

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

Transversal © ADRINFOR S.L.

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

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

Indice

Transversal	3
Introducción	3
Acceso transversal a elementos del DOM	3
Acceso a elementos predecesores (parents)	3
Método .parent()	3
Método .parents()	8
Método .parentsUntil()	12
Método .offsetParent()	17
Método .closest()	17
Acceso a elementos descendientes (children)	20
Método .children()	20
Método .find()	22
Acceso a elementos hermanos (siblings)	27
Método .next()	27
Método .nextAll()	29
Método .nextUntil()	31
Método .prev()	33
Método .prevAll()	35
Método .prevUntil()	36
Método .siblings()	38
Métodos de filtrado	40
Método .eq()	40
Método .filter()	43
Método .filter() con selector como parámetro	43
Método .filter() con función como parámetro	45
Método .filter() con elemento como parámetro	48
Método .filter() con objeto jQuery como parámetro	50
Diferencias entre los métodos .find() y .filter()	52
Método .first()	53
Método .has()	55
Método .is()	59
Método .last()	63
Método .not()	65
Método .slice()	67
Iteraciones: .each y .map	69
Iteraciones con el método .each() o jQuery.each()	69
Método .each()	69
Método jQuery.each()	70
Iteraciones con el método .map() o jQuery.map()	70
Método .map()	70
Método jQuery.map()	70
Ejercicios	71
Ejercicio 1: Transversal I	71
Lo necesario para comenzar	71
Ejercicio 2: Transversal II	72
Lo necesario para comenzar	72

Transversal

Introducción

Una vez disponemos de un conjunto de elementos envuelto en un objeto jQuery, es habitual que nos surja la necesidad de acceder de manera más directa a elementos relacionados con los seleccionados previamente. Para ello disponemos de un conjunto de métodos transversales con los que modificar la selección sobre la que actuar.

Acceso transversal a elementos del DOM

Por medio de los métodos de acceso transversal de jQuery tendremos la posibilidad de "atravesar" los elementos del DOM de nuestra página (o dicho de otro modo, movernos a través de ellos). Este conjunto de métodos se aplicará sobre una selección inicial, sobre la que nos moveremos por el DOM de los elementos relativos a esta selección. A medida que utilicemos estos métodos para movernos por el DOM, la selección sobre la que se estén aplicando se verá alterada.

El acceso transversal puede dividirse en tres tipologías diferentes:

- Acceso a elementos predecesores (parents)
- Acceso a elementos descendientes (children)
- Acceso a elementos hermanos (siblings)

jQuery incluye un amplio conjunto de métodos de fácil uso para aprovechar al máximo las tres tipologías de acceso transversal entre los elementos del documento.

La mayoría de métodos podrán aceptar parámetros opcionales, como un selector o incluso un objeto de tipo jQuery, para conseguir una funcionalidad diferente. También encontraremos métodos que aceptarán parámetros de diferente tipo.



Este concepto es conocido en el mundo de la programación como "sobrecarga" de métodos (o funciones). Se refiere a la posibilidad de tener dos o más funciones con el mismo nombre pero funcionalidad diferente. Es decir, dos o más funciones con el mismo nombre realizan acciones diferentes.

Acceso a elementos predecesores (parents)

Método .parent()

El método .parent() obtendrá el elemento padre de cada uno de los elementos devueltos por el selector. Cuando utilicemos este método podremos proporcionar como parámetro adicional un selector, con el que filtrar los elementos padre de la selección previa.

\$(selector).parent()

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .parent() sin
ningún parámetro

HTML

Veamos un ejemplo de uso del método .parent(). Vamos a utilizarlo sobre el siguiente fragmento de código HTML:

```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
```

Vamos a aplicar el método .parent() sobre la selección \$("#site_title") para obtener el identificador del elemento padre que contiene este elemento.

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola el atributo "id" del elemento padre de la selección.
  console.log( $("#site_title") );
  console.log( $("#site_title").parent() );
  console.log( $("#site_title").parent().attr("id") );
});
</script>
```

Resultado

La selección principal, indicada por el fragmento de código \$("#site_title") contendrá el siguiente resultado:

```

▼ [div#site_title, context: document, selector: "#site_title", constructor: function, init:
function, selector: "..."] 0
  ▶ 0: div#site_title
  ▶ context: document
  ▶ length: 1
  ▶ selector: "#site_title"
  ▶ __proto__: Object[0]
ud05demo1.htm:3

▼ [div#header, prevObject: p.fn.p.init[1], context: document, selector:
"#site_title.parent()", constructor: function, init: function...] 0
  ▶ 0: div#header
  ▶ context: document
  ▶ length: 1
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "#site_title.parent()"
  ▶ __proto__: Object[0]
ud05demo1.htm:4
header
ud05demo1.htm:5
>

```

Al aplicar el método .parent() esta selección es modificada, siendo sustituida por el siguiente resultado:

```

▼ [div#site_title, context: document, selector: "#site_title", constructor: function, init:
function, selector: "..."] 0
  ▶ 0: div#site_title
  ▶ context: document
  ▶ length: 1
  ▶ selector: "#site_title"
  ▶ __proto__: Object[0]
ud05demo1.htm:3

▼ [div#header, prevObject: p.fn.p.init[1], context: document, selector:
"#site_title.parent()", constructor: function, init: function...] 0
  ▶ 0: div#header
  ▶ context: document
  ▶ length: 1
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "#site_title.parent()"
  ▶ __proto__: Object[0]
ud05demo1.htm:4
header
ud05demo1.htm:5
>

```

Finalmente, al aplicar el método .attr() con el parámetro "id", obtenemos el atributo "id" del primer elemento de la selección sobre la que se aplica.

```

▼ [div#site_title, context: document, selector: "#site_title", constructor: function, init:
function, selector: "..."] 0
  ▶ 0: div#site_title
  ▶ context: document
  ▶ length: 1
  ▶ selector: "#site_title"
  ▶ __proto__: Object[0]
ud05demo1.htm:3

▼ [div#header, prevObject: p.fn.p.init[1], context: document, selector:
"#site_title.parent()", constructor: function, init: function...] 0
  ▶ 0: div#header
  ▶ context: document
  ▶ length: 1
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "#site_title.parent()"
  ▶ __proto__: Object[0]
ud05demo1.htm:4
header
ud05demo1.htm:5
>

```

\$(selector).parent(selector)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.parent()

Selector opcional con el que
filtrar los elementos padre

HTML

Veamos otro ejemplo del método .parent() proporcionándole un parámetro adicional. Otra vez vamos a utilizar el mismo fragmento de HTML:

```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
```

Vamos a aplicar sobre la selección \$("li") el método .parent(), indicando además el parámetro ":first" para obtener el primero de los dos UL devueltos por el método.

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola el atributo "id" del elemento padre de la selección.
  console.log( $("li") );
  console.log( $("li").parent() );
  console.log( $("li").parent(":first") );
});
</script>
```

Resultado

La selección principal, indicada por el fragmento de código \$("li") contendrá todos los elementos de tipo "li" de la página:

```

Elements Resources Network Sources Timeline Profiles Audits Console
▼ [li, li, li, li, li, li, li, prevObject: p.fn.p.init[1], context: document, selector: "li", constructor: function, init: function...]
  ► 0: li
  ► 1: li
  ► 2: li
  ► 3: li
  ► 4: li
  ► 5: li
  ► 6: li
  ► context: document
  ► length: 7
  ► prevObject: p.fn.p.init[1]
  ► selector: "li"
  ► __proto__: Object[0]
  ud05demo2.htm:3

▼ [ul#lang_links, ul#social_box, prevObject: p.fn.p.init[7], context: document, selector: "li.parent()", constructor: function, init: function...]
  ► 0: ul#lang_links
  ► 1: ul#social_box
  ► context: document
  ud05demo2.htm:4
  
```

Al aplicar el método .parent() la selección se modificad, pasando a contener los dos elementos "ul" padres de los "li":

```

▼ [ul#lang_links, ul#social_box, prevObject: p.fn.p.init[7], context: document, selector: "li.parent()", constructor: function, init: function...]
  ► 0: ul#lang_links
  ► 1: ul#social_box
  ► context: document
  ► length: 2
  ► prevObject: p.fn.p.init[7]
  ► selector: "li.parent()"
  ► __proto__: Object[0]
  ud05demo2.htm:3

▼ [ul#lang_links, prevObject: p.fn.p.init[7], context: document, selector: "li.parent(:first)", constructor: function, init: function...]
  ► 0: ul#lang_links
  ► context: document
  ud05demo2.htm:4
  
```

Por último, si al método .parent() le proporcionamos el selector ":first", el resultado únicamente contendrá el primero de estos "ul".

```

▼ [ul#lang_links, prevObject: p.fn.p.init[7], context: document, selector: "li.parent(:first)", constructor: function, init: function...]
  ► 0: ul#lang_links
  ► context: document
  ► length: 1
  ► prevObject: p.fn.p.init[7]
  ► selector: "li.parent(:first)"
  ► __proto__: Object[0]
  ud05demo2.htm:5
  
```



Método .parent()

Método .parents()

El método .parents() modificará la selección sobre la que se aplique, devolviendo un nuevo resultado compuesto por todos los ancestros de cada uno de los elementos obtenidos en la selección previa.

\$(selector).parents()

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .parents() sin
ningún parámetro

HTML

Veamos otro ejemplo del método .parents() frente a nuestro habitual fragmento de código HTML:

```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
```

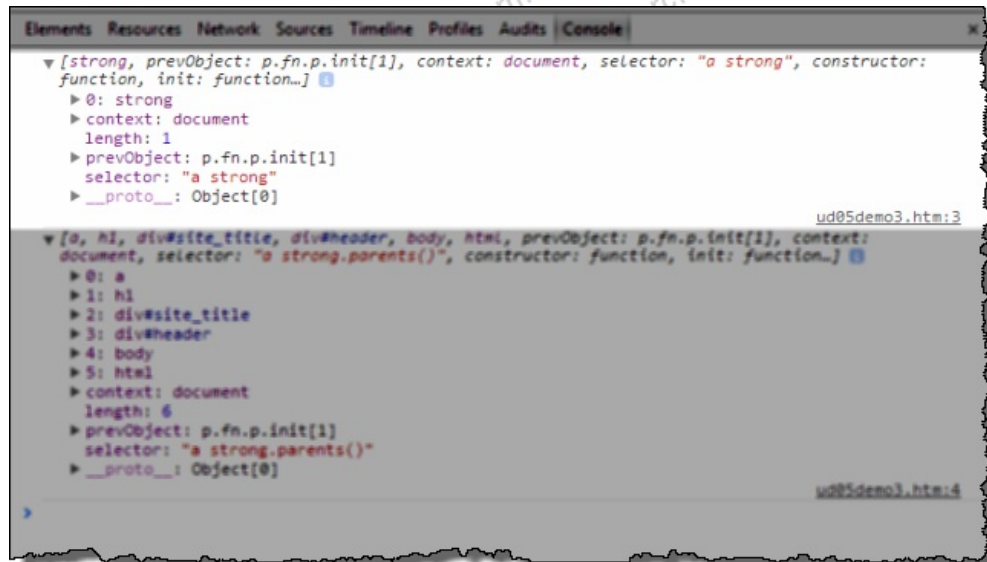
Vamos a aplicar sobre la selección \$("a strong") el método .parents(), para analizar los elementos contenidos en el resultado.

Código de Ejemplo

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


Resultado

El selector sobre el que se aplica el método contiene el único elemento "strong" descendiente de un "a":



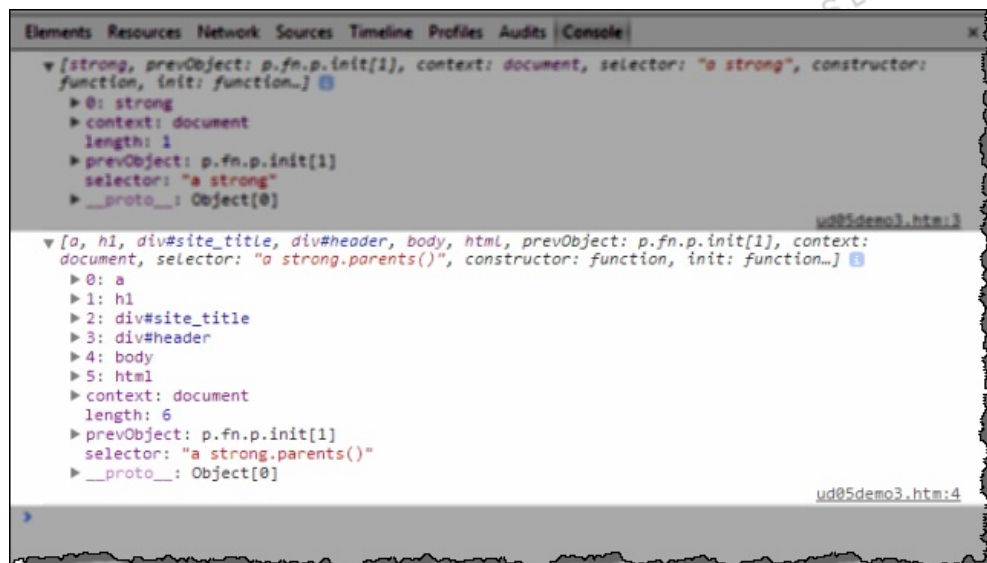
```

Elements Resources Network Sources Timeline Profiles Audits Console
▼ [strong, prevObject: p.fn.p.init[1], context: document, selector: "a strong", constructor: function, init: function...]
  ► 0: strong
  ► context: document
  ► length: 1
  ► prevObject: p.fn.p.init[1]
  ► selector: "a strong"
  ► __proto__: Object[0]
ud05demo3.htm:3

▼ [a, h1, div#site_title, div#header, body, html, prevObject: p.fn.p.init[1], context: document, selector: "a strong.parents()", constructor: function, init: function...]
  ► 0: a
  ► 1: h1
  ► 2: div#site_title
  ► 3: div#header
  ► 4: body
  ► 5: html
  ► context: document
  ► length: 6
  ► prevObject: p.fn.p.init[1]
  ► selector: "a strong.parents()"
  ► __proto__: Object[0]
ud05demo3.htm:4

```

Al llamar al método `.parents()` sobre la anterior selección, obtenemos como resultado un objeto jQuery que contiene todos los predecesores del elemento de la anterior selección, ordenados de más próximo a más cercano (invirtiendo el árbol del DOM):



```

Elements Resources Network Sources Timeline Profiles Audits Console
▼ [strong, prevObject: p.fn.p.init[1], context: document, selector: "a strong", constructor: function, init: function...]
  ► 0: strong
  ► context: document
  ► length: 1
  ► prevObject: p.fn.p.init[1]
  ► selector: "a strong"
  ► __proto__: Object[0]
ud05demo3.htm:3

▼ [a, h1, div#site_title, div#header, body, html, prevObject: p.fn.p.init[1], context: document, selector: "a strong.parents()", constructor: function, init: function...]
  ► 0: a
  ► 1: h1
  ► 2: div#site_title
  ► 3: div#header
  ► 4: body
  ► 5: html
  ► context: document
  ► length: 6
  ► prevObject: p.fn.p.init[1]
  ► selector: "a strong.parents()"
  ► __proto__: Object[0]
ud05demo3.htm:4

```

Este método también podrá recibir un selector como parámetro adicional, que se aplicará sobre el nuevo resultado y filtrará los elementos resultantes con aquellos que cumplan con las condiciones de este selector.

\$(selector).parents(selector)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.parents()

Selector opcional con el que filtrar
los elementos predecesores

HTML

Veamos otro ejemplo del método .parents() frente a nuestro habitual fragmento de código HTML:

```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></
a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
```

Vamos a aplicar sobre la selección \$("a strong") el método .parents(), para analizar los elementos contenidos en el resultado.

Código de Ejemplo

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

Resultado

El selector sobre el que se aplica el método contiene el único elemento "strong" descendiente de un "a":

```

Elements Resources Network Sources Timeline Profiles Audits Console
▼ [strong, prevObject: p.fn.p.init[1], context: document, selector: "a strong", constructor:
function, init: function...]
  ▶ 0: strong
  ▶ context: document
  ▶ length: 1
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "a strong"
  ▶ __proto__: Object[0]
ud05demo4.htm:3

▼ [div#site_title, div#header, prevObject: p.fn.p.init[1], context: document, selector: "a str
ong.parents(div)", constructor: function, init: function...]
  ▶ 0: div#site_title
  ▶ 1: div#header
  ▶ context: document
  ▶ length: 2
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "a strong.parents(div)"
  ▶ __proto__: Object[0]
ud05demo4.htm:4
> |
  
```

En esta ocasión, la llamada al método `parents()` se realiza con el parámetro "div", que actuará como selector filtrando el resultado previo, lo que reducirá el conjunto de elementos predecesores a aquellos que son elementos de tipo "div". Del mismo modo que en el anterior ejemplo, estos resultados se ordenarán en forma inversa a la estructura existente en el DOM:

```

Elements Resources Network Sources Timeline Profiles Audits Console
▼ [strong, prevObject: p.fn.p.init[1], context: document, selector: "a strong", constructor:
function, init: function...]
  ▶ 0: strong
  ▶ context: document
  ▶ length: 1
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "a strong"
  ▶ __proto__: Object[0]
ud05demo4.htm:3

▼ [div#site_title, div#header, prevObject: p.fn.p.init[1], context: document, selector: "a str
ong.parents(div)", constructor: function, init: function...]
  ▶ 0: div#site_title
  ▶ 1: div#header
  ▶ context: document
  ▶ length: 2
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "a strong.parents(div)"
  ▶ __proto__: Object[0]
ud05demo4.htm:4
> |
  
```



Método .parents()



Los métodos `.parent()` y `.parents()` pueden crear confusión por su similitud. Su diferencia radica en que el método `.parents()` "escalará" en el DOM y obtendrá todos los predecesores de cada elemento, mientras que el método `.parent()` únicamente devolverá el elemento padre (primer antecesor) de cada elemento.

Método .parentsUntil()

Este método obtendrá todos los antecesores de cada elemento devuelto por la selección sobre la que se aplique hasta alcanzar un elemento indicado como parámetro. En la selección, el predecesor límite no será incluido como resultado.

Este método puede utilizarse de dos formas diferentes:

1. Proporcionándole como parámetro un selector que defina el elemento límite hasta el que ascender en el DOM.
2. Un nodo del DOM, que al igual que ocurría con el selector, establecerá el elemento límite hasta el que incluir.

En ambos casos, podremos indicar opcionalmente un segundo parámetro que filtrará el conjunto de elementos que devolverá el método.

`$(selector).parentsUntil(selector, filtro)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.parentsUntil()`

Selector que
especifica el
elemento límite

Filtro opcional que
filtrará los elementos
a devolver

HTML

Volvamos al código HTML visto en los ejemplos anteriores

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>

```

En este caso, a la selección \$("a strong") le vamos a aplicar el método .parentsUntil() con el parámetro "div[id]" para que actúe como selector y establezca como elemento límite el primer div con atributo ID. De este modo, la selección final contendrá a todos los predecesores de la selección anterior hasta que se localice un predecesor que, siendo un elemento de tipo "DIV", disponga del atributo "ID" establecido.

Para ver cómo modifica el resultado el segundo parámetro, realizaremos la misma llamada indicando en este segundo parámetro el selector "h1", para filtrar el resultado y solamente incluir aquellos que sean enabezados de primer nivel.

Código de Ejemplo

```

<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola los elementos que componen la seleccion
  console.log( $("a strong") );
  console.log( $("a strong").parentsUntil("div[id]") );
  console.log( $("a strong").parentsUntil("div[id]","h1") );
});
</script>

```

Resultado

El selector sobre el que se aplica el método contiene el único elemento "strong" descendiente de un "a":

```

Elements Resources Network Sources Timeline Profiles Audits Console
▼ [strong, prevObject: p.fn.p.init[1], context: document, selector: "a strong", constructor: function, init: function...]
  ▶ 0: strong
  ▶ context: document
  ▶ length: 1
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "a strong"
  ▶ __proto__: Object[0]
  ud05demo5.htm:3

▼ [a, h1, prevObject: p.fn.p.init[1], context: document, selector: "a strong.parentsUntil(div[id])", constructor: function, init: function...]
  ▶ 0: a
  ▶ 1: h1
  ▶ context: document
  ▶ length: 2
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "a strong.parentsUntil(div[id])"
  ▶ __proto__: Object[0]
  ud05demo5.htm:4

```

Como comentamos anteriormente, el método .parentsUntil() localizará todos los predecesores del elemento seleccionado, y el parámetro ("div[id]") limitará éstos solamente hasta el primero de ellos de tipo "div" que posea el atributo "id" asignado:

```

Elements Resources Network Sources Timeline Profiles Audits Console
▼ [strong, prevObject: p.fn.p.init[1], context: document, selector: "a strong", constructor: function, init: function...]
  ▶ 0: strong
  ▶ context: document
  ▶ length: 1
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "a strong"
  ▶ __proto__: Object[0]
  ud05demo5.htm:3

▼ [a, h1, prevObject: p.fn.p.init[1], context: document, selector: "a strong.parentsUntil(div[id])", constructor: function, init: function...]
  ▶ 0: a
  ▶ 1: h1
  ▶ context: document
  ▶ length: 2
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "a strong.parentsUntil(div[id])"
  ▶ __proto__: Object[0]
  ud05demo5.htm:4

```

Si observamos el resultado tras aplicarle el segundo parámetro, el conjunto resultante solamente contendrá un elemento, el predecesor de tipo "H1".

```

selector: "a strong"
  ▶ __proto__: Object[0]

▼ [a, h1, prevObject: p.fn.p.init[1], context: document, selector: "a strong.parentsUntil(div[id])", constructor: function, init: function...] 3
  ▶ 0: a
  ▶ 1: h1
  ▶ context: document
  ▶ length: 2
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "a strong.parentsUntil(div[id])"
  ▶ __proto__: Object[0]

▼ [h1, prevObject: p.fn.p.init[1], context: document, selector: "a strong.parentsUntil(div[id],h1)", constructor: function, init: function...] 3
  ▶ 0: h1
  ▶ context: document
  ▶ length: 1
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "a strong.parentsUntil(div[id],h1)"
  ▶ __proto__: Object[0]

```

\$(selector).parentsUntil(elemento, filtro)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .parentsUntil()

Elemento u objeto jQuery
que especifica el
elemento límite

Filtro opcional que
filtrará los elementos
a devolver

HTML

Veamos el mismo efecto utilizando un elemento del DOM como parámetro limitador. Una vez más, utilizamos nuestro habitual fragmento de HTML:

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>

```

En este caso vamos a llamar al método .parentsUntil() con un elemento que actuará como elemento límite en el recorrido ascendente de obtención de predecesores.

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola los elementos que componen la seleccion
  var elementoLimite = document.getElementById("site_title");
  //la variable elementoLimite hará referencia al objeto del DOM div#site_title
  console.log( $("a strong").parentsUntil(elementoLimite) );
});

</script>
```

Resultado

El resultado de este comando contendrá los elementos padres del elemento "strong" hasta llegar al div#site_title, referenciado por la variable elementoLimite tras llamar al método nativo de JavaScript document.getElementById().



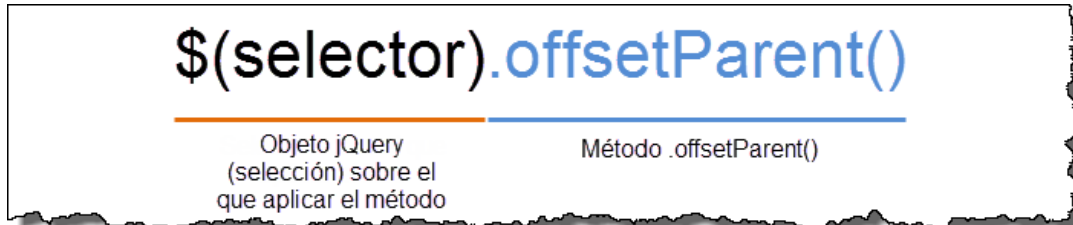
Si no se localizase ningún elemento entre los predecesores que cumpla con el límite establecido, el funcionamiento del método será idéntico a realizar una llamada al método .parents()



Método .parentsUntil()

Método .offsetParent()

El método .offsetParent() obtendrá el elemento ancestro que posea el atributo de CSS "position" con uno de los valores "absolute", "relative" o "fixed".



Método .closest()

El método .closest() obtendrá, para cada elemento de la selección sobre la que se aplica, el elemento más próximo ascendentemente en la estructura del DOM que cumpla con el selector proporcionado como parámetro al método.

Aunque su funcionamiento es similar al método .parent() existen pequeñas diferencias entre ambos métodos

Método .closest()	Método .parent()
Iniciará el análisis en los elementos sobre los que se aplica el método	Iniciará el análisis en los padres directos de los elementos sobre los que se aplica el método
Realizará el análisis hasta localizar un elemento que coincida con el selector	Analizará todos los predecesores e incluirá en el resultado todos aquellos que cumplan con las especificaciones del selector
El objeto devuelto contendrá, como máximo, un elemento por cada elemento incluido en la selección previa	El objeto devuelto podrá obtener varios o ningún elemento por cada elemento incluido en la selección previa



Al llamar a este método podremos indicar, además del selector que se encargará de especificar el elemento próximo a obtener, un objeto jQuery opcional que especificará el contexto sobre el que localizar el elemento próximo.

HTML

Veamos el funcionamiento de este método sobre este fragmento de HTML:

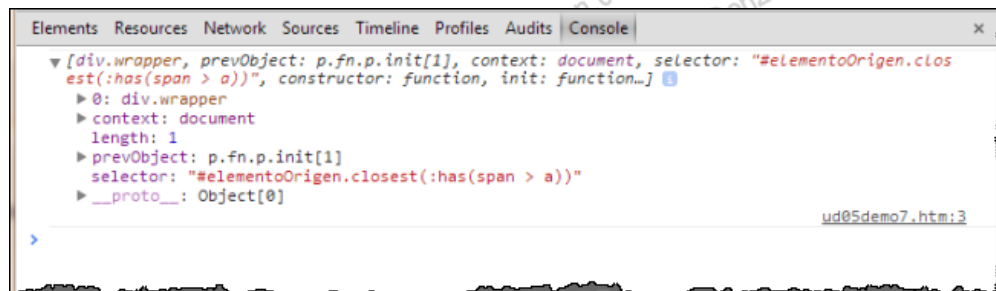
```
<div class="wrapper">
  <h1>
    <span>Encabezado de la p&acute;gina <a href="#">Nombre P&acute;gina</a></span>
  </h1>
</div>
<img />
<span id="elementoOrigen"></span>
</div>
</div>
```

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola los elementos que componen la seleccion
  console.log( $("#elementoOrigen").closest(":has(span > a)") );
});
</script>
```

Resultado

Este comando modificará la selección inicial (que contendrá el span#elementoOrigen), obteniendo el primer predecesor que contiene un span seguido de un a.



\$(selector).closest(elemento)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .closest()

Objeto jQuery o
elemento del DOM
próximo a obtener

HTML

Otra forma de llamar a este método es utilizarlo con un elemento del DOM como parámetro, o un elemento jQuery. Vamos a ver estas dos formas de llamar al método y cómo conseguir el mismo resultado con diferente tipo de parámetro. Veámoslo el HTML anterior ligeramente modificado:

```
<div id="mywrapper" class="wrapper">
  <h1>
    <span>Encabezado de la p&acute;gina <a href="#">Nombre P&acute;gina</a></span>
  </h1>
</div>
<img />
<span id="elementoOrigen"></span>
</div>
</div>
```

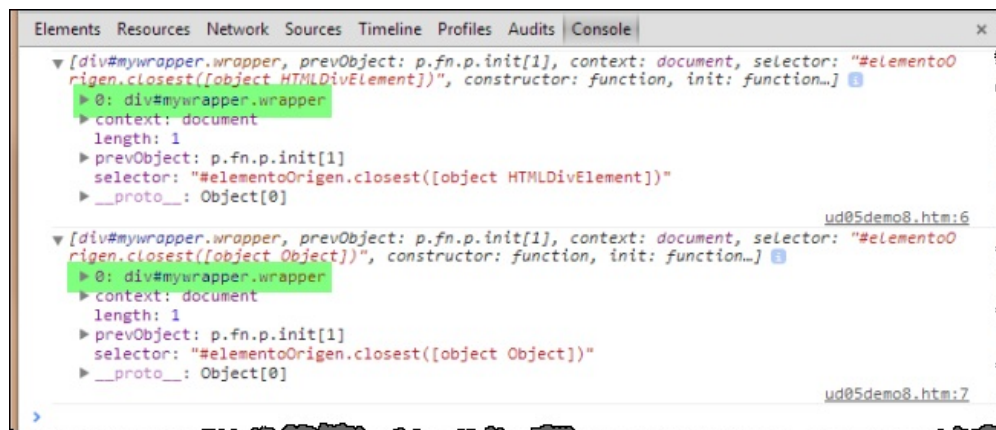
Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  elementoLocalizar=document.getElementById('mywrapper');
  $objjQueryLocalizador=$("#mywrapper");
  //mostrar por consola los elementos que componen la seleccion
  console.log( $("#elementoOrigen").closest( elementoLocalizar ) );
  console.log( $("#elementoOrigen").closest( $objjQueryLocalizador ) );
});

</script>
```

Resultado

Como resultado, los dos comandos seleccionarán el mismo elemento. La diferencia entre ellos radica en el parámetro recibido, siendo en uno de ellos un elemento del DOM y en el otro un objeto jQuery, resultado de otra selección previa.





Método .closest()

Acceso a elementos descendientes (children)

Método .children()

Con el método .children() podremos obtener los elementos descendientes de primer nivel (elementos hijo) pertenecientes a los elementos incluidos en el objeto jQuery sobre el que se aplique el método.

`$(selector).children(selector)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .children()

Selector opcional que
especificará los hijos a
devolver



Es importante tener en cuenta que el método .children() no devolverá entre los resultados elementos de tipo nodo de texto.

Al llamar al método .children() podremos, opcionalmente, proporcionarle un selector como parámetro con el que filtrar el conjunto de elementos resultante.

HTML

Veamos el funcionamiento de este método. Vamos a utilizar nuestro habitual fragmento de HTML:

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>

```

En este caso vamos a llamar al método .children() sin ningun parámetro.

Código de Ejemplo

```

<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola los elementos que componen la seleccion
  console.log( $("h1 > a").children( ) );
  console.log( $("h1 > a").children( "strong" ) );
});
</script>

```

Resultado

Como resultado de la primera llamada, la selección contendrá los elementos strong y span existentes dentro del elemento a. Observa que, el nodo de texto que formaba la cadena "BOX" no es incluido en la selección.



```

[strong, span, prevObject: p.fn.p.init[1], context: document, selector: "h1 > a.children()"], constructor: function, init: function...
  0: strong
  1: span
  context: document
  length: 2
  prevObject: p.fn.p.init[1]
  selector: "h1 > a.children()"
  __proto__: Object[0]
  
```

Por otro lado, la segunda llamada filtrará los resultados, manteniendo en la selección solamente aquellos de tipo "strong". Así, el número de elementos se ve reducido a uno.



```

[strong, prevObject: p.fn.p.init[1], context: document, selector: "h1 > a.children(strong)"], constructor: function, init: function...
  0: strong
  context: document
  length: 1
  prevObject: p.fn.p.init[1]
  selector: "h1 > a.children(strong)"
  __proto__: Object[0]
  
```



Método .children()

Método .find()

El método .find() nos permitirá obtener, a partir de los elementos devueltos por la selección sobre la que se aplique, un nuevo conjunto compuesto por los descendientes de éstos que cumplan con el selector proporcionado como parámetro al método.

\$(selector).find(selector)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.find()

Selector que especificará
los descendientes a
devolver

HTML

Vamos a aplicar el método .find() a una selección que realizaremos sobre el siguiente fragmento de HTML:

```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>
```

El método .find() buscará en todos los descendientes, así que podremos descender a lo largo del DOM de los elementos incluidos en la selección sobre la que se aplique.

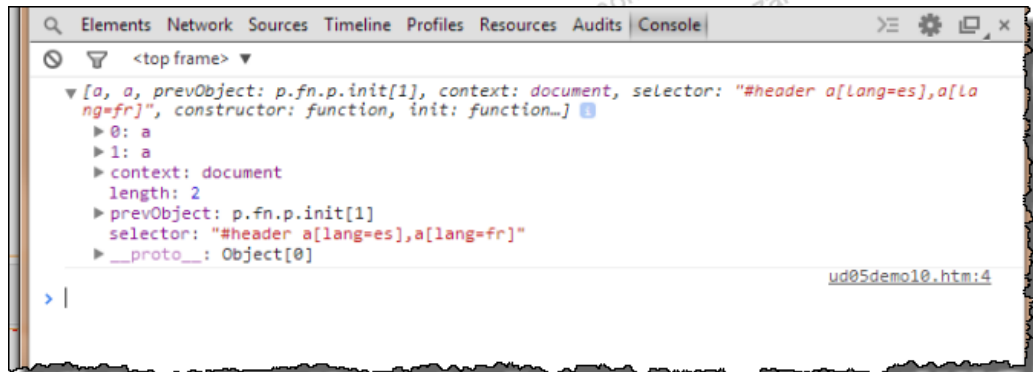
Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola los elementos que componen la seleccion
  $seleccion = $("#header");
  console.log( $seleccion.find("a[lang=es],a[lang=fr]") );
});
</script>
```

Resultado

Inicialmente, guardamos el resultado de realizar la selección `#header` en la variable `$seleccion`. Esta selección contendrá únicamente el elemento `div#header`.

Al aplicar sobre esta selección el método `.find()` realizaremos una búsqueda de todos sus descendientes y, dado que indicamos el selector `"a[lang=es],a[lang=fr]"` limitamos esta búsqueda a únicamente los elementos de tipo `"a"` con el atributo `"lang"` con los valores `"es"` y `"fr"`.



`$(selector).find(elemento)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.find()

Objeto jQuery o elemento del
DOM frente al que comparar
los descendientes

HTML

En esta ocasión modificamos la llamada al método `.find()` indicando como parámetro dos datos diferentes: en primer lugar le indicaremos un elemento del DOM, y en segundo lugar, un objeto jQuery.


```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

El método .find() buscará en todos los descendientes, así que podremos descender a lo largo del DOM de los elementos incluidos en la selección sobre la que se aplique.

Código de Ejemplo

```

<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola los elementos que componen la seleccion
  $seleccion = $("#header");

  var elementoUILink = document.getElementById("lang_links");
  $seleccionFiltro = $("div");

  console.log( $seleccion.find( elementoUILink ) );
  console.log( $seleccion.find( $seleccionFiltro ) );
});
</script>

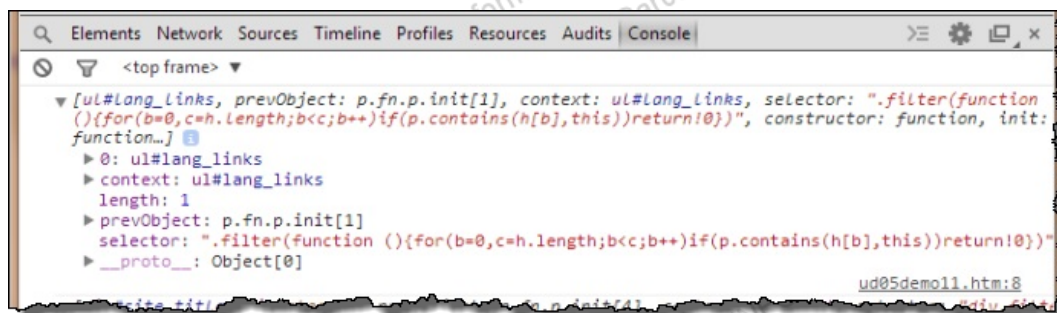
```

Resultado

Al igual que en el ejemplo anterior, almacenamos el resultado de realizar la selección #header en la variable \$seleccion. Esta selección contendrá únicamente el elemento div#header.

Con la selección guardada, obtenemos, por un lado, el elemento #lang_links llamando al método de JavaScript document.getElementById y, por otro lado, guardamos el resultado de realizar la selección \$("div") que contendrá todos los elemento de tipo "div" de la página.

Así, al llamar al método .find sobre la selección con el parámetro elementoUILink el método localizará de entre todos los descendientes del elemento #header el almacenado en la variable. Dado que este elemento sí es un descendiente del elemento #header la selección resultante contiene este elemento como resultado. Si este elemento no fuese un descendiente de #header, la selección final contendría cero elementos.



Del mismo modo, al llamar al método con el resultado almacenado en la variable \$seleccionFiltro, el resultado final solamente contendrá los elementos de tipo "div" descendientes de #header. Si observamos el resultado, el elemento div#content (que estaba incluido como resultado en \$seleccionFiltro por cumplir con el selector usado para obtener esta selección) no se encontrará en la selección final dado que no es un elemento descendiente del div#header.





Método .find()

Acceso a elementos hermanos (siblings)

Método .next()

El método .next() obtendrá el elemento hermano siguiente de cada uno de los elementos seleccionados por el objeto jQuery sobre el que sea aplicado. Opcionalmente este método aceptará como parámetro un selector, que filtrará el resultado conseguido por el método resultando únicamente aquellos elementos que cumplan con el selector.

`$(selector).next(selector)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .next()

Selector opcional con el que filtrar
los hermanos siguientes

HTML

Vamos a ver cómo funciona el método .next(). En este ejemplo haremos dos llamadas al mismo: una sin ningún parámetro para ver su funcionamiento básico y otra con un selector, que filtrará de entre los hermanos siguientes de los elementos sobre los que se aplique el método aquellos que cumplan con la condición indicada por el selector.

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

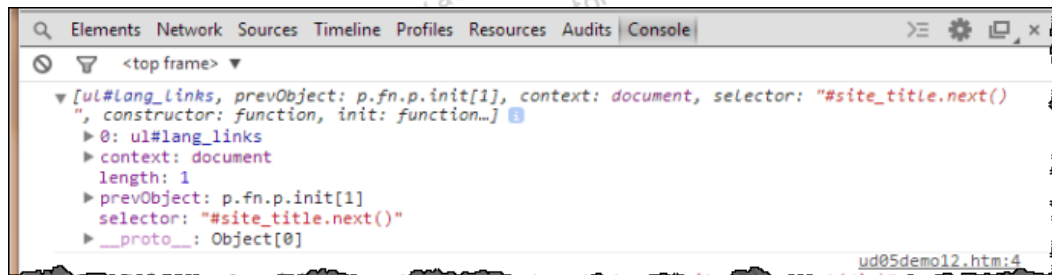
```

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

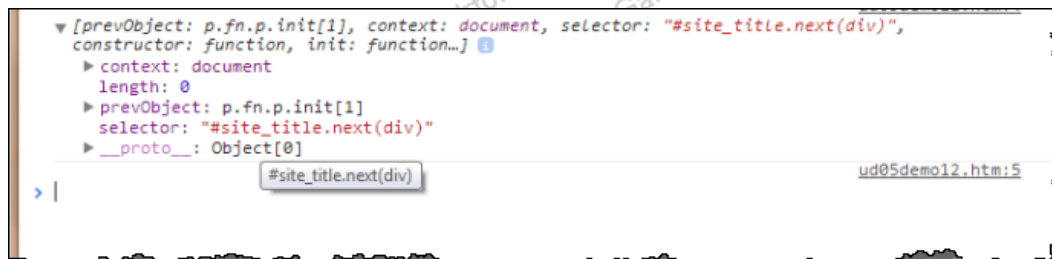
```

Resultado

Como resultado de la primera llamada al método `.next()` el objeto jQuery resultante contendrá un único elemento: el hermano siguiente del `div#site_title`

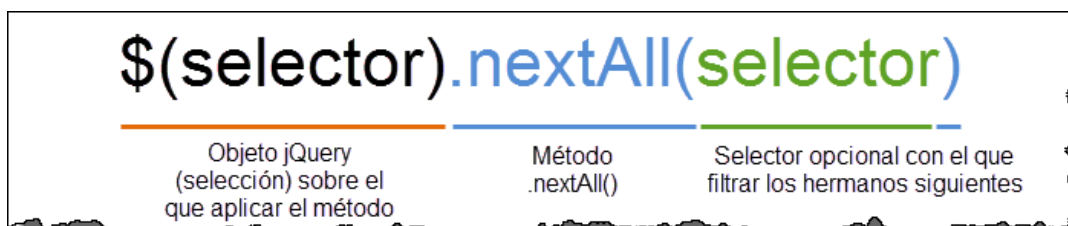


Sin embargo, al aplicarle al método el selector "div", el resultado contará con cero elementos, puesto que el hermano siguiente del elemento `div#site_title` no es un elemento `div`, sino que es un elemento `ul`.



Método `.nextAll()`

El método `.nextAll()` devolverá un conjunto con todos los hermanos posteriores de cada uno de los elementos seleccionados por el objeto jQuery sobre el que se aplique el método.



Al igual que el método `.next()`, este método también podrá recibir un selector como parámetro, que filtrará el conjunto de hermanos posteriores devolviendo únicamente los que cumplan con el selector.

HTML

Veamos cómo cambian los resultados respecto al uso del método `.next()`

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

```

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

```

Resultado

Al llamar al método `.nextAll()` el objeto jQuery resultante contendrá todos los hermanos siguientes del `div#site_title`

```

[ul#lang_links, ul#social_box, div.cleaner, prevObject: p.fn.p.init[1], context: document,
selector: "#site_title.nextAll()", constructor: function, init: function...]
  ▶ 0: ul#lang_links
  ▶ 1: ul#social_box
  ▶ 2: div.cleaner
  ▶ context: document
  ▶ length: 3
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "#site_title.nextAll()"
  ▶ __proto__: Object[0]
  
```

Y, en esta ocasión, al aplicarle al método el selector "div", el resultado sí contendrá un elemento: el último hermano del `div#site_title`, el div con la clase `.cleaner`

```

[div.cleaner, prevObject: p.fn.p.init[1], context: document, selector: "#site_title.nextAll(
div)", constructor: function, init: function...]
  ▶ 0: div.cleaner
  ▶ context: document
  ▶ length: 1
  ▶ prevObject: p.fn.p.init[1]
  ▶ selector: "#site_title.nextAll(div)"
  ▶ __proto__: Object[0]
  
```

Método `.nextUntil()`

El método `.nextUntil()` devolverá, para cada uno de los elementos seleccionados por el objeto jQuery sobre el que se aplique el método, todos sus hermanos posteriores hasta localizar un elemento que cumpla con el selector proporcionado al método.

Si para algún elemento no se localiza ningún hermano posterior que cumpla con la condición indicada, entonces todos los hermanos posteriores serán incluidos en el resultado.

\$(selector).
nextUntil
(selector, filtro)

Objeto jQuery
 (selección) sobre el
 que aplicar el método

Método .nextUntil()

Selector que
 especifica el
 elemento límite

Filtro opcional que
 filtrará los elementos
 a devolver



El elemento hermano que cumpla con el selector no será incluido en el resultado.

HTML

Volvamos a nuestro fragmento de código HTML. Vamos a aplicar el método `.nextUntil()` sobre el objeto `#site_title`, y obtener todos los hermanos que se encuentren entre éste y entre el `ul #social_box`

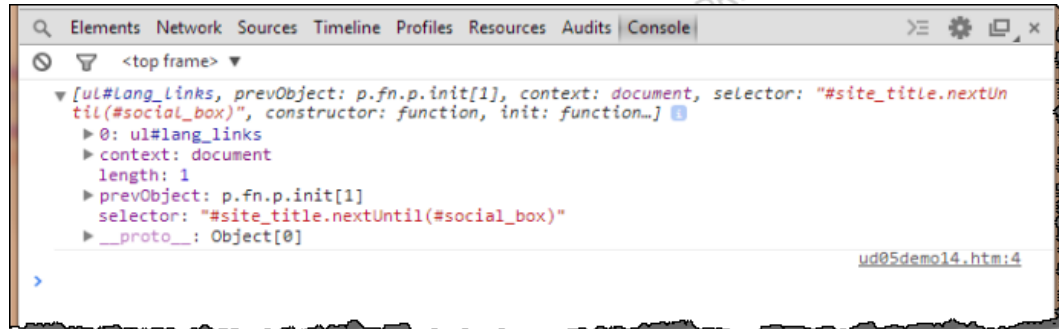
```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>
```

Código de Ejemplo

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


Resultado

Al llamar a este método con el parámetro `#social_box`, se limitarán los resultados obtenidos, incluyendo todos los elementos hermanos hasta que el elemento `#social_box` se localice. Como se puede observar, éste último no se incluirá en los resultados tras llamar al método.



En caso de que no se proporcione un selector como parámetro, este método tendrá el mismo funcionamiento que el método `.nextAll()`.

Método `.prev()`

El método `.prev()` obtendrá el elemento hermano anterior a cada uno de los elementos seleccionados por el objeto jQuery sobre el que sea aplicado. Si el método es provisto de un selector como parámetro, el resultado contendrá únicamente a los hermanos anteriores que cumplan con este selector.

`$(selector).prev(selector)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
`.prev()`

Selector opcional con el que filtrar
los elementos predecesores

HTML

Vamos a ver un uso básico del método `.prev()`. Este método devolverá el hermano inmediatamente anterior a cada uno de los elementos incluidos en la selección sobre la que se aplique.

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

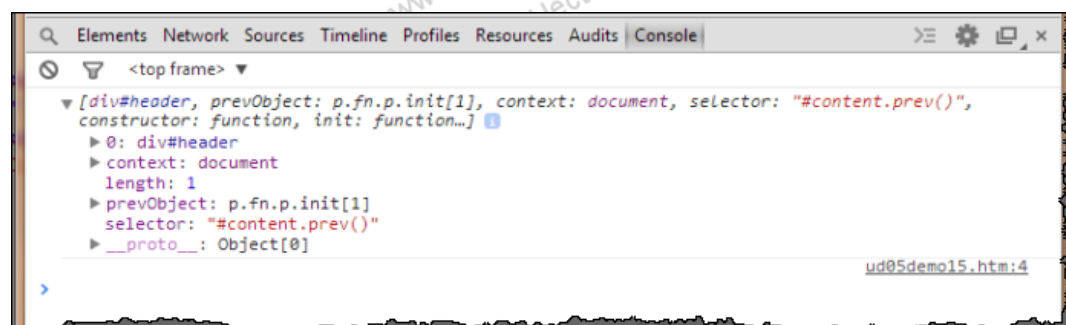
```

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

```

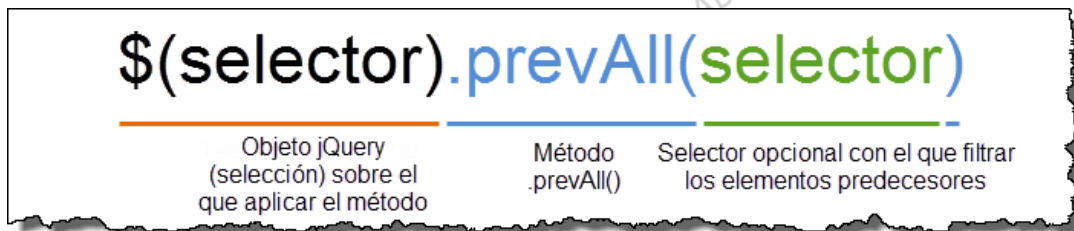
Resultado

El método recogerá el elemento previo al div#content obteniendo como resultado un único elemento, correspondiéndose con el div#header



Método .prevAll()

Con el método .prevAll(), obtendremos todos los hermanos anteriores de cada uno de los elementos seleccionados por el objeto jQuery sobre el que se aplique el método. Este método aceptará un selector como parámetro, que filtrará el resultado a devolver limitándolo únicamente a los elementos que cumplan con el selector.



HTML

En esta ocasión vamos a ver cómo funciona el método .prevAll(). Vamos a aplicar el método sobre una selección que contendrá el último elemento li contenido en el ul#lang_links

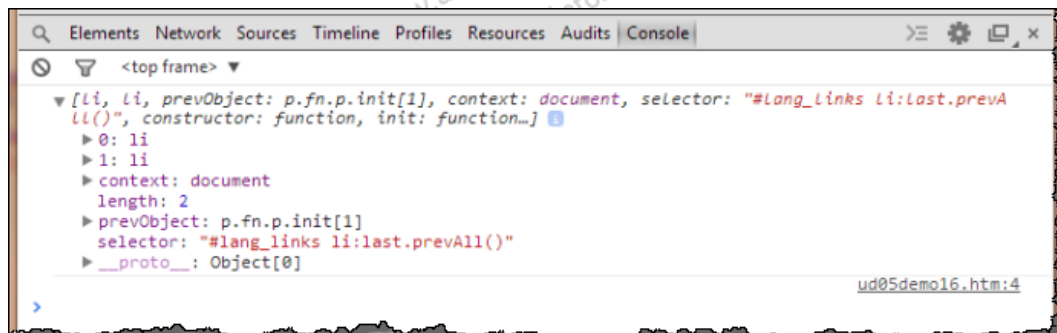
```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>
```

Código de Ejemplo

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

Resultado

Una vez hemos aplicado el método a la selección, el objeto resultante contendrá dos elementos correspondientes a los dos primeros elementos li existentes en el ul#lang_links

**Método .prevUntil()**

El método .prevUntil() funcionará de forma similar al método .prevUntil(), con la diferencia de que la búsqueda se realizará sobre los elementos previos. Al igual que el anterior método mencionado, también aceptará un selector como parámetro, indicando el límite de los elementos a seleccionar.

En el resultado devuelto por este método, los elementos hermanos devolverán ordenándose inversamente, inicialmente los más cercanos a los elementos del conjunto sobre el que se aplica y posteriormente los más lejanos a éstos.

`$(selector).prevUntil(selector)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.prevUntil()

Selector opcional con el que filtrar
los elementos predecesores

HTML

Veamos cómo funciona el método `.prevUntil()`. Vamos a hacer una llamada a este método con un selector que actuará como límite de la selección.

```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>
```

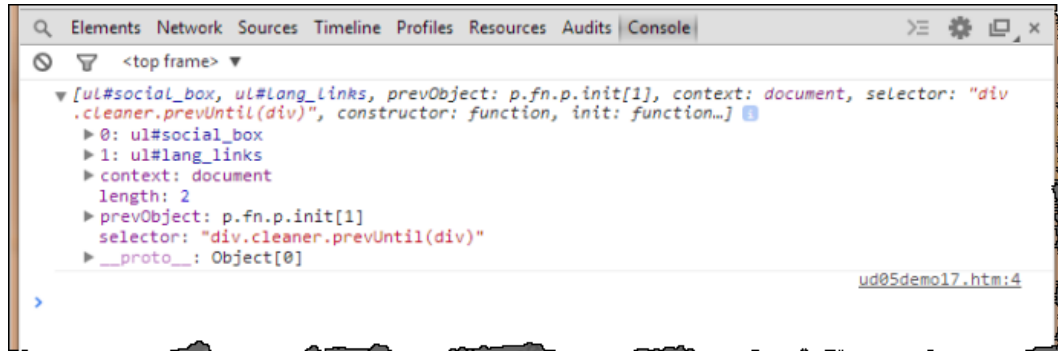
Recuerda que, si no indicásemos ningún selector, este método ofrecerá la misma funcionalidad que el método `.prevAll()`

Código de Ejemplo

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

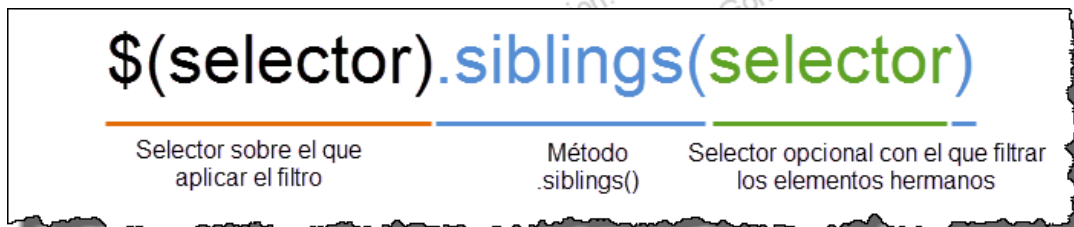
Resultado

La selección resultante contendrá los elementos `ul#social_box` y `ul#lang:links` dado que son hermanos previos del `div.cleaner` y se encuentran entre éste y el primer `div` que se localiza entre éstos en orden inverso.



Método .siblings()

El método `.siblings()` nos permite obtener para cada uno de los elementos de una selección los elementos hermanos de cada uno de ellos, devolviendo un nuevo objeto jQuery con la nueva selección. Este método aceptará como parámetro un selector, que filtrará los elementos a devolver, limitando el resultado únicamente a aquellos elementos hermanos que cumplan con el selector.



HTML

Vamos a hacer uso del método `.siblings()` para obtener los elementos hermanos del elemento de tipo `a` cuyo atributo `lang` tiene el valor `en`.

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

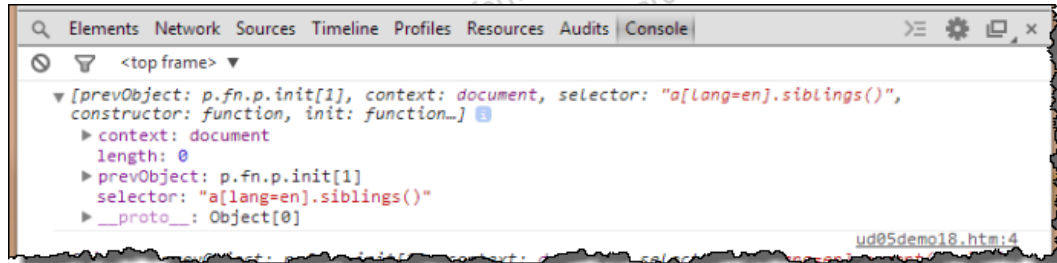
```

<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola los elementos que componen la seleccion
  console.log( $("a[lang=en]").siblings() );
  console.log( $("a[lang=en]").parent().siblings() );
});
</script>

```

Resultado

Al llamar al método `.siblings()` sobre la selección `"a[lang=en]"` el objeto resultante no contendrá ningún elemento, dado que es el único elemento hijo del elemento `"li"` que lo contiene.



Sin embargo, aplicando antes el método `.parent()`, el método `.siblings()` localizará dos elementos hermanos que se corresponden a los dos elementos `"li"` del `ul#lang_links`



Métodos de filtrado

Los métodos de filtrado se aplicarán sobre objetos jQuery (resultantes de selecciones previas) para modificar el resultado de la selección.

A diferencia de los selectores de filtrado, estos métodos se aplicarán sobre un resultado previo, sobre el que previamente podremos haber realizado diferentes acciones.

Método `.eq()`

El método `.eq()` reducirá el conjunto de elementos seleccionados a un único elemento. El elemento seleccionado será aquel cuyo índice se indique como parámetro al método. El parámetro deberá ser un número, que irá del cero a la longitud de la selección menos uno.

\$(selector).eq(index)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .eq()

Índice del elemento
a seleccionar



Los índices comienzan por el número 0, llegando como máximo al número total de elementos localizados por el selector menos uno



Desde la versión 1.4 de jQuery es posible indicar un índice negativo. En este caso, la librería hará una búsqueda inversa, comenzando por el final de la selección.

HTML

Vamos a utilizar el método .eq() para localizar dos enlaces concretos. El primero se buscará entre los descendientes del ul#lang_links, y el segundo, entre los descendientes del ul#social_box.

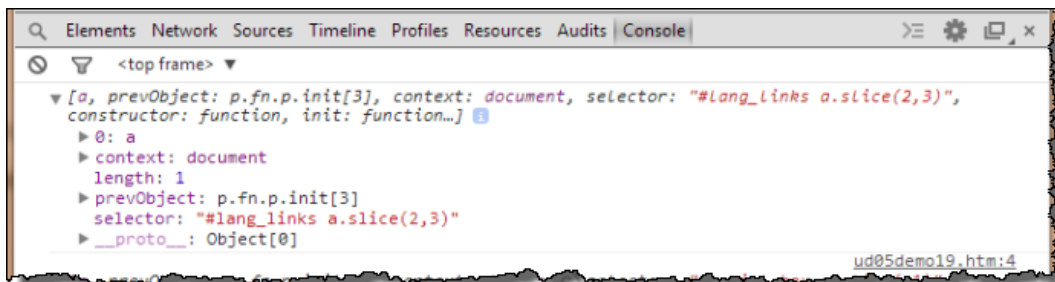
```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>
```

Código de Ejemplo

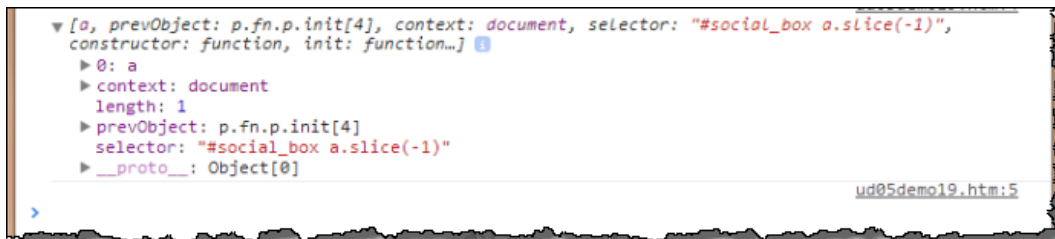
```
<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola el elementos que componen la seleccion
  console.log($("#lang_links a").eq(2));
  console.log($("#social_box a").eq(-1));
});</script>
```

Resultado

El primer comando filtrará de entre los enlaces existentes en el ul#lang_links el elemento con índice 2 que se corresponde con el elemento a cuyo atributo lang está establecido a francés [lang=fr]



Si observamos el resultado del segundo comando veremos que, habiendo seleccionado los elementos a del ul#socialbox, éstos se filtran obteniendo el de índice -1 con lo que la librería realiza la búsqueda inversa y devuelve el último de éstos elementos.



Método .eq()

Método .filter()

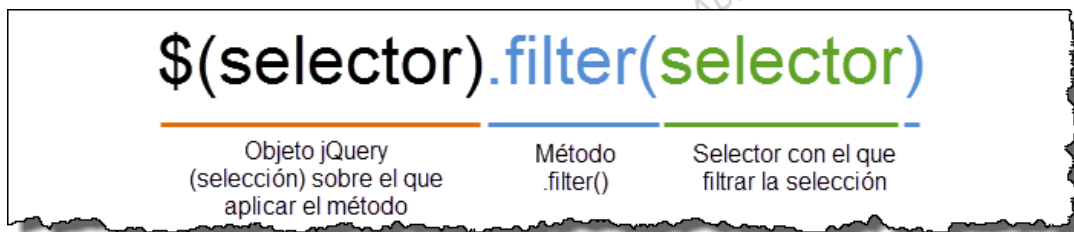
Con el método .filter() podremos filtrar una selección de elementos, reduciéndola por medio de un filtro que se le indique como parámetro.

Este método es uno de los más potentes existentes en jQuery. Permitirá recibir como parámetro hasta cuatro tipos de parámetro diferente:

- selector
- funcion
- elemento
- objeto jQuery

Método .filter() con selector como parámetro

La primera forma de realizar un filtrado es proporcionándole como parámetro un selector. De este modo, se analizará el conjunto de elementos del objeto jQuery y se obtendrá un nuevo objeto con aquellos elementos que cumplan con el selector indicado como parámetro.



HTML

En este ejemplo vamos a ver cómo filtrar una selección por medio del método .filter() proporcionándole como parámetro un selector.

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

```

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

```

Resultado

En primer lugar, seleccionamos todos los elementos hijos del div#header y almacenamos el resultado dentro de la variable \$seleccion. Una vez guardado este resultado, le aplicamos el método .filter() indicándole como parámetro el selector "ul". Como resultado, la consola nos mostrará que la selección final contiene únicamente los dos elementos ul de entre todos los elementos hijos anteriormente capturados.



Método .filter() I

Método .filter() con función como parámetro

Otra forma de realizar el filtrado es por medio de una función, que realizará las labores de filtro. Así, el parámetro podrá ser el nombre de una función previamente definida o el código de una función anónima (también llamadas funciones autoejecutables).

`$(selector).filter(function(){/*codigo*/})`

Objeto jQuery
(selección) sobre el que
aplicar el método

Método
.filter()

Nombre de la función o código de función
anónima



Las funciones anónimas, tal como su nombre indica, son funciones que no tienen un nombre asignado. Son ejecutadas automáticamente por el intérprete de JavaScript, del mismo modo que si hiciésemos una llamada a una función declarada.

Función Declarada

```
var mi_funcion = function(){
    alert('soy la función con nombre');
}

if(condicion){
    mi_funcion();
}
```

Función Anónima

```
if(condicion){function(){
    alert('soy la función sin nombre');
}}
```

La función proporcionada deberá devolver un valor booleano, que indique si el elemento analizado ha pasado o no el filtro.

HTML

Vamos a ver cómo funciona el método `.filter()` utilizando como condición de filtro una función.

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

```

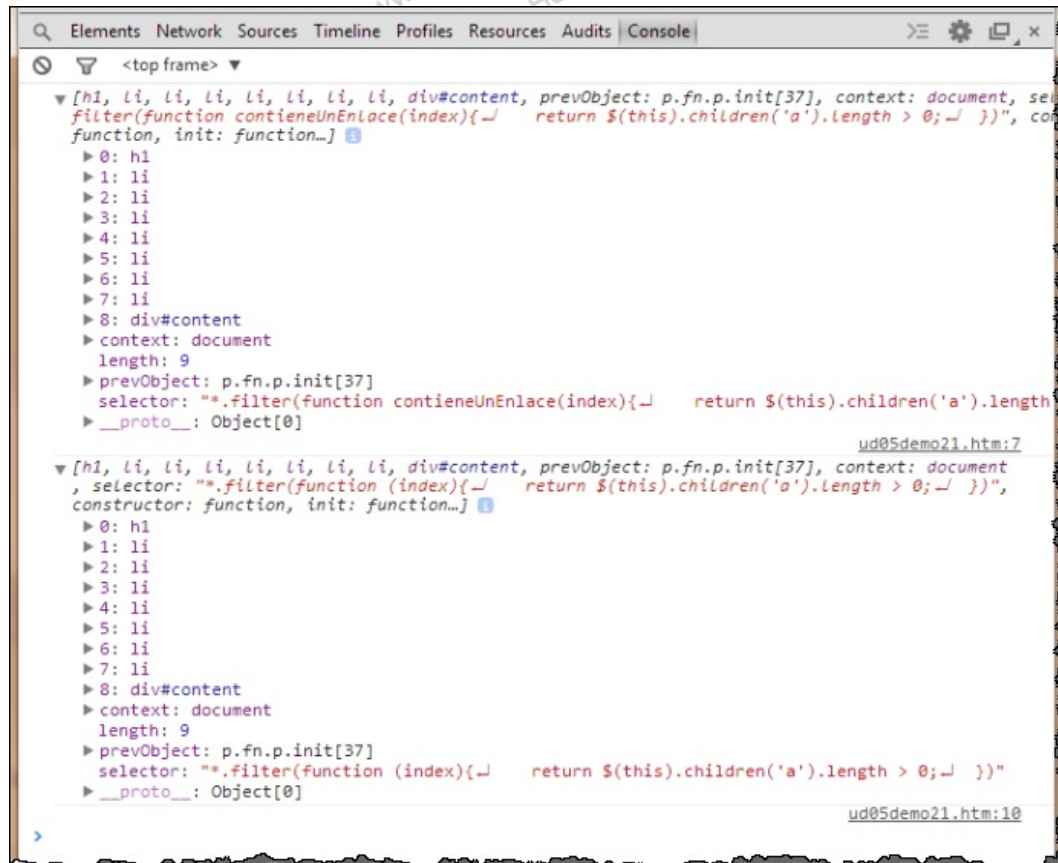
<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola los elementos que componen la seleccion
  function contieneUnEnlace(index){
    return $(this).children('a').length > 0;
  }
  //llamada al método con una función declarada
  console.log( $("***").filter( contieneUnEnlace ) );

  //llamada al método con una función anónima
  console.log( $("***").filter( function(index){
    return $(this).children('a').length > 0;
  } ) );
});
</script>

```

Resultado

Como puedes observar el resultado de ambas llamadas al método `.filter()` devuelven el mismo resultado.



Dentro de la función de filtrado podemos hacer uso de la variable `this`, que para cada llamada a la función (una por cada elemento existente en la selección), contendrá como valor el elemento del DOM de cada interacción. Así, al utilizar éste con la función `$`, obtenemos un objeto jQuery sobre el que podemos aplicar otro método de la librería.

Método `.filter()` con elemento como parámetro

El método `.filter()` también podrá recibir como parámetro un elemento del DOM. En este caso, el nuevo resultado solamente contendrá el elemento que coincida con el parámetro proporcionado y, en caso de que el elemento no se encuentre en la selección, el resultado será un conjunto vacío.

`$(selector).filter(elemento)`

Objeto jQuery
(selección) sobre el que
aplicar el método

Método
`.filter()`

Elemento del DOM que
filtrar en la selección

HTML

Vamos a ver cómo funciona el método .filter() utilizando como condición de filtro una función.

```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>
```

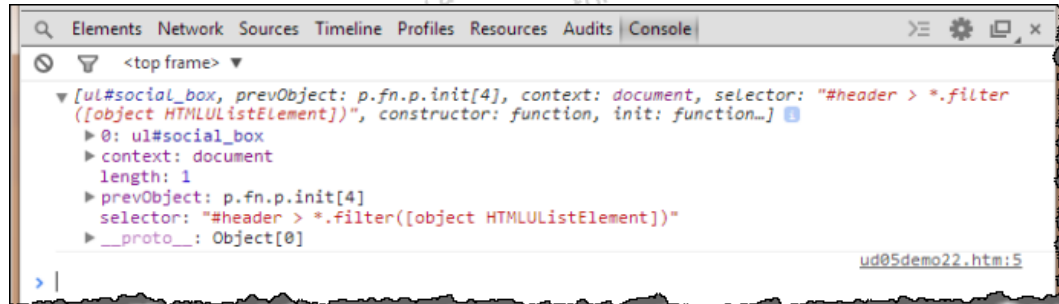
Código de Ejemplo

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

  console.log( $("#header > *").filter( elemento ) );
});
</script>
```

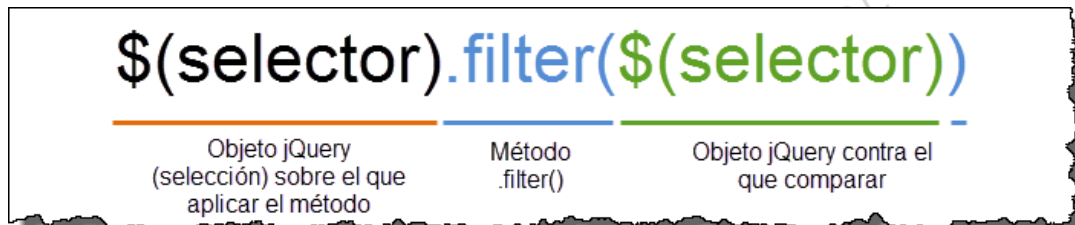
Resultado

La selección principal contiene todos los elementos hijos del div#header. Al aplicarle el método conseguimos reducir la selección, manteniendo únicamente el ul#social_box, que habíamos seleccionado previamente con la función de JavaScript document.getElementById().



Método .filter() con objeto jQuery como parámetro

Por último, este método podrá también recibir un objeto jQuery. En este caso, solamente los elementos del resultado sobre el que se aplique el método que se encuentren en el objeto pasado como parámetro se incluirán en el nuevo resultado.



HTML

Vamos a ver cómo conseguir el mismo funcionamiento que en el ejemplo anterior utilizando, como parámetro del método .filter(), un objeto jQuery.

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

```

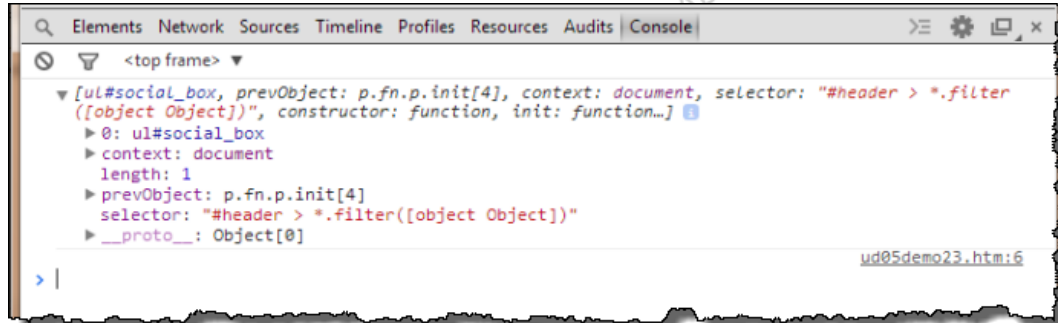
<script type="text/javascript">
  $(document).ready(function(){
    //mostrar por consola los elementos que componen la seleccion
    $seleccion = $("#social_box");

    console.log( $("#header > *").filter( $seleccion ) );
  });
</script>

```

Resultado

Tal y cómo se esperaba, la selección resultante es idéntica a la obtenida en el ejemplo anterior.



La principal diferencia radica en que el parámetro indicado al método no sólo tiene un valor diferente sino que son tipos de datos completamente distintos.



Método .filter() II

Diferencias entre los métodos .find() y .filter()

Dada su similitud de funcionamiento, los métodos .find() y .filter() pueden generarse dudas respecto a cuándo debe usarse uno u otro.

Por un lado, el método .filter() se usa para reducir la selección de elementos sobre la que se aplica el método.



Antes de utilizar el método `.filter()` analiza si puedes evitar el uso del mismo modificando el selector principal:

Por ejemplo, estos dos comandos devolverán un objeto jQuery con la misma selección final:

```
$(".navigation").filter("p :odd")
```

```
$("p.navigation:odd")
```

Por otro lado, el método `.find()` puede utilizarse para localizar elementos descendientes de la selección sobre la que sea aplicado que cumplan con el selector indicado como parámetro al método.

Método `.first()`

El método `.first()` reducirá el conjunto de elementos resultante de la selección, limitándolo únicamente al primero de los elementos.

`$(selector).first()`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.first()`

HTML

Con nuestro habitual fragmento de HTML vamos a hacer uso del método `.first()`. Este método reducirá la selección sobre la que lo apliquemos y únicamente mantendrá un elemento: el primero de ellos.

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

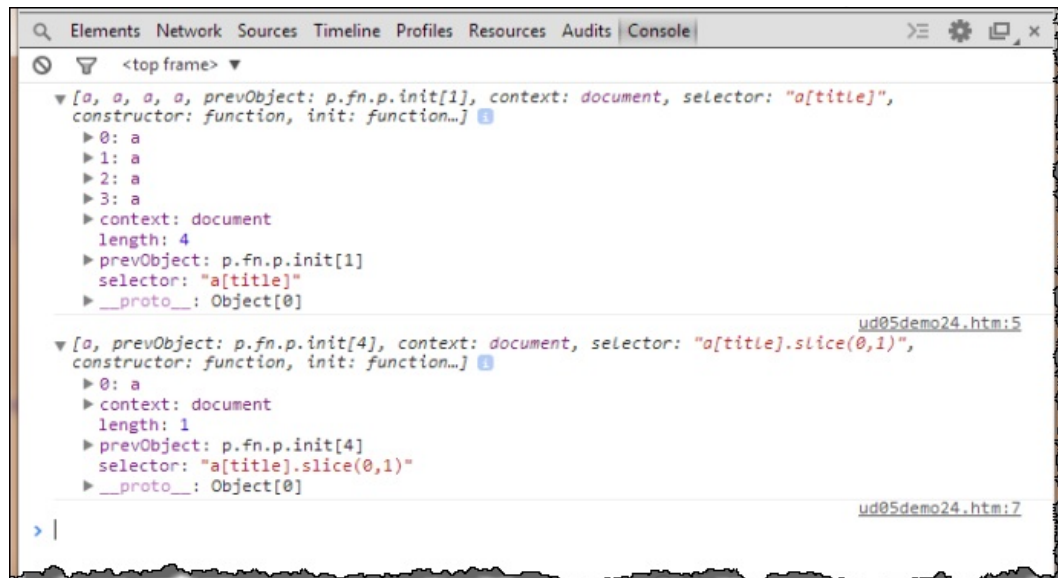
```

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

```

Resultado

Pese a que la selección original contenía cuatro elementos, una vez aplicada el método `.first()` el resultado final contendrá un único elemento, siendo éste el primer elemento `a` con título de la página.



Método .has()

Con el método `.has()` filtraremos el resultado de la selección sobre la que se aplique, reduciéndolo a aquellos elementos que, entre sus descendientes, exista al menos un elemento que cumpla con el selector proporcionado como parámetro.

`$(selector).has(selector)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.has()`

Selector que especificará los
descendientes a contener

HTML

Vamos a utilizar el método `.has()` para filtrar una selección y mantener, de entre los elementos disponibles en la selección, solamente aquellos que contengan elementos de tipo `img`:

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

```

<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola los elementos que componen la seleccion
  $seleccion = $("a[title]");
  console.log( $seleccion.has("img") );
});
</script>

```


Resultado

Con la llamada al método `.has()` la selección de elementos "a" con atributo "title" se reducirá a aquellos que contengan un elemento "img" entre sus descendientes. En esta ocasión el resultado es el mismo, dado que todos los elementos "a" de nuestra página contienen una imagen.



`$(selector).has(elemento)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.has()`

Elemento del DOM que deben
contener como descendiente

El método `.has()` también puede ser invocado con un parámetro que se corresponda a un elemento específico del DOM. La funcionalidad conseguida será similar a la conseguida con el método `.parents(selector)` ya que obtendremos el elemento predecesor del indicado.

HTML

Veamos el mismo efecto utilizando un elemento del DOM como parámetro limitador. Una vez más, utilizamos nuestro habitual fragmento de HTML:

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

```

<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola los elementos que componen la seleccion
  var elemento = document.getElementsByTagName('a')[0]
  console.log( $("div").has( elemento ) );
});
</script>

```

Resultado

Con el uso de la función nativa de JavaScript `document.getElementsByTagName()` obtendremos un array con todos los elementos de tipo "a" del DOM. De este array, seleccionaremos el primero de ellos (indicando el índice [0]) y lo almacenamos en la variable elemento.

Así, dispondremos del elemento que deseamos indicar al método `.has()` como filtro. Como resultado, limitaremos la selección de todos los elementos "div" de la página, filtrando solamente aquellos que son predecesores en el DOM de nuestro elemento.



Método .is()

El método `.is()` comparará cada elemento de la selección frente a una condición pasada como argumento al método, devolviendo el valor booleano `TRUE` (verdadero) si existe al menos un elemento en la selección que cumpla con la condición indicada al método.

Este método aceptará varios tipos de parámetro diferentes, contra los que comparar el elemento seleccionado.

Un selector



Una función

`$(selector).is(function(){/*codigo*/})`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .is()

Función que se ejecutará para cada
elementos de la selección

Un elemento del DOM

`$(selector).is(elemento)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .is()

Elemento del DOM
con el que comparar
selección

Un objeto jQuery

`$(selector).is($(selector))`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .is()

Objeto jQuery contra el
que filtrar la selección

HTML

Vamos a ver cómo utilizar el método con los cuatro tipos de parámetros admitidos.

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  $elemento=$("#social_box");
  if($elemento.is(":first-child")){
    console.log("Es el primer elemento hijo de su padre");
  }
  else
  {
    console.log("No es el primer elemento hijo de su padre");
  }

  if($elemento.is(function(index){
    return this.tagName == "ul";
    //equivale a $(this).is("ul");
  })){
    console.log("El elemento seleccionado es un UL");
  }
  else
  {
    console.log("El elemento seleccionado no es un UL");
  }

  miElemento=document.getElementById("lang_links");
  if($elemento.is(miElemento)){
    console.log("El elemento seleccionado es #lang_links");
  }
  else
  {
    console.log("El elemento seleccionado no es #lang_links");
  }

  if($elemento.is($("#*[id]"))){
    console.log("El elemento es un elemento con id");
  }
  else
  {
    console.log("El elemento es un elemento sin id");
  }
});
</script>
```

Resultado

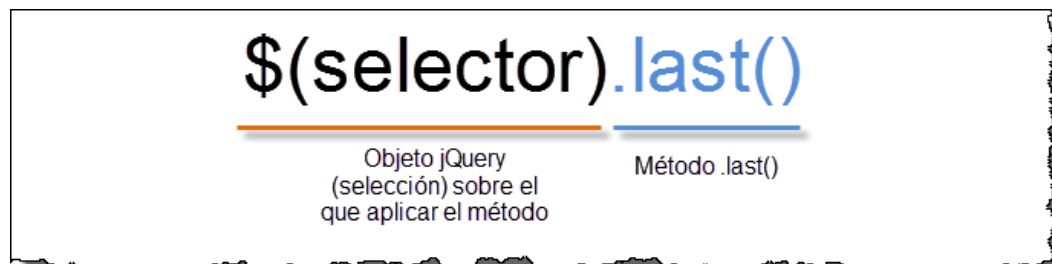
En este ejemplo vemos las cuatro formas de llamar al método `.is()`.

descripcion



Método `.last()`

El método `.last()` reducirá el conjunto de elementos sobre el que se aplique el método reduciéndolo únicamente al último de estos elementos.



HTML

Con el método `.last()` vamos a conseguir reducir una selección y mantener en la misma únicamente al último elemento contenido en ésta.

```

<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>

```

Código de Ejemplo

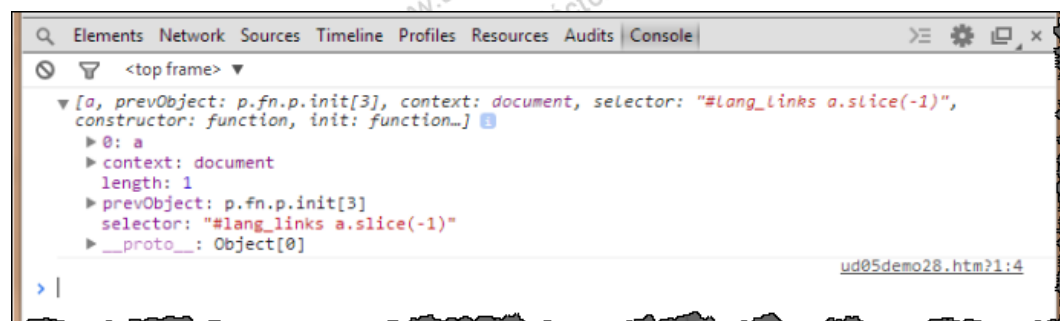
```

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

```

Resultado

Pese a que la selección original contendrá los tres elementos de tipo enlace descendientes del `ul#lang_links`, el resultado mostrado por consola únicamente el último de todos ellos, por efecto de aplicar el método `.last()`



Método .not()

El método .not() eliminará de la selección aquellos elementos que cumplan con la condición pasada como parámetro al método. Este método nos ofrecerá la función inversa del método .is()

A la hora de hacer uso del método podemos invocarlo con tres tipos de parámetros diferentes:

Un selector

`$(selector).not(selector)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .not()

Selector que indicará
el tipo de elemento a
excluir

Una función

`$(selector).not(function(){/*codigo*/})`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .not()

Función que evaluará para cada elemento de
la selección si debe ser excluido

Un objeto jQuery

`$(selector).not($(selector))`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .not()

Objeto jQuery que
contendrá los elementos a
excluir

HTML

Vamos a ver el funcionamiento de este método con sus tres formas de uso sobre una selección aplicada a nuestro habitual fragmento de código.

```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>
```

Código de Ejemplo

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

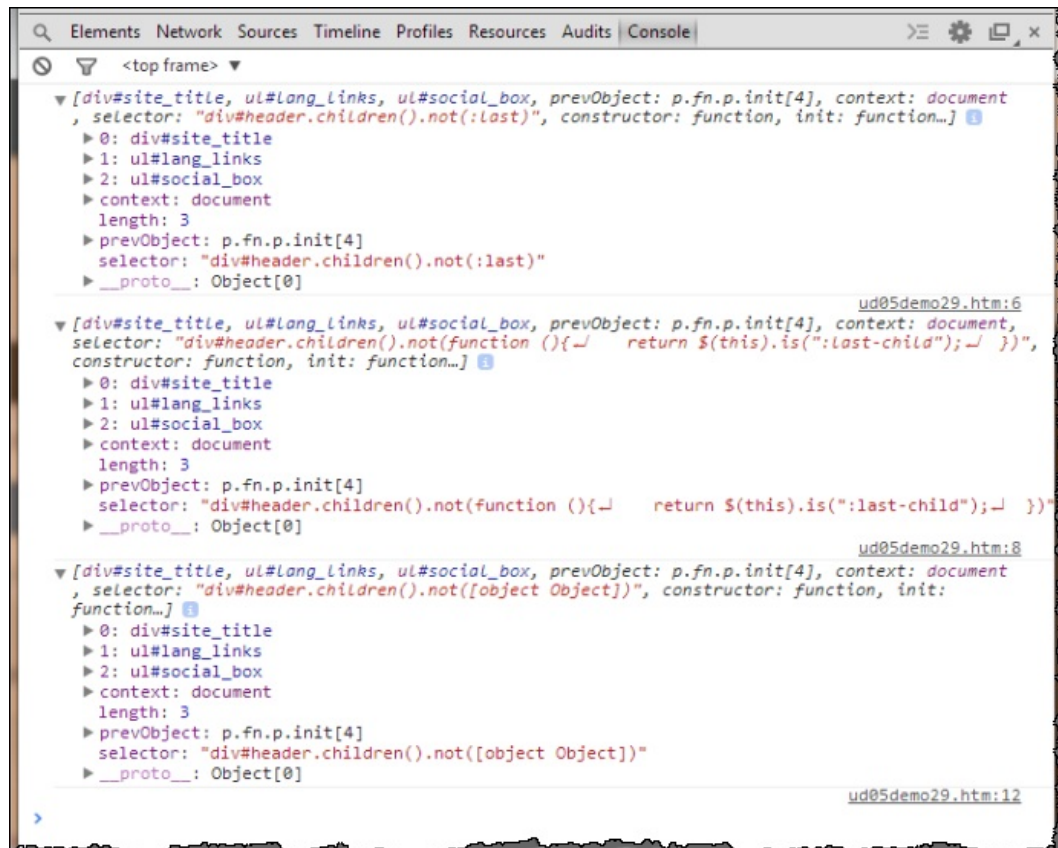
  console.log( $seleccion.children().not(":last") );

  console.log( $seleccion.children().not(function(){
    return $(this).is(":last-child");
  }));

  console.log( $seleccion.children().not($(":last-child")));
});
</script>
```

Resultado

Como resultado de las tres llamadas al método, el objeto resultante contendrá los mismos elementos. Estos elementos serán aquellos elementos hijo del `div#header` que no sean el último de ellos.



Método .slice()

Con el método `.slice()` podremos obtener un subconjunto de elementos de la selección sobre la que se aplique, incluyendo los elementos cuyo índice se encuentre en el rango indicado al método.

Podremos definir el rango de elementos a seleccionar por medio de dos parámetros. el primero indicará el índice del primer elemento del rango y el segundo parámetro (opcional) indicará el índice del último elemento a incluir. Si no se proporciona el segundo parámetro, el método incluirá en el resultado a todos los elementos a partir del parámetro indicado como comienzo.

`$(selector).slice (inicio,fin)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.slice()

Posición
inicial

Opcional
Posición
final

HTML

Veamos el mismo efecto utilizando un elemento del DOM como parámetro limitador. Una vez más, utilizamos nuestro habitual fragmento de HTML:

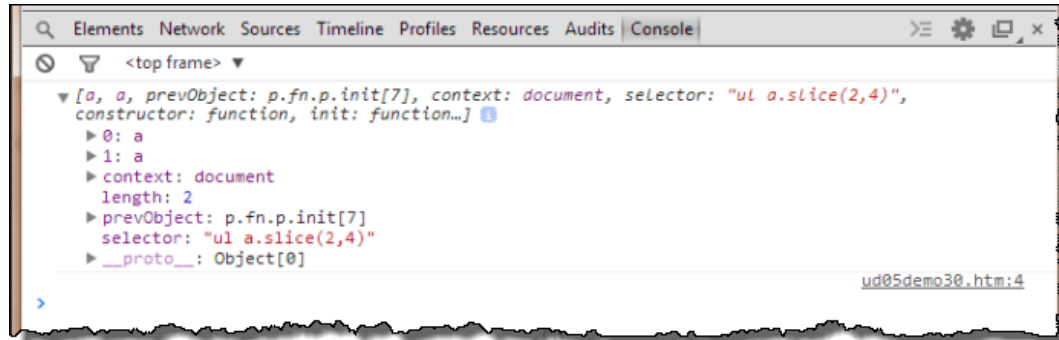
```
<div id="header">
  <div id="site_title">
    <h1><a href="#"><strong>GLOSSY</strong> BOX<span>YOUR TAGLINE GOES HERE</span></a></h1>
  </div> <!-- end of site_title -->
  <ul id="lang_links">
    <li><a href="#" lang="es" dir="ltr">ES</a></li>
    <li><a href="#" lang="en" dir="ltr">EN</a></li>
    <li><a href="#" lang="fr" dir="rtl">FR</a></li>
  </ul>
  <ul id="social_box">
    <li><a href="#" title="Publicar en Facebook"></a></li>
    <li><a href="#" title="Publicar en Twitter"></a></li>
    <li><a href="#" title="Publicar en LinkedIn"></a></li>
    <li><a href="#" title="En Technorati"></a></li>
  </ul>
  <div class="cleaner"></div>
</div>
<div id="content">
  <a href="#">Enlace 1</a>
  <a href="#">Enlace 2</a>
</div>
```

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  //mostrar por consola los elementos que componen la seleccion
  console.log( $("ul a").slice(2, 4));
});
</script>
```

Resultado

Al llamar al método `.slice()` reduciremos la selección, dejando únicamente aquellos elementos cuyos índices se encuentren entre los parámetros indicados. En este caso, la selección final tendrá el `a` con atributo `lang="fr"` y el `a` cuyo atributo `title="Publicar en Facebook"`.



Iteraciones: `.each` y `.map`

jQuery ofrece diversas posibilidades con las que realizar iteraciones sobre un conjunto de resultados. ...

Las iteraciones consisten en la repetición de un proceso en la ejecución de nuestro código, que se ejecutará un determinado número de veces.

La estructura de una iteración sería la siguiente:

```
MIENTRAS NoFinDeElementos(elementos) HACER
  HacerAlgo(ElementoActual(elementos))
FIN MIENTRAS
```

Iteraciones con el método `.each()` o `jQuery.each()`

Por medio de los métodos `.each()` y `jQuery.each()` podremos realizar iteraciones sobre objetos jQuery, otro tipo de objeto y arrays de elementos.

Por un lado, el método `.each()` se aplicará sobre un objeto jQuery, realizando la iteración sobre los elementos que componen la selección. Por el otro, el método auxiliar de jQuery `jQuery.each()` aplicará la iteración sobre el elemento que se le proporcione como parámetro.

Método `.each()`

Por medio del método `.each()` podremos ejecutar varias acciones de forma iterativa sobre los elementos incluidos como resultado en un objeto jQuery, evitando posibles errores en la escritura y definición de estos bucles iterativos. Para cada una de las iteraciones el método llamará a una función (comúnmente denominada "Callback"), proporcionada al mismo como parámetro, que podrá recibir varios argumentos en cada llamada:

- El índice del elemento sobre el que se esté aplicando la iteración
- El elemento que se esté tratando

La función de callback podrá hacer uso de éstos parámetros u omitirlos si no se consideran necesarias.

Método `jQuery.each()`

El método `jQuery.each()` ofrece una forma de realizar iteraciones sobre objetos y arrays de elementos. En el caso de este método, deberemos proporcionarle dos parámetros:

- El objeto o array sobre el que aplicar la iteración
- La función callback que se aplicará en cada una de las iteraciones

Al igual que ocurría con el método `.each()`, la función de callback recibirá dos parámetros en cada llamada. Sin embargo, estos parámetros tendrán un contexto diferente:

- El primero de ellos indicará el índice que cada uno de los valores tenga en el array objeto de la iteración
- El segundo contendrá el valor correspondiente del elemento en el array

Iteraciones con el método `.map()` o `jQuery.map()`

Los métodos `.map()` y `jQuery.map()` tendrán un comportamiento muy similar a los métodos `.each()` y `jQuery.each()` respectivamente. Estos dos métodos también realizarán una iteración con los elementos sobre los que se apliquen. Del mismo modo, ambos dispondrán de una función de retrollamada o callback, que recibirá diversos parámetros en cada iteración.

Método `.map()`

El método `.map()` pasará cada uno de los elementos del conjunto sobre el que se aplique a través de una función. Como resultado, obtendremos un nuevo objeto jQuery que contendrá los valores devueltos por la función.


Método `jQuery.map()`

De forma similar al funcionamiento del método `.map()`, este otro método aplicará una función sobre los elementos de un array o de un objeto.

Ejercicios

Ejercicio 1: Transversal I

Duración estimada del ejercicio


25
minutos

El documento "ejercicio-transversal-i.htm" tiene los siguientes elementos del DOM:

```


<div id="divUno">
  <p>
    Div n&uacute;mero 1 con identificador divUno
  </p>
  <p>
    <input id="btnOcultarDiv1" type="button" value="Ocultar Div 1"/>
  </p>
</div>
<div id="divDos">
  <p>
    Div n&uacute;mero 2 con identificador divDos
  </p>
  <div id="divDescendiente3">
    <p>
      Div descendiente del div #divDos
    </p>
    <p>
      <input id="btnOcultarDiv2" type="button" value="Ocultar Div 2"/>
    </p>
  </div>
</div>

```

En la cabecera del documento, están capturados los eventos de pulsación de los botones. Completa el selector indicado en cada uno de ellos, haciendo uso de un método transversal.

Lo necesario para comenzar

Descarga el archivo y extráelo en tu directorio de trabajo.



ejercicio-transversal-i.zip

Documento comprimido con el archivo necesario para realizar el ejercicio.

Ejercicio 2: Transversal II

Duración estimada del ejercicio



25
minutos

El documento "ejercicio-transversal-ii.htm" tiene los siguientes elementos del DOM:

```
<form id="formUno">
<div id="divUno">
  <fieldset>
    <legend>Legend Name</legend>
    <label for="name1">Name</label>
    <input type="text" name="name1" id="name1" />
  </fieldset>
  <p>
    <input id="btnObtieneName1" type="button" value="Obtener input name1"/>
  </p>
</div>
</form>
<form id="formDos">
<div id="divDos">
  <p>
    <input id="btnObtieneName2" type="button" value="Obtener input name2"/>
  </p>
  <fieldset>
    <legend>Legend Name</legend>
    <label for="name2">Name</label>
    <input type="text" name="name2" id="name2" />
  </fieldset>
</div>
</form>
```

En la cabecera del documento, están capturados los eventos de pulsación de los botones. Completa el selector indicado en cada uno de ellos, utilizando método transversales encadenados, para lograr conseguir como selección final:

- El elemento `input#name1`
- El elemento `input#name2`

Lo necesario para comenzar

Descarga el archivo y extráelo en tu directorio de trabajo.



ejercicio-transversal-ii.zip

Documento comprimido con el archivo necesario para realizar el ejercicio.

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