \$("div > table").

Selector \$("div > table")	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("div > table")); }); </script></pre>
Resultado de la selección	<p>Este selector solamente devolverá el conjunto con los elementos de tipo table cuyo padre directo sea un elemento div.</p> <p>La selección resultante contendrá dos elementos: la tabla #tabla1 que se encuentra dentro del div #hijo1 y la tabla #tabla2 que se encuentra dentro del div #subhijo1.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">  <p>Observa que la tabla #tabla3 no es incluida como resultado de la selección, ya que su padre directo es un elemento span.</p> </div> 



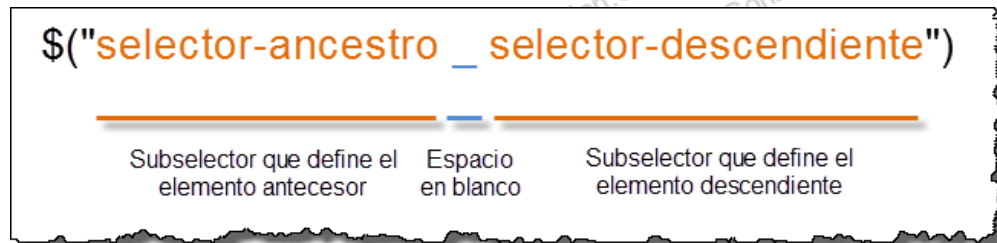
Selector Hijo II

Veamos qué resultado obtendríamos si utilizamos el selector `$("div.parte > table")` en nuestro código.

Selector <code>\$("div.parte > table")</code>	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("div.parte > table")); }); </script></pre>
Resultado de la selección	<p>A diferencia del selector anterior, éste también devolverá elementos de tipo table, pero solamente aquellos cuyo padre directo sea un elemento div con la clase "parte".</p> <p>En este caso, el resultado solamente incluirá un elementos: la tabla #tabla1, cuyo elemento padre es un elemento que cumple con el selector <code>div.parte</code>.</p> 

Selector Descendiente `$("div table")`

Con el Selector Descendiente podremos realizar consultas sobre el DOM y localizar los elementos descendientes de cualquier nivel que cumplan con el fragmento del selector situado a la derecha del símbolo " ", siendo elementos que se encuentren dentro del árbol del DOM de los elementos correspondientes al fragmento del selector previo al espacio .

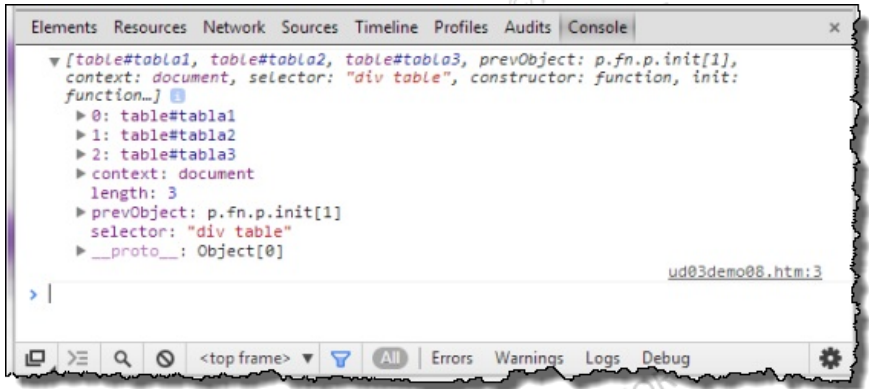


Este selector es menos restrictivo que el Selector Hijo, puesto que jQuery buscará sobre los elementos descendientes de cualquier nivel, mientras que el Selector Hijo solamente realizará la búsqueda de los descendientes de primer nivel (hijos directos).

Veamos un ejemplo sobre el mismo fragmento de HTML utilizado con el Selector Hijo.

```
<div id="todo">
  <div class="parte" id="hijo1">
    <table id="tabla1"><tr><td></td></tr></table>
  </div>
  <div class="parte" id="hijo2">
    <div id="subhijo1">
      <table id="tabla2"><tr><td></td></tr></table>
    </div>
  </div>
  <div class="parte" id="hijo3">
    <span id="subhijo2">
      <table id="tabla3"><tr><td></td></tr></table>
    </span>
  </div>
</div>
```

Vamos a modificar el selector `$("div > table")` - Selector Hijo - para formar el Selector Descendiente `$("div table")`

Selector \$("div table")	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("div table")); }); </script></pre>
Resultado de la selección	<p>Con el selector jerárquico Selector Descendiente, los elementos devueltos en el resultado serán elementos de tipo table que, en la estructura de árbol del DOM, contengan algún predecesor de tipo div.</p> <p>La selección contendrá tres elementos: la tabla #tabla1 que se encuentra dentro del div #hijo1, la tabla #tabla2 que se encuentra dentro del div #subhijo1 y la tabla #tabla3 que tiene entre sus predecesores el div #hijo3.</p>  <p>A diferencia del selector \$("div > table"), el resultado obtenido esta ocasión sí incluye la tabla #tabla3, puesto que uno de sus ancestros - aunque no sea el padre directo - es un elemento div.</p>



Selector Descendiente

Selector Contiguo \$("div + table")

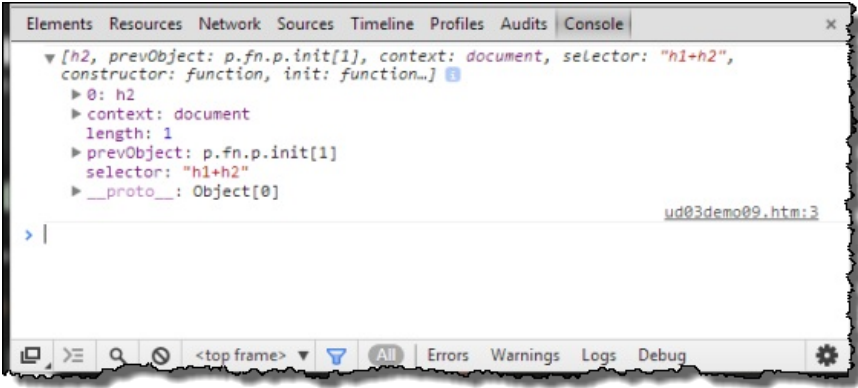

El Selector Contiguo permitirá definir un selector con el que localizar los elementos del DOM que cumplan con las especificaciones indicadas en el fragmento del selector situado a la derecha del símbolo "+" y que se encuentre inmediatamente precedido de un elemento que cumpla con la parte del selector a la izquierda del "+". Ambos elementos (el que cumple con las especificaciones del fragmento de la izquierda y el que cumple con las especificaciones del fragmento de la derecha) deben estar contenidos dentro de un elemento padre común.



Veamos un ejemplo básico de este selector. Supongamos que disponemos del siguiente fragmento de HTML:

```
<div id=\"todo\">
<h1>Encabezado de primer nivel</h1>
<h2>Primer Encabezado de segundo nivel</h2>
<p>Contenido del documento</p>
<p>Contenido del documento</p>
<h2>Segundo Encabezado de segundo nivel</h2>
<h3>Encabezado de primer nivel</h3>
<p>Contenido del documento</p>
<p>Contenido del documento</p>
</div>
```

El selector `$(\"h1 + h2\")` obtendrá los elementos h2 que se encuentran directamente precedidos de un elemento h1

Selector \$("h1 + h2")	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("h1 + h2")); }); </script></pre>
Resultado de la selección	<p>Con el selector \$("h1 + h2"), jQuery seleccionará únicamente los elementos h2 que están inmediatamente después de un elemento h1. En el caso que nos compete, solamente el elemento <h2>Primer Encabezado de segundo nivel</h2> , que se encuentra justo después del encabezado h1 será seleccionado.</p>  <div>  <p>Con este selector, el elemento<h2>Segundo Encabezado de segundo nivel</h2> no será seleccionado dado que el elemento que lo precede es de tipo p y no h1.</p> </div>

Selector Próximo Hermano \$("div ~ table")

El Selector Próximo Hermano funcionará de forma similar al Selector Contiguo, siendo este segundo menos restrictivo en lo referente a la proximidad de los elementos. Mientras que el Selector Contiguo busca elementos que se encuentren obligatoriamente el uno junto al otro, el Selector Próximo Hermano solamente requerirá que ambos elementos compartan su elemento padre y, el elemento especificado tras el símbolo "~" sea posterior al elemento que cumpla con el selector previo.



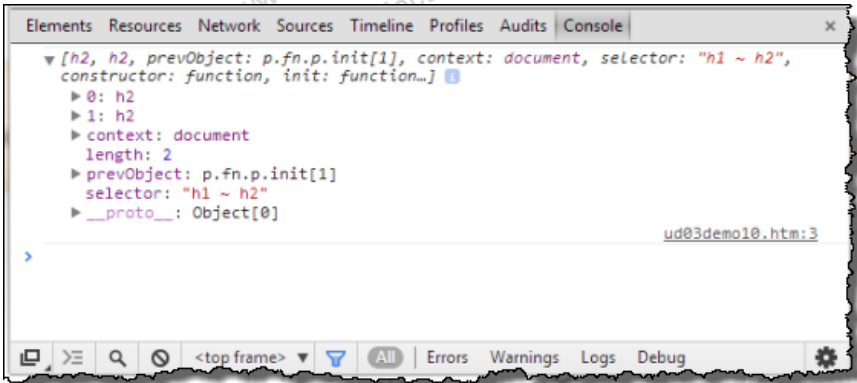
Para conseguir el símbolo ~ pulsaremos la tecla Alt y, sin soltarla, pulsaremos **en el teclado numérico 1 2 6**.



Veamos la diferencia de funcionamiento del Selector Próximo Hermano. Vamos a trabajar con el mismo fragmento de HTML sobre el que habíamos aplicado el Selector Contiguo:

```
<div id="todo">
<h1>Encabezado de primer nivel</h1>
<h2>Primer Encabezado de segundo nivel</h2>
<p>Contenido del documento</p>
<p>Contenido del documento</p>
<h2>Segundo Encabezado de segundo nivel</h2>
<h3>Encabezado de primer nivel</h3>
<p>Contenido del documento</p>
<p>Contenido del documento</p>
</div>
```

En este caso, el selector `$("h1 ~ h2")` obtendrá los elementos h2 posteriores a un elemento h1 que comparta su elemento padre.

Selector \$("h1 ~ h2")	
Código de ejemplo	<pre><script type="text/javascript"> \$(document).ready(function(){ //mostrar por consola los elementos que componen la seleccion console.log(\$("h1 ~ h2")); }); </script></pre>
Resultado de la selección	<p>Con el selector \$("h1 ~ h2"), jQuery seleccionará los elementos h2 posteriores a un elemento h1. En este caso se seleccionarán los dos elementos de tipo h2 que comparten padre con el elemento h1.</p>  <p>En este caso, el elemento <h2>Segundo Encabezado de segundo nivel</h2> también será seleccionado puesto que se encuentra dentro del mismo elemento padre y posteriormente al h1.</p>



Selectores contiguo y próximo hermano

En el siguiente enlace encontrarás una utilidad que te permitirá ver en vivo sobre una página web de demostración cómo actúan diferentes selectores, entre los que hay ejemplos de los vistos hasta ahora.



Banco de pruebas de selectores jQuery
<https://www.w3schools.com/jquery/trysele.asp>

Guardar Selecciones

Al lanzar una selección con jQuery, la librería realizará una gran cantidad de procesos, analizando los elementos del documento en busca de aquellos que cumplan con el selector indicado. Cuando un comando finaliza, el resultado de esta selección no es guardado para su posterior uso con lo que, si necesitamos reutilizar una selección, todos estos procesos deberán volver a ser realizados.

A pesar de este comportamiento podemos utilizar estrategias para evitar la pérdida de este resultado y aprovechar que la selección ya haya sido realizada, salvando la selección en una variable.

```
var $filasImpares = $('tr.impar');
```



La función jQuery() - o su alias \$() - devuelve un objeto que contiene los elementos resultantes de la selección. Este objeto puede ser asignado a una variable que será una instancia de la clase jQuery.

El símbolo \$ en el nombre de la variable no tiene ninguna funcionalidad especial. Actúa como un carácter más en el nombre de la variable, pero nos servirá para saber que la variable posee un objeto jQuery y diferenciarla de otras variables de otro tipo.

Esta notación es opcional, pero a su vez recomendable para acostumbrarnos al uso de buenas prácticas en nuestro código.

Disponiendo del resultado en la variable, podremos utilizar el resultado con los métodos de jQuery como si éstos fuesen ejecutados sobre la propia selección:

```
alert($filasImpares.length);  
//equivale a alert($('tr.impar').length)
```

```
$filasImpares.css('outline','ff0000 solid');  
//equivale a $('tr.impar').css('outline','ff0000 solid')
```



Si el DOM es modificado (se añaden, eliminan o modifican elementos) posteriormente al guardado de una selección, se necesitará repetir la selección y asignación a la variable para que los elementos contenidos por ésta se encuentren actualizados.

Uso de la función jQuery() con un contexto

Existe un uso diferente de la función jQuery() - o su alias \$() - que podrá aprovechar selecciones realizadas previamente.

Cada vez que llamamos a la función `jQuery()` con un selector como parámetro, ésta analiza el DOM entero en busca de elementos que cumplan con las condiciones indicadas en el selector. Dicho de otra forma, se utiliza el documento completo como contexto frente a la "consulta".

Otra forma de utilizar esta función es proporcionándole un parámetro adicional, posterior a la cadena indicadora del selector, que actuará como contexto alternativo y que hará que jQuery solamente analice los elementos resultantes del contexto y sus descendientes en busca de coincidencias, es decir, solamente analizará un árbol determinado del DOM.

Vamos a ver un ejemplo. Disponemos de este fragmento de HTML

```
<form id="formulario1">
  <label>Formulario 1</label>
  <input name="" type="checkbox" />
  <input name="" type="radio" />
  <input name="" type="text" />
  <input name="" type="button" />
</form>
<form id="formulario2">
  <label>Formulario 2</label>
  <input name="" type="checkbox" />
  <input name="" type="radio" />
  <input name="" type="text" />
  <input name="" type="button" />
</form>
```

Vamos a ver el comportamiento de tres fragmentos diferentes de código. En el primero de ellos, el comando jQuery no utilizará ningún contexto, mientras que en los dos siguientes sí se le proporcionará contexto de dos formas diferentes.

Este primer fragmento de código obtendrá todos los elementos de tipo "input" independientemente de dónde se encuentren éstos.

```
<script type="text/javascript">
$(document).ready(
function(){
  //mostrar por consola los elementos que componen la seleccion
  console.log( $("input") );
});
</script>
```

En este segundo fragmento de código, proporcionaremos un contexto a la función `$()` siendo este contexto el resultado previo guardado de una selección. De este modo, el comando solamente obtendrá los elementos de tipo "input" localizados en los elementos seleccionados anteriormente.

```
<script type="text/javascript">
$(document).ready(
function(){
  $formulario = $("#formulario1");
  //mostrar por consola los elementos que componen la seleccion
  console.log( $("input", $formulario) );
});
</script>
```


Por último, en este fragmento también se le indicará un contexto a la función \$() pero en este caso, se pasará a su vez como otra llamada a la función \$() sin necesidad de guardar el resultado de esta selección anteriormente en una variable. Al igual que el fragmento anterior, el comando obtendrá los elementos "input" del formulario #formulario1.

```
<script type="text/javascript">
$(document).ready(
function(){
    //mostrar por consola los elementos que componen la seleccion
    console.log( $("input", $("#formulario1")) );
});
</script>
```

Ejercicios

Ejercicio 1: Selectores Básicos

Duración estimada del ejercicio



15
minutos

Utiliza el documento "ejercicio-selectores.htm" y, por medio del interface de la página, construye los selectores para seleccionar:

1. Todos los elementos con la clase "rojo" asignada.
2. Todos los elementos de tipo "div"
3. Todos los elementos con las clases "rojo" y "fondogris"
4. El elemento con identificador "p3" y la clase "rojo"
5. El elemento con identificador "p3" y la clase "verde"

Lo necesario para comenzar

Descarga el archivo adjunto y extráelo en tu directorio de trabajo.




ejercicio-selectores-1.zip

Archivo comprimido con la plantilla para realizar el ejercicio

Ejercicio 2: Selectores Jerárquicos

Duración estimada del ejercicio



25
minutos

Utiliza el documento "ejercicio-selectores-jerarquicos.htm" y, por medio del interface de la página, construye los selectores para seleccionar:

1. Los elementos de tipo "div" que tienen asignadas la clase "bodyPreviewAlert" o la clase "footPreviewAlert"
2. Los elementos de tipo "div" que tienen asignadas la clase "bodyPreviewAlert" y la clase "sbPreviewAlert"

3. Los elementos de tipo "a" descendientes del "div" con identificador "userMensajes"
4. Los elementos con clase "sbPreviewAlert" hijos del "div" con identificador "userMensajes"
5. Los elementos con clase "footPreviewAlert", extendiendo el selector "#userMensajes > .headPreviewAlert..." (reemplaza los puntos suspensivos por la parte del selector que falta)
6. Los elementos de tipo "li" hijos de elementos "div" con clase "notificacionesDisplay"
7. Los elementos de tipo "li" con clase "messageContent" hijos de elementos "ul" con clase "mensajesBox"
8. Los elementos de tipo "a" con clase "readMore" hijos de un "li" que es el hermano siguiente de un "li" con la clase "messageContent"

Lo necesario para comenzar

Descarga el archivo adjunto y extráelo en tu directorio de trabajo.



Recursos

Enlaces de Interés



<https://www.w3schools.com/jquery/tryse1.asp>

<https://www.w3schools.com/jquery/tryse1.asp>

Utilidad de w3schools.com para ver frente a una página web qué elementos serían seleccionados por diferentes selectores predefinidos.