

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

Manipulación del DOM © ADRINFOR S.L.

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

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

Indice

Manipulación del DOM	4
Introducción	4
Modificar elementos del DOM y sus propiedades	4
Métodos genéricos de modificación de Atributos	4
Método .attr()	5
Método .removeAttr()	9
Métodos específicos de modificación de Atributos	11
Método .html()	11
Método .text()	13
Método .val()	15
Método .height()	18
Método .innerHeight()	19
Método .outerHeight()	20
Método .width()	20
Método .innerWidth()	22
Método .outerWidth()	22
Método .offset()	22
Método .position()	23
Método .scrollLeft()	24
Método .scrollTop()	25
Clases y Estilos CSS	25
Método .addClass()	25
Método .hasClass()	27
Método .removeClass()	28
Método .toggleClass()	30
Método .css()	30
Funcionamiento como obtenedor	31
Funcionamiento como establecedor	32
Inserción y copiado de elementos del DOM	34
Creación de elementos con la función \$()	34
Crear elementos dinámicamente con la función \$()	34
Creación de elementos con métodos específicos	35
Método .after()	35
Método .insertAfter()	36
Método .before()	38
Método .insertBefore()	40
Método .append()	42
Método .appendTo()	43
Método .prepend()	43
Método .prependTo()	44
Método .wrap()	45
Método .wrapAll()	46
Método .wrapInner()	46
Método .clone()	47
Eliminación de elementos del DOM	48
Métodos específicos para la eliminación de elementos de la página	48
Método .detach()	48
Método .empty()	48

Método .remove()	49
Método .replaceAll()	49
Método .replaceWith()	49
Método .unwrap()	50
Ejercicios	51
Ejercicio 1: Utilidad de creación de formularios.	51
Lo necesario para comenzar	51
Ejercicio 2: Utilidad de creación de formularios mejorada.	51

Manipulación del DOM

Introducción

Como hemos visto anteriormente, jQuery ofrece un gran rango de utilidades para realizar selección de elementos, pero hasta ahora las acciones que realizábamos sobre los conjuntos de elementos seleccionados solamente nos servían para verificar que la condición indicada como selector era la correcta, mostrando por consola los elementos seleccionados o aplicando un sencillo estilo muy visual con el que destacar los elementos seleccionados.

En esta lección vamos a ver las múltiples operaciones que podremos realizar con jQuery con las que modificar los elementos seleccionados, alterando sus atributos y modificando la estructura del DOM a nuestro antojo.

Modificar elementos del DOM y sus propiedades

Una de las tareas más comunes que se puede realizar sobre el resultado de una selección es aplicar un método con el que modificar alguna de las propiedades de los elementos contenidos en la selección. En este apartado vamos a ver diferentes métodos con los que realizar esta acción, analizando por separado aquellos que actúan sobre atributos específicos de estos elementos de los métodos genéricos, que por medio de sus parámetros podremos definir los atributos sobre los que actuar.

La mayoría de estos métodos pueden funcionar tanto para obtener el valor de un atributo (obtenedores o getters) como para asignar un nuevo valor al mismo (establecedores o setters).

Setters

Cuando estos métodos funcionan como obtenedores, devolverán el valor del atributo únicamente del primer elemento contenido en el objeto jQuery sobre el que sean aplicados.

Getters

Sin embargo, cuando funcionen como establecedores, asignarán el valor indicado a todos los elementos contenidos en el objeto jQuery en el que se ejecute el método.

Métodos genéricos de modificación de Atributos

Este conjunto de métodos nos permitirán actuar sobre cualquier atributo de los elementos del DOM. Para ello, uno de los parámetros que deberemos indicarles hará referencia al atributo sobre la que deberá trabajar.

Algunos de estos métodos soportan la modificación de varios atributos en una única llamada. Esta actuación se realizará proporcionando como parámetro de llamada a los mismos un "objeto anónimo" cuyas propiedades contengan los pares atributo - valor.



Los objetos anónimos permiten crear tipos de objeto sin la necesidad de disponer de una definición de clase previa. Un objeto anónimo sólo puede tener propiedades.

```
var objetoAnonimo = { name : "Objeto Anonimo", color : "red", value : "25" }
```

Método .attr()

El método .attr() podrá ser utilizado tanto para obtener el valor de un atributo del primer elemento del conjunto seleccionado, así como para establecer uno o varios atributos sobre todos los elementos de la selección.

\$(selector).attr(atributo)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .attr()

Cadena indicando el nombre del
atributo del que obtener el valor

Para utilizar el método .attr() como obtenedor solamente será necesario indicar el nombre del parámetro que deseemos obtener. De este modo, el método devolverá como resultado el valor del atributo indicado correspondiente al primer elemento de la selección sobre la que se aplique el método.



Si el atributo indicado como parámetro no ha sido establecido en el elemento, el método devolverá como resultado el valor **undefined**.

\$(selector).attr(atributo, valor)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .attr()

Nombre del atributo
a modificar

Valor a establecer

Otra forma de utilizar el método .attr() es como establecedor. Para que actúe de este modo, el método deberá recibir dos parámetros:

- El nombre del atributo a establecer
- El valor a asignar a este atributo.

En caso de existir el parámetro indicado, el valor del mismo será sobrescrito. Si no estuviese asignado, el atributo sería creado con el valor.

HTML

Vamos a ver cómo se comporta este método. Para ello, vamos a realizar dos comandos sobre este escenario:

```
<div id="srchContent">
  <form id="srchForm" type="get" action="#">
    <input id="srchText" name="srchText" type="text" />
  </form>
</div>
```

Disponemos de un div que contiene un formulario. Por un lado, vamos a consultar el atributo "action" del mismo, y posteriormente, añadiremos un nuevo atributo que por su definición en HTML no ha sido establecido.

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  //mostrar el atributo action del formulario en la consola
  console.log( "El valor del atributo 'action' es: " + $("#srchForm").attr("action") );

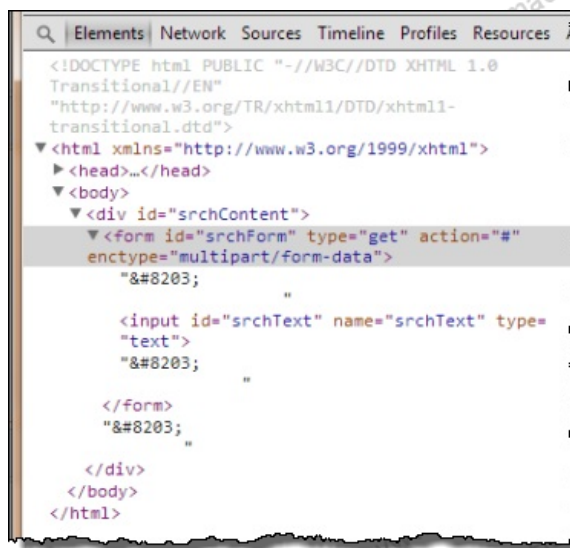
  //modificar el atributo enctype del formulario
  $("#srchForm").attr( "enctype", "multipart/form-data" );
});
</script>
```

Resultado

El primero de los comandos, en los que se utiliza el método `.attr()` como obtenedor, muestra por consola de pantalla el valor del atributo "action".



Por otro lado, si inspeccionamos los elementos del formulario podemos observar cómo se ha añadido el atributo "enctype" al formulario tal y cómo se pretendía con el segundo comando.



También podremos hacer uso de otra sintaxis, con la que utilizamos el método como establecedor de varios atributos sobre un objeto en bloque.

`$(selector).attr({attr1:valor1, attr2:valor2, attr3:valor3})`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.attr()`

Objeto anónimo con una lista de los atributos y sus
correspondientes valores

Para conseguir este comportamiento el método solamente será llamado con un parámetro, con forma de objeto anónimo, en el que cada una de las propiedades del objeto será un atributo a establecer con su valor correspondiente.

HTML

Volvamos al código HTML visto en el ejemplo anterior:

```
<div id="srchContent">
  <form id="srchForm" type="get" action="#">
    <input id="srchText" name="srchText" type="text" />
  </form>
</div>
```

Vamos a utilizar el método .attr() para indicar, por medio de un objeto anónimo, varios atributos a establecer en bloque.

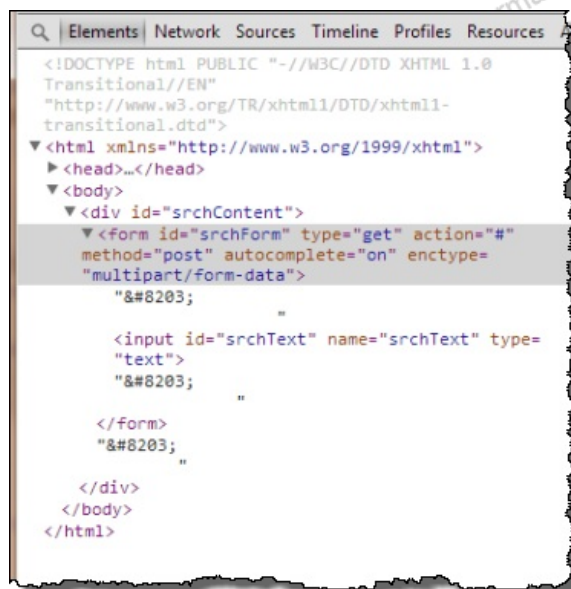
Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  var objetoAnonimo = { method: "post", autocomplete: "on", enctype: "multipart/form-data"};

  $("#srchForm").attr( objetoAnonimo );
});
</script>
```

Resultado

Al examinar el objeto del DOM una vez aplicados los nuevos atributos podemos observar que el atributo "method" ha sido modificado, y que se han añadido los dos atributos adicionales indicados en el objeto anónimo.



Por último, el método podrá recibir como segundo parámetro una función que se ejecutará para cada elemento de la selección sobre la que se aplique, estableciendo el resultado de la llamada a esta función como valor en el atributo.


```
$(selector).attr(atributo, function(){ /*code*/ })
```

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.attr()

Nombre del
atributo a
modificar

Función que devolverá el valor a
establecer en el atributo



Método .attr()

Método .removeAttr()

El método .removeAttr() eliminará el atributo proporcionado como parámetro de cada uno de los elementos seleccionados por el objeto sobre el que se aplique la llamada.

Este método recibirá como parámetro el nombre de atributo a eliminar.

```
$(selector).removeAttr(atributo)
```

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.removeAttr()

Atributo a eliminar
de los elementos
seleccionados



Desde la versión 1.7 de jQuery, el método podrá recibir varios atributos a eliminar, separando éstos por el símbolo ",".

HTML

Vamos a aplicar el método sobre un sencillo fragmento de HTML.

```
<div id="document_links">
  <ul id="internal_links">
    <li><a href="#destino1" target="_self">Destino Interno 1</a></li>
    <li><a href="#destino2" target="_blank">Destino Interno 2</a></li>
  </ul>
  <ul id="external_links">
    <li><a href="#destino_externo1" target="_self">Destino Externo 1</a></li>
    <li><a href="#destino_externo2" target="_blank">Destino Externo 2</a></li>
    <li><a href="#destino_externo3" target="_blank">Destino Externo 3</a></li>
  </ul>
</div>
```

En este ejemplo vamos a eliminar los atributos "target" de los enlaces contenidos en el `ul#external_links`.

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  $("#external_links a[target]").removeAttr("target");
});
</script>
```

Resultado

Al aplicar el método, todos los enlaces descendientes del `ul#external_links` que disponían del atributo "target" han visto eliminado el mismo, removiendo la funcionalidad que este método ofrece sobre los enlaces.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>...</head>
  <body>
    <div id="document_links">
      <ul id="internal_links">
        <li>
          <a href="#destino1" target="_self">Destino Interno 1</a>
        </li>
        <li>
          <a href="#destino2" target="_blank">Destino Interno 2</a>
        </li>
      </ul>
      <ul id="external_links">
        <li>
          <a href="#destino_externo1">Destino Externo 1</a>
        </li>
        <li>
          <a href="#destino_externo2">Destino Externo 2</a>
        </li>
        <li>
          <a href="#destino_externo3">Destino Externo 3</a>
        </li>
      </ul>
    </div>
  </body>
</html>
```

Métodos específicos de modificación de Atributos

Los siguientes métodos de jQuery actuarán únicamente sobre un atributo específico de los elementos seleccionados.

Método .html()

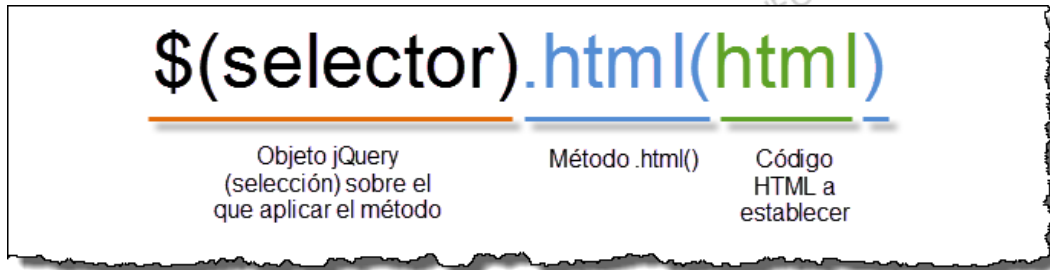
Con el método `.html()` obtendremos el contenido HTML del primer elemento en el conjunto de elementos pertenecientes a la selección sobre la que el método sea aplicado.

`$(selector).html()`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.html()`

Si el método recibe un parámetro, éste se establecerá como contenido HTML de todos los elementos de la selección.



HTML

Veamos un sencillo ejemplo de uso del método .html():

```
<div id="div_original">
  <ul id="internal_links">
    <li><a href="#destino1" target="_self">Destino Interno 1</a></li>
    <li><a href="#destino2" target="_blank">Destino Interno 2</a></li>
  </ul>
</div>
<div id="div_destino"></div>
```

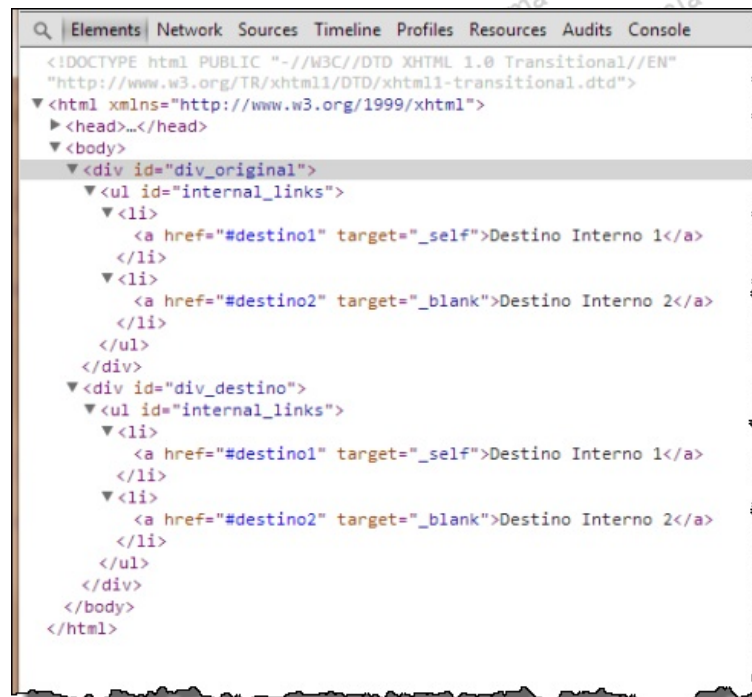
Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  htmlOriginal = $("#div_original").html();
  $("#div_destino").html(htmlOriginal);
});
</script>
```

Resultado

La primera llamada al método `html()` devolverá el código HTML del contenido del `div#div_original` y lo almacenará en la variable temporal `htmlOriginal`.

Con la segunda llamada, conseguimos establecer dentro del `div#div_destino` el HTML que habíamos almacenado en la variable `htmlOriginal`. Como resultado, los dos `divs` tendrán el mismo destino.



Método `.text()`

El método `.text()` obtendrá el texto contenido en los elementos seleccionados, combinando el texto existente en el propio elemento con el existente en todos los descendientes contenidos en el mismo.



Otro uso del método `.text()` será como establecedor. Si le proporcionamos un parámetro al método, éste lo utilizará para establecer como contenido de todos los elementos incluidos en la selección.

\$(selector).text(valor)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .text()

Texto a establecer
en los elementos
incluidos en la
selección

Es importante saber que este método escapará el contenido del parámetro, cuyo valor no será interpretado como HTML. También hay que tener en cuenta que este método no debe ser utilizado sobre elementos de tipo "input", para los que deberemos utilizar el método .val().

HTML

Veamos el funcionamiento de este método en el mismo escenario que el ejemplo anterior:

```
<div id="div_original">
  <ul id="internal_links">
    <li><a href="#destino1" target="_self">Destino Interno 1</a></li>
    <li><a href="#destino2" target="_blank">Destino Interno 2</a></li>
  </ul>
</div>
<div id="div_destino_text"></div>
<div id="div_destino_html"></div>
```

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  htmlOriginal = $("#div_original").html();
  textOriginal = $("#div_original").text();
  $("#div_destino_text").text(textOriginal);
  $("#div_destino_html").text(htmlOriginal);
});
</script>
```

Resultado

En las dos primeras líneas de nuestro código, obtenemos tanto el código HTML como el texto del `div#div_original`, para posteriormente utilizarlos con el método `.text()` como establecedor.

Al utilizar el método `.text()` como establecedor, conseguimos, tras la primera llamada al método con el parámetro `textOriginal`, establecemos como texto del `div#div_destino_text` este valor.

```
</div>
<div id="div_destino_text">
  "
  Destino Interno 1
  Destino Interno 2
  "
</div>
</body>
</html>
```

Del mismo modo, al llamarlo con el parámetro `htmlOriginal` el texto será establecido en el `div#div_destino_text`. Sin embargo, a diferencia de utilizar el método `.html()`, el texto no será interpretado como HTML.

```
</div>
<div id="div_destino_html">
  "
  <ul id="internal_links">
    <li><a href="#destino1" target="_self">Destino
    Interno 1</a></li>
    <li><a href="#destino2" target="_blank">Destino
    Interno 2</a></li>
  </ul>
  "
</div>
<div id="div_destino_html">
  "
```



Métodos `.html()` y `.text()`

Método `.val()`

El método `.val()` devolverá el valor del primer elemento de la selección sobre la que se aplique el método. Principalmente, este método se suele utilizar sobre elementos de formulario de tipo `input`, `select` y `textarea`, y nos permitirá obtener su valor en el momento en que se procese el código que haga uso del mismo.

`$(selector).val()`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.val()`

HTML

Vamos a utilizar una vez más el código HTML del formulario de búsqueda ligeramente modificado:

```
<div id="srchContent">
  <form id="srchForm" type="get" action="#">
    <input id="srchText" name="srchText" type="text" value="Indique el texto a buscar" />
  </form>
</div>
```

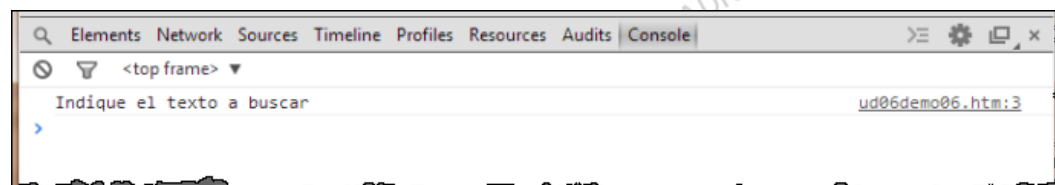
Por medio del método `.val()` vamos a mostrar por consola el valor .

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  console.log( $("#srchText").val() );
});
</script>
```

Resultado

Al llamar al método `.val()` sobre la selección, éste devolverá la cadena de texto correspondiente al valor de `input#srchText`, siendo éste el valor mostrado por consola.



El método `.val()` también podrá funcionar como establecedor, dándonos la posibilidad de definir el valor para este atributo simplemente proporcionándole al método un parámetro con el valor que deseemos asignar.

\$(selector).val(valor)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .val()

Nuevo valor a
establecer en los
elementos incluidos en
la selección

HTML

Volvamos al código HTML del anterior ejemplo:

```
<div id="srchContent">
  <form id="srchForm" type="get" action="#">
    <input id="srchText" name="srchText" type="text" value="Indique el texto a buscar" />
  </form>
</div>
```

En esta ocasión vamos a utilizar el método .val() para establecer un nuevo valor en el campo de búsqueda.

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  $("#srchText").val("Nuevo Valor");
  console.log( $("#srchText").val() );
});
</script>
```

Resultado

Al aplicar el método .val() como establecedor conseguiremos sobrescribir el valor del campo #srchText con el valor "Nuevo Valor".





Método .val()

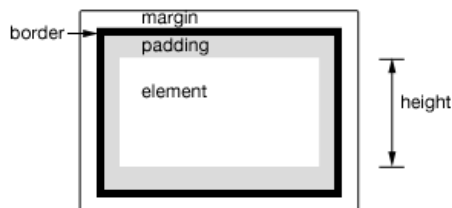
Método .height()

Con el método .height() obtendremos la altura del primer elemento del conjunto de elementos seleccionado.

`$(selector).height()`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .height()



Es recomendable utilizar el método .height() cuando vayamos a utilizar el valor devuelto para realizar cálculos matemáticos ya que, a diferencia de la propiedad "height" de CSS, el valor devuelto no contendrá el tipo de medida (por ejemplo, "px").

`$(selector).height(valor)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .height()

Valor a establecer
como altura en los
elementos incluidos en
la selección

Para que el método funcione como establecedor, haremos la llamada al mismo con un parámetro adicional. De este modo, la altura para todos los elementos contenidos en la selección se establecerá con este valor. El parámetro podrá ser un número, en cuyo caso se utilizará como medida el píxel (px) o una cadena, que deberá contener un número junto con una medida (px, em o %).

HTML

Volvamos al código HTML visto en los ejemplos anteriores:

```
<div class="all">
  <div id="header">Header de la p&aacute;gina</div>
  <div id="content">Contenido<br />de la p&aacute;gina</div>
</div>
```

Vamos a utilizar un selector de comparación por comienzo del atributo "title", en concreto para los elementos cuyo título comience con el texto "Publicar".

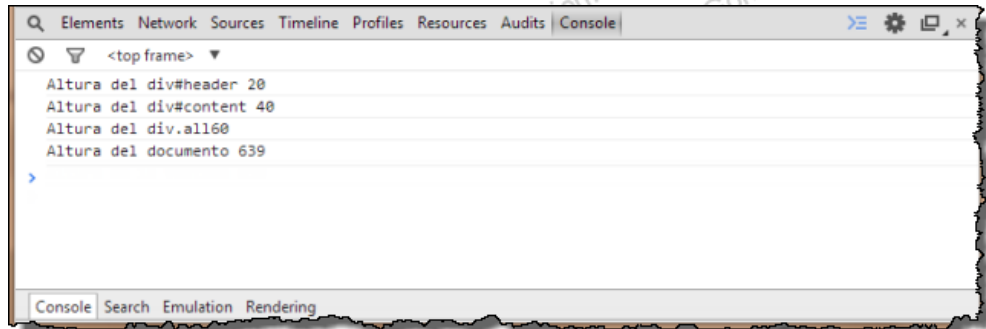
Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  console.log( 'Altura del div#header ' + $("#header").height() );
  console.log( 'Altura del div#content ' + $("#content").height() );
  console.log( 'Altura del div.all ' + $(".all").height() );

  console.log( 'Altura del documento ' + $(document).height() );
});
</script>
```

Resultado

La consola del navegador nos mostrará la altura de los elementos consultados. Así, veremos la altura de los tres elementos de tipo div del contenido, además de la altura del documento.



Si en alguna de las llamadas al método hubiésemos indicado un parámetro, la altura del elemento sobre el que se hubiese aplicado tomaría el nuevo valor.

Método .innerHeight()

El método .innerHeight() es una adaptación del método .height(). En este caso, el método solamente funcionará como obtenedor, y el valor devuelto será el correspondiente a la altura del elemento más la altura del relleno (height + padding).

`$(selector).innerHeight()`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.innerHeight()`

Método `.outerHeight()`

El último método relativo a la altura es el método `.outerHeight()`. Al igual que el método `.innerHeight()` este método solamente realizará las funciones de obtenedor, devolviendo en su caso la suma de la altura, el relleno, borde y margen del elemento (`height + padding + borde + margin`).

`$(selector).outerHeight()`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.outerHeight()`

Método `.width()`

El método `.width()` nos devolverá la anchura del primer elemento del conjunto de elementos seleccionado.

`$(selector).width()`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.width()`

Al igual que el método `.height()`, éste podrá actuar como establecedor, para modificar el ancho de los elementos de la selección. Para ello, haremos una llamada al método indicando la nueva anchura por medio de un parámetro. El parámetro podrá ser un número, en cuyo caso se utilizará como medida el píxel (px) o una cadena, que deberá contener un número junto con una medida (px, em o %).

`$(selector).width(valor)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.width()`

Anchura a establecer
en los elementos
incluidos en la
selección



Si el elemento del DOM al que intentamos modificar este atributo tiene establecidas la propiedad CSS max-width o min-width, el cambio en la anchura se aplicará como máximo (o mínimo) al valor indicado en estas propiedades.

HTML

Vamos a ver cómo funciona este método, utilizándolo tanto en su función de obtenedor como en su función de establecedor:

```
<div class="all">
  <div id="header">Header de la página</div>
  <div id="content">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et
    dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
    Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
    Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
  </div>
</div>
```

También vamos a disponer de los siguientes estilos CSS que se aplican sobre los dos div con identificador:

```
#header{
  width: 120px;
  max-width: 160px;
}
#content{
  width: 220px;
}
```

Código de Ejemplo

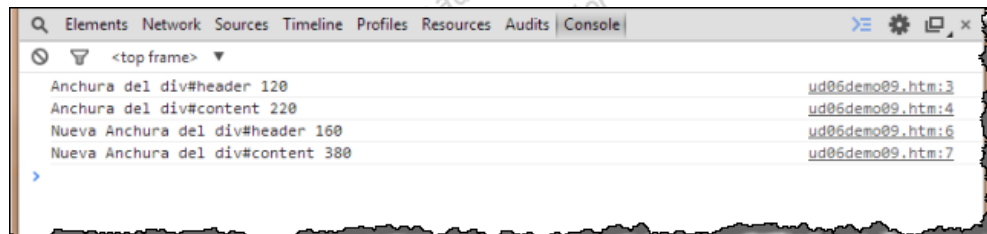
```
<script type="text/javascript">
$(document).ready(function(){
  console.log( 'Anchura del div#header ' + $("#header").width() );
  console.log( 'Anchura del div#content ' + $("#content").width() );

  $("#content, #header").width(380);

  console.log( 'Nueva Anchura del div#header ' + $("#header").width() );
  console.log( 'Nueva Anchura del div#content ' + $("#content").width() );
});
</script>
```

Resultado

Si analizamos la respuesta por consola, vemos que en las dos primeras llamadas al método `.header()` éste funciona como obtenedor, devolviendo la anchura en cada momento que tienen tanto el `div#header` como el `div#content`. En su tercera llamada, intentamos establecer la anchura de ambos `div` a 380 píxeles, pero vemos que solamente el `div#content` es modificado hasta lograr esta anchura, mientras que para el `div#header` este atributo se modifica hasta el máximo indicado en el atributo `max-width` de CSS.

**Método `.innerWidth()`**

El método `.innerWidth()` solamente funcionará como obtenedor. El valor devuelto por el método será el correspondiente a la anchura del elemento más la anchura del relleno (`width + padding`).

`$(selector).innerWidth()`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.innerWidth()`

Método `.outerWidth()`

Del mismo modo que el método `.innerWidth()`, el método `.outerWidth()` solamente realizará las funciones de obtenedor. Este método devolverá la suma de la anchura del elemento, el relleno, borde y margen del elemento (`width + padding + borde + margin`).

`$(selector).outerWidth()`

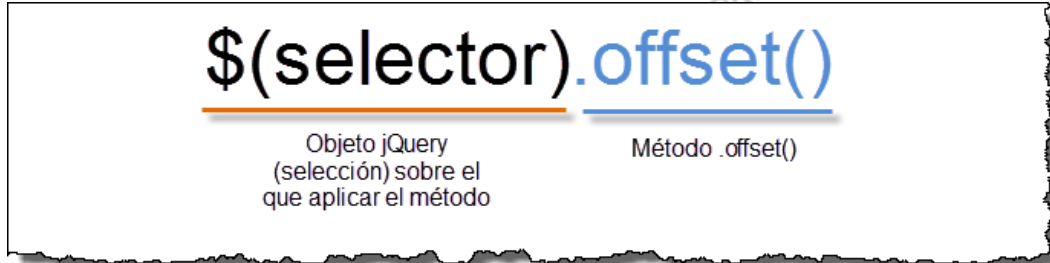
Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.outerWidth()`

Método `.offset()`

El método `.offset()` devolverá las coordenadas del primer elemento de la selección sobre la que se aplique, dando como resultado un objeto con las propiedades "left" y "top".

Las coordenadas devueltas obtendrán su valor respecto a la posición del elemento en el documento.



Este método también funcionará como establecedor, cuando se le proporcione como parámetro un objeto anónimo con las propiedades "top" y "left", que serán usadas para asignar la posición horizontal y vertical de todos los elementos del DOM contenidos en la selección.



Objeto anónimo con las propiedades top y left indicando el valor de cada una de ellas

Método `.position()`

Al igual que `.offset()`, el método `.position()` también devolverá las coordenadas del primer elemento de la selección sobre la que se aplique (devolviendo un objeto con las propiedades "left" y "top"), pero en esta ocasión, los datos devueltos serán relativos al elemento padre del mismo. Sin embargo, este método no podrá funcionar como establecedor.



HTML

Volvamos al código HTML visto en los ejemplos anteriores:

```
<div id="document_links">
  <ul id="internal_links">
    <li><a href="#destino1" target="_self">Destino Interno 1</a></li>
    <li><a href="#destino2" target="_blank">Destino Interno 2</a></li>
  </ul>
  <ul id="external_links">
    <li><a href="#destino_externo1" target="_self">Destino Externo 1</a></li>
    <li><a href="#destino_externo2" target="_blank">Destino Externo 2</a></li>
    <li><a href="#destino_externo3" target="_blank">Destino Externo 3</a></li>
  </ul>
</div>
```

Vamos a llamar al método `.position()` sobre el `ul#external_links` para ver cómo la librería nos devolverá un objeto con las propiedades "left" y "top".

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  console.log( $("#external_links").position() );
});
</script>
```

Resultado

Como resultado, vemos que la consola muestra un objeto con los atributos "top" y "left" cuyos valores se corresponden a las coordenadas de posición horizontal y vertical del `ul#external_link`

Método `.scrollLeft()`

El método `.scrollLeft()` funcionará tanto de obtenedor como de establecedor, permitiendo según la forma en la que sea llamado bien obtener la posición en la que se encuentre el scroll horizontal de la página para el primer elemento de la selección, o estableciendo este atributo para todos los elementos de la selección sobre la que el método sea aplicado.

`$(selector).scrollLeft()`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.scrollLeft()`

`$(selector).scrollLeft(valor)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.scrollLeft()`

Posición a
establecer en el
scroll horizontal
de los elementos

Método `.scrollTop()`

Con el método `.scrollTop()` podremos realizar las mismas acciones que con el método `.scrollLeft()` pero actuando sobre el scroll vertical en vez de frente al horizontal.

`$(selector).scrollTop()`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.scrollTop()`

`$(selector).scrollTop(valor)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.scrollTop()`

Posición a
establecer en el
scroll vertical de
los elementos

Clases y Estilos CSS

Método `.addClass()`

El método `.addClass()` nos permitirá añadir una o varias clases a los elementos seleccionados. Este método podrá funcionar de dos formas, pudiendo recibir como parámetro tanto una cadena de texto como una función.



Una de las formas de utilizar este método es proporcionándole una cadena de texto, en la que indicaremos la clase (o clases, separadas por un espacio) que deseemos añadir a los elementos de la selección.

HTML

Vamos a ver el funcionamiento del método en un ejemplo de HTML muy sencillo:

```
<div id="sinEstilo">
</div>
```

Para este elemento, vamos a llamar varias veces al método `.addClass` de diferentes modos para observar los efectos que éste tendrá sobre el elemento del DOM.

Código de Ejemplo

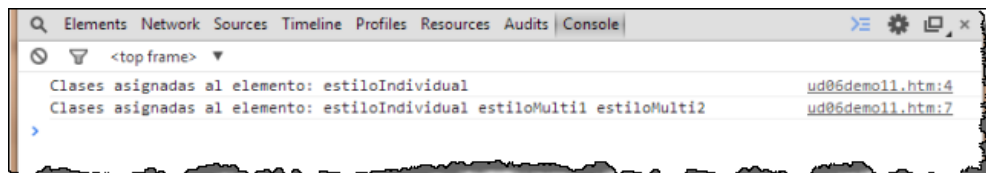
```
<script type="text/javascript">
$(document).ready(function(){
  $("#sinEstilo").addClass("estiloIndividual");
  console.log( "Clases asignadas al elemento: " + $("#sinEstilo").attr("class") );

  $("#sinEstilo").addClass("estiloMulti1 estiloMulti2");
  console.log( "Clases asignadas al elemento: " + $("#sinEstilo").attr("class") );
});
</script>
```

Resultado

La primera llamada al método `.addClass()` añadirá una única clase al elemento. Así se muestra por consola que el elemento del DOM solamente tiene asignada la clase "estiloIndividual".

En la segunda llamada, se añaden dos clases adicionales, las clases "estiloMulti1" y "estiloMulti2".



`$(selector).addClass(function(){})`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.addClass()`

Función que para
cada elemento
devolverá la clase a
añadir

La otra forma de llamar a este método es con una función anónima. Esta función recibirá en cada llamada dos parámetros: el índice correspondiente al elemento con el que se le está llamando (en cada una de las iteraciones) y el valor del atributo "class" para el correspondiente elemento. Como resultado de llamada a la función, ésta deberá devolver una cadena de texto que contendrá las clases, separadas por espacio, a añadir al elemento de la iteración.

Método `.hasClass()`

Con el método `.hasClass()` podremos determinar si alguno de los elementos de la selección sobre la que se aplique el método tienen asignada una determinada clase.

`$(selector).hasClass(clase)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.hasClass()`

Clase a comprobar si
se encuentra
establecida en el
elemento seleccionado

Este método devolverá el valor `true` si alguno de los elementos de la selección tiene asignada la clase indicada como parámetro, o `false` si ninguno de los elementos tiene asignada la clase.

HTML

En esta ocasión también vamos a ver un ejemplo muy sencillo, con un fragmento de código HTML muy reducido y en el que solamente tendremos dos elementos:

```
<div id="header">
</div>
<div id="content" class="mainContent">
</div>
```

Vamos a utilizar un selector de comparación por comienzo del atributo "title", en concreto para los elementos cuyo título comience con el texto "Publicar".

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  $seleccion1 = $("#header");
  $seleccion2 = $("#content");
  if( $seleccion1.hasClass('mainContent') ){
    console.log("El div #" + $seleccion1.attr('id') + " sí tiene asignada la clase");
  }
  if( $seleccion2.hasClass('mainContent') ){
    console.log("El div #" + $seleccion2.attr('id') + " sí tiene asignada la clase");
  }
});
</script>
```

Resultado

Al ejecutar el código, la consola mostrará un único mensaje, indicando que el elemento incluido en el segundo selector es el que tiene asignada la clase `.mainContent` y, con la ayuda del método `.attr('id')` componemos un mensaje que muestra el atributo "id" del elemento que tiene la clase asignada.

Método `.removeClass()`

El método `.removeClass()` nos permitirá eliminar una o varias de las clases de los elementos incluidos en el objeto sobre el que se aplica el método.

`$(selector).removeClass()`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.removeClass()`

Si al método le proporcionamos un parámetro, éste deberá contener la clase (o clases, separadas por un espacio) a eliminar de los elementos de la selección. En cambio, si no se le indica un parámetro, todas las clases serán removidas de todos los elementos.

\$(selector).removeClass(clase)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .removeClass()

Clase a
desasignar a los
elementos de la
selección

HTML

Volvamos al ejemplo visto en el método .addClass() :

```
<div id="header">
</div>
<div id="content" class="mainContent">
</div>
```

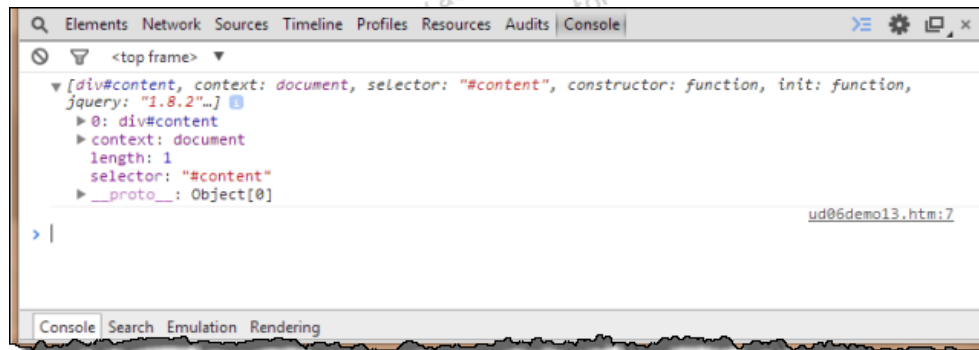
Una vez comprobemos que el elemento seleccionado tiene asignada la clase, vamos a proceder a "desasignarla" para dejar el atributo sin ninguna clase de estilo.

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  $seleccion2 = $("#content");
  if( $seleccion2.hasClass('mainContent') ){
    $seleccion2.removeClass('mainContent');
    //para esta caso también nos valdría la llamada $seleccion2.removeClass()
    console.log($seleccion2);
  }
});
</script>
```

Resultado

Puesto que el objeto disponía de la clase asignada, la llamada al método removerá esta clase del mismo. Al mostrar la selección por consola, veremos que el div ya no tiene la clase asignada.



Métodos `.addClass()` `.hasClass()` y `.removeClass()`

Método `.toggleClass()`

El método `.toggleClass()` realizará una transmutación de clase a los elementos de la selección. En otras palabras, asignará (si no la tienen asignada) o desasignará (si la tuviesen asignada) la clase proporcionada como parámetro al método sobre todos los elementos de la selección. Si un elemento tenía asignada la clase, ésta será desasignada y viceversa.

`$(selector).toggleClass(clase)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.toggleClass()`

Clase a
transmutar en los
elementos de la
selección

Método `.css()`

El método `.css()` permitirá obtener el valor de una propiedad CSS del primer elemento de la selección sobre la que se aplique, o establecerá una o varias propiedades CSS sobre todos los elementos de la selección.

Funcionamiento como obtenedor

Este método podrá funcionar como obtenedor de dos formas diferentes. La primera y más habitual es proporcionándole, por medio de un parámetro, la propiedad CSS a obtener.



Otra opción para hacer uso del método es pasándole un array como parámetro, conteniendo éste las diferentes propiedades que deseemos obtener. En este caso, el valor devuelto por el método será así mismo un array, que contendrá los valores para cada una de las propiedades indicadas.

HTML

Volvamos al código HTML visto en los ejemplos anteriores:

```
<div id="header">
</div>
```

En este caso, nuestra página también va a tener la siguiente definición de estilo:

```
#header{
  float: right;
  max-width: 250px;
}
```

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  console.log($("#header").css("maxWidth"));
  console.log($("#header").css("max-width"));
});
</script>
```

Resultado

Las dos llamadas al método realizarán la acción. Para poder obtener la propiedad CSS `max-width` (así como todas aquellas que contengan el símbolo `-` en su nombre) podremos hacer uso de esta cadena con el selector o escrita en modo `"CamelCase"` (y más concretamente en formato `lowerCamelCase`)

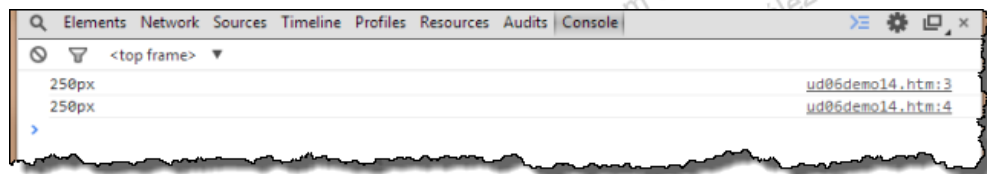


CamelCase es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en CamelCase se asemejan a las jorobas de un camello. El nombre CamelCase se podría traducir como Mayúsculas/Minúsculas Camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula y tiene su origen en la disposición de los tipos móviles en casilleros o cajas. Existen dos tipos de CamelCase:

- UpperCamelCase, cuando la primera letra de cada una de las palabras es mayúscula. Ejemplo: EjemploDeUpperCamelCase.
- lowerCamelCase, igual que la anterior con la excepción de que la primera letra es minúscula. Ejemplo: ejemploDeLowerCamelCase.

Fuente: Wikipedia

Así, la propiedad es mostrada por consola en dos ocasiones.



Funcionamiento como establecedor

Del mismo modo que otros métodos de la librería, éste podrá también actuar como establecedor, permitiéndonos así modificar propiedades CSS de todos los elementos incluidos en la selección.

Para que el método funcione como establecedor, le proporcionaremos dos parámetros: el primero, indicará la propiedad CSS a establecer, y el segundo, le indicará al método el nuevo valor a establecer en la propiedad.



Otra forma de utilizar el método como establecedor será proporcionándole como parámetro un objeto anónimo. De este modo podremos modificar no sólo una sino varias propiedades CSS de los objetos incluidos en la selección, estableciendo cada una de las propiedades definidas en el objeto sobre los elementos seleccionados.

HTML

Vamos a utilizar el mismo fragmento de HTML del ejemplo anterior:

```
<div id="header">  
</div>
```

También vamos a mantener la misma definición de estilo:

```
#header{  
  float: right;  
  max-width: 250px;  
}
```

Código de Ejemplo

```
<script type="text/javascript">  
$(document).ready(function(){  
  $("#header").css("maxWidth", "350px");  
  $("#header").css( {"height": "20px", "font-size": "13px"});  
  
  console.log($("#header").css("maxWidth"));  
  console.log($("#header").css("height"));  
  console.log($("#header").css("fontSize"));  
});  
</script>
```

Resultado

Como resultado, se modificará el atributo CSS "max-width" del elemento y se asignarán los atributos "height" y "font-size". Así, al llamar al método como obtenedor, la consola nos mostrará los valores asignados a cada uno de estos atributos.



Método .css()

Inserción y copiado de elementos del DOM

Creación de elementos con la función \$()

Crear elementos dinámicamente con la función \$()

Hasta ahora hemos utilizado la función `$()` como selector, para localizar entre los elementos del DOM un subconjunto de elementos dependiendo del selector indicado en la función.

Ahora vamos a ver su uso para la creación de elementos "al vuelo" a partir de una cadena de código HTML.

```
var $div = $("<div/>");
```

```
var $div2 = $("<div>Elemento de tipo DIV</div>");
```

Este funcionamiento se conseguirá siempre y cuando la cadena de código HTML se componga principalmente por un elemento, aunque en ésta podremos incluir también el HTML de los descendientes del mismo, así como cualquiera de sus atributos.

```
var $div3 = $("<div id='dynamicDiv'> Elemento de tipo DIV <span>con un SPAN</span></div>");
```



Al hacer una llamada a estas funciones, los elementos del DOM son creados, pero no posicionados dentro del árbol del DOM del documento. Será necesario "insertarlos" en algún punto del documento para que formen parte de éste.

```
$( "<div>DIV creado al vuelo</div>" ).appendTo( "body" )
```

Otra forma de utilizar esta función es enviándole dos parámetros. En este caso, el primero de ellos deberá contener el HTML del elemento a crear, y en el segundo proporcionaremos un objeto anónimo con todas las propiedades que queramos establecer en el nuevo elemento:

```
$( "<div />", {id:"newDiv", class:"dynamic", text:"Este es un nuevo DIV creado al vuelo con propiedades"} );
```

Como resultado, se creará un elemento con el siguiente código:

```
<div id="newDiv" class="dynamic">Este es un nuevo DIV creado al vuelo con propiedades</div>
```

Esta llamada será equivalente a las siguientes:

```
$( "<div id='newDiv' class='dynamic'> Este es un nuevo DIV creado al vuelo con propiedades </div> " );
```

```
$( "<div />" )
  .attr("id","newDiv")
  .addClass("dynamic")
  .text("Este es un nuevo DIV creado al vuelo con propiedades");
```

Como puedes ver, jQuery nos ofrece de gran versatilidad para la realización de ciertos procesos. Es interesante conocerlos aunque en el uso diario puedes utilizar el que mayor comodidad te ofrezca.

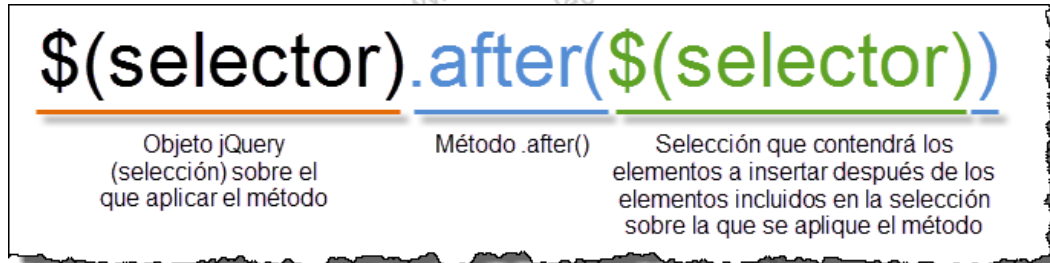
Creación de elementos con métodos específicos

Método .after()

El método .after() insertará el HTML proporcionado como parámetro después de cada uno de los elementos existentes en la selección sobre la que se aplique.



Otra forma de llamar al método será proporcionarle como parámetro un selector que, en caso de localizar elementos en el DOM cumpliendo con sus especificaciones, serán movidos a la posición posterior. Si la selección original contiene dos ó mas elementos, el objeto indicado como parámetro será movido a la posición posterior del primer elemento de la selección, y se crearán nuevas copias para los elementos restantes de la selección.



HTML

Vamos a ver cómo se comporta este método frente a un fragmento sencillo de HTML con el que entender su funcionamiento.

```
<div id="urls">
  <span class="listHead">Lista de Urls:</span>
</div>
<div id="controls">
  <div class="urlTemplate" style="display: none;">
    <label>URL</label> <input type="text">
  </div>
  <input type="button" id="btnAddUrl" value="+">
</div>
```

Código de Ejemplo

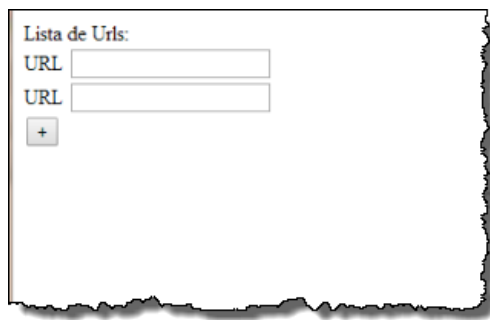
```

<script type="text/javascript">
$(document).ready(function(){
    $("#btnAddUrl").on("click", function(e){
        e.preventDefault();
        $("#urls .listHead").after( $("#controls .urlTemplate").clone() );
        $("#urls .urlTemplate").removeAttr("style");
    });
});
</script>

```

Resultado

En este ejemplo capturamos el evento click del botón #btnAddUrl y, cada vez que es pulsado, insertamos después del elemento span.listHead una copia (obtenida con la llamada al método .clone()) del div.urlTemplate existente dentro del div#controls . El motivo por el que llamamos al método .clone() es porque de lo contrario, en vez de realizarse una copia, se insertaría el propio div.urlTemplate y el botón solamente funcionaría la primera vez que se pulsara.



Una vez el elemento es copiado e insertado después del span.listHead llamamos al método .removeAttr("style") para que el estilo "display: none;" deje de aplicarse sobre el nuevo div.

Otra forma de haber conseguido este mismo funcionamiento es invocando al método .after() del siguiente modo:

```

$("#urls .listHead").after( '<div class="urlTemplate"><label>URL</label> <input type="text"></div>' );

```

Si hubiésemos optado por esta opción, podríamos eliminar del documento el div.urlTemplate así como realizar la llamada al método .removeAttr() para eliminar el estilo "display:none;"

Método .insertAfter()

Con el método .insertAfter() podremos insertar todos los elementos incluidos en el objeto jQuery sobre el que se aplique después del elemento indicado al método como elemento destino.

Para definir el destino en el que copiar, el método recibirá un parámetro que podrá corresponderse a uno de los siguientes tipos de elementos:

- Un selector
- Un elemento del DOM
- Un objeto jQuery que contenga el elemento destino

`$(selector).insertAfter($(selector))`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.insertAfter()

Selección de elementos
después de los que insertar
los elementos de la selección
sobre la que se aplique el
método

`$(selector).insertAfter(selector)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.insertAfter()

Selector que definirá los
elementos después de los que
insertar los elementos de la
selección sobre la que se
aplique el método



El método .insertAfter() realizará una tarea similar al método .after(). La diferencia entre ambos radica en que, en el método .after() se aplicará sobre el destino y se le indicará como parámetro el objeto a mover / insertar, mientras que en el método .insertAfter() estas posiciones se verán invertidas.

HTML

Vamos a utilizar el ejemplo visto con el método .after() modificando su código para obtener el mismo resultado con el método .insertAfter().

```
<div id="urls">
  <span class="listHead">Lista de Urls:</span>
</div>
<div id="controls">
  <div class="urlTemplate" style="display: none;">
    <label>URL</label> <input type="text">
  </div>
  <input type="button" id="btnAddUrl" value="+ ">
</div>
```

Código de Ejemplo

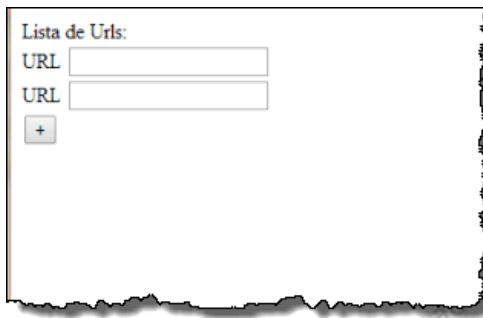
```

<script type="text/javascript">
$(document).ready(function(){
    $("#btnAddUrl").on("click", function(e){
        e.preventDefault();
        $newControl = $("#controls .urlTemplate").clone();
        $newControl.insertAfter( $("#urls .listHead") );
        $("#urls .urlTemplate").removeAttr("style");
    });
});
</script>

```

Resultado

En este caso, almacenamos la copia del div `.urlTemplate` en una variable, sobre la que posteriormente llamaremos al método `.insertAfter()` indicándole como parámetro la selección del div donde deseamos insertar el nuevo elemento. Una vez más, llamamos al método `.removeAttr("style")` para dejar de aplicar el estilo `"display: none;"` al nuevo div.



Al igual que en el ejemplo anterior, podemos lograr el mismo resultado utilizando la función `$()` para crear el elemento dinámicamente. Si optásemos por esta solución nuestro código quedaría así:

```

$('<div class="urlTemplate"><label>URL</label> <input type="text"></div>').insertAfter( '#urls .listHead' );

```

Método .before()

El método `.before()` insertará en el DOM los elementos proporcionados como parámetro al método antes de cada uno de los elementos incluidos en el objeto jQuery sobre el que se aplique.

`$(selector).before(html)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .before()

HTML a insertar
antes de todos
los elementos de
la selección

El parámetro proporcionado podrá ser un fragmento de HTML que defina los elementos a introducir en el DOM o bien un selector que, en caso de localizar algún elemento en el DOM que cumpla con sus especificaciones, moverá éstos a la posición anterior de los elementos de origen.

Si la selección sobre la que se aplica el método incluye dos o más elementos, los elementos de la selección pasada como parámetro se moverán delante del primer elemento de éstos y se crearán nuevas copias para los demás elementos de la selección.

`$(selector).before($(selector))`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .before()

Selección que contendrá los
elementos a insertar antes de
los elementos incluidos en la
selección sobre la que se
aplique el método

HTML

En este ejemplo vamos a disponer del siguiente fragmento de HTML;

```
<div class="listContainer">
  <ul id="listaFrutas">
    <li><span class="nombreFruta">Manzanas</span></li>
    <li><span class="nombreFruta">Peras</span></li>
    <li><span class="nombreFruta">Piñan</span></li>
  </ul>
  <div class="listControls">
    <input type="button" id="btnAddFruit" value="Añadir Fruta">
  </div>
</div>
```

Tenemos una lista de frutas, así como un botón con el que poder añadir elementos a la lista. Al pulsar el botón, el navegador nos mostrará una ventana emergente en la que podremos indicar el nombre de una fruta a añadir a la lista. Si indicamos un nombre, ese nuevo nombre se añadirá al comienzo de la lista.

Código de Ejemplo

```

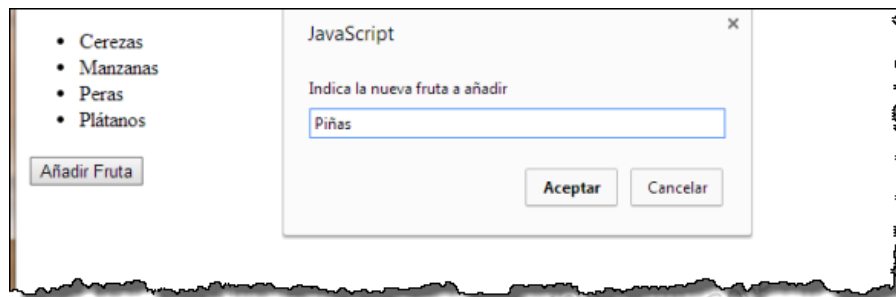
<script type="text/javascript">
$(document).ready(function(){
    $("#btnAddFruit").on("click", function(e){
        e.preventDefault();

        var nuevaFruta = prompt("Indica la nueva fruta a añadir","");
        if(nuevaFruta){
            $("#listaFrutas li:first").before("<li>" + nuevaFruta + "</li>");
        }
    });
});
</script>

```

Resultado

Cada vez que pulsemos el botón, aparecerá un mensaje emergente en el que podremos introducir el nombre del elemento a añadir a la lista. Si en este emergente indicamos un valor, nuestra lista será ampliada, insertando al comienzo de la misma un nuevo elemento con el valor que indiquemos.



Método .insertBefore()

El método .insertBefore() insertará todos los elementos del objeto jQuery sobre el que se aplique delante del elemento "destino", proporcionado como parámetro al método.

\$(selector)	.insertBefore()	\$(selector)
Objeto jQuery (selección) sobre el que aplicar el método	Método .insertBefore()	Selección de elementos destinatarios en los que insertar los elementos incluidos en la selección sobre la que se aplique el método

\$(selector).insertBefore(selector)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.insertBefore()

Selector que definirá los
elementos destinatarios en los
que insertar los elementos
incluidos en la selección sobre
la que se aplique el método

HTML

En este ejemplo vamos a transformar el ejemplo visto para el método .before() y, en vez de este método, utilizar el método .insertBefore()

```
<div class="listContainer">
  <ul id="listaFrutas">
    <li><span class="nombreFruta">Manzanas</span></li>
    <li><span class="nombreFruta">Peras</span></li>
    <li><span class="nombreFruta">Piñanitos</span></li>
  </ul>
  <div class="listControls">
    <input type="button" id="btnAddFruit" value="Añadir Fruta">
  </div>
</div>
```

En este script vamos a tener dos de los componentes vistos en un ejemplo anterior. Vamos a ejecutar un fragmento de código que va a hacer que estos dos elementos sean "envueltos" por un div que los contenga, y así conseguir el mismo escenario que teníamos en anteriores ocasiones.

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  $("#btnAddFruit").on("click", function(e){
    e.preventDefault();

    var nuevaFruta = prompt("Indica la nueva fruta a añadir", "");
    if(nuevaFruta){
      $("<li>" + nuevaFruta + "</li>").insertBefore( $("#listaFrutas li:first") );
    }
  });
});
</script>
```

Resultado

Del mismo modo que ocurría en el ejemplo anterior, el nuevo elemento li se insertará delante del primer elemento li contenido en el ul#listaFrutas.

Una vez más, hemos evidenciado que con jQuery no solo existe una forma de realizar una acción, sino que disponemos de diversas formas de realizar una acción sobre los elementos del DOM.



Métodos .before() e .insertBefore()

Método .append()

Con el método .append() podremos insertar los elementos especificados como parámetro al método (bien indicando el HTML del nuevo elemento, un selector o un objeto jQuery) al final de cada uno de los elementos resultantes de la selección sobre la que se aplique.

\$(selector).append(html)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .append()

HTML a insertar al final
de los elementos
existentes en la
selección sobre la que
se aplique el método

\$(selector).append(selector)

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .append()

Selector que definirá los
elementos a insertar al final de
los elementos existentes en la
selección sobre la que se
aplique el método

`$(selector).append($(selector))`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método `.append()`

Selección de elementos a
insertar al final de los
elementos existentes en la
selección sobre la que se
aplique el método

Del mismo modo que ocurría con el método `.after()`, si la selección sobre la que se aplique el método contiene un único elemento, el elemento indicado como parámetro será establecido como último hijo de éste, mientras que si la selección contiene más de un elemento, el objeto indicado como parámetro se establecerá como último hijo del primer elemento de la selección y en los demás elementos unas copias de éste ocuparán la misma posición.

`.appendTo()`

Este método insertará los elementos incluidos en la selección sobre la que se aplique al final del elemento indicado como parámetro.

`$(selector).appendTo($(selector))`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
`.appendTo()`

Selección de elementos en los
que insertar los elementos
existentes en la selección
sobre la que se aplique el
método

`$(selector).appendTo(selector)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
`.appendTo()`

Selector que definirá el
elemento en el que insertar los
elementos existentes en la
selección sobre la que se
aplique el método



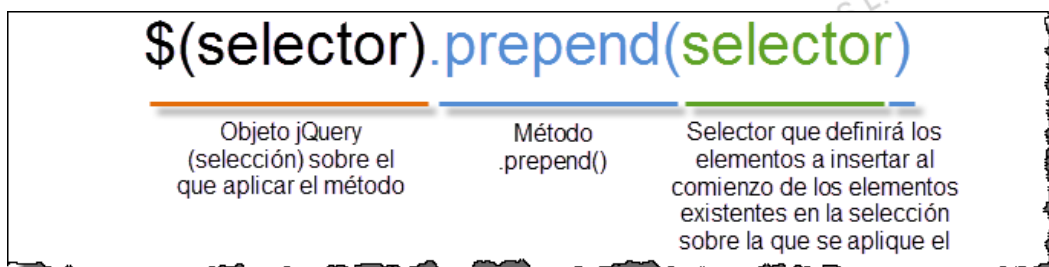
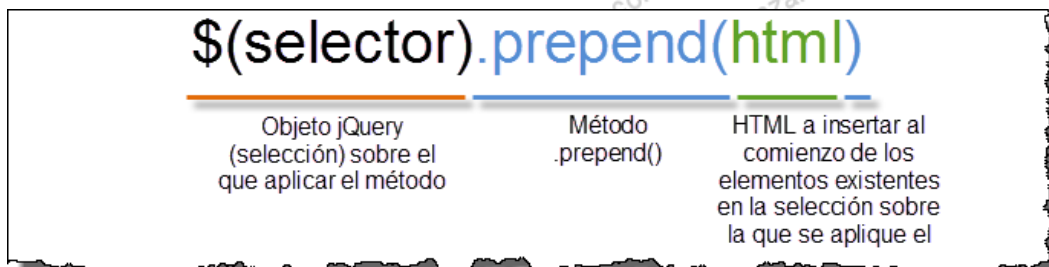
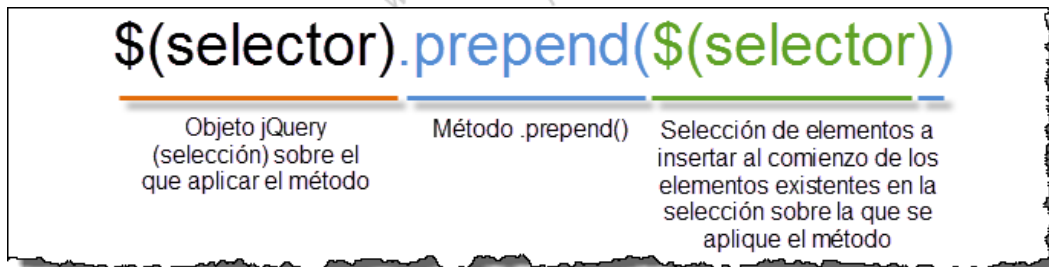
Este resultado del método será muy similar en su funcionamiento al método `.append()`. La principal diferencia entre ambos métodos radica en que, en el método `.append()`, los elementos de destino serán los existentes en la selección sobre la que se aplique, mientras que en el método `.appendTo()` serán indicados como parámetro al método.

Método `.prepend()`

Con el método `.prepend()` podremos insertar el elemento especificado como parámetro al comienzo de cada uno de los elementos incluidos en la selección sobre la que el método sea aplicado. Podremos proporcionar como parámetro tantos elementos como queramos insertar al comienzo de los elementos de la selección, indicando éstos como diferentes parámetros separados por comas.

Los elementos a insertar podrán ser proporcionados de diferente modo:

- Como una cadena de código HTML que defina el elemento a insertar
- Como un objeto jQuery (selección)
- Como un elemento del DOM
- Como una función anónima que devuelva uno de los tres tipos de elemento anteriormente indicados



Método `.prependTo()`

Con el método `.prependTo()` y `.prepend()` ocurre algo similar a la relación entre los métodos `.appendTo()` y `.append()`, ya que ambos métodos realizarán la misma acción pero intercambiando los elementos origen / destino.

`$(selector).prependTo($(selector))`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.prependTo()

Selección de elementos en los
que insertar, al comienzo de
los mismos, los elementos
incluidos en la selección sobre
la que se aplique el método

`$(selector).prependTo($(selector))`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.prependTo()

Selección de elementos en los
que insertar, al comienzo de
los mismos, los elementos
incluidos en la selección sobre
la que se aplique el método

El método .prepend(), se aplica sobre la selección destino, y se le proporcionarán como parámetro los elementos a insertar. Sin embargo, el método .prependTo() se aplicará sobre los elementos a insertar, y se le indicará como parámetro el destino en el que insertar los elementos.

Método .wrap()

Con el método .wrap() podremos "envolver" los elementos incluidos en la selección sobre la que apliquemos el método con un elemento HTML, cuya estructura indicaremos como parámetro al método.

`$(selector).wrap(html)`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método .wrap()

Código HTML con el que
envolver cada uno de los
elementos contenidos en
la selección

HTML

Vamos a ver un ejemplo de cómo utilizar el método `.wrap()` de jQuery;

```
<ul id="listaFrutas">
  <li><span class="nombreFruta">Manzanas</span></li>
  <li><span class="nombreFruta">Peras</span></li>
  <li><span class="nombreFruta">Plátanos</span></li>
</ul>
<div class="listControls">
  <input type="button" id="btnAddFruit" value="Añadir Fruta">
</div>
```

En este script vamos a tener dos de los componentes vistos en un ejemplo anterior. Vamos a ejecutar un fragmento de código que va a hacer que estos dos elementos sean "envueltos" por un div que los contenga, y así conseguir el mismo escenario que teníamos en anteriores ocasiones.

Código de Ejemplo

```
<script type="text/javascript">
$(document).ready(function(){
  $("body *").wrap('<div class="listContainer">');
});
</script>
```

Resultado

Al ejecutarse el código, los dos elementos de la página verán cambiado su elemento padre, pasando de ser éste el elemento body de la página a ser un nuevo elemento creado a partir del código HTML que indicamos en la llamada al método.

--imagen

Método `.wrapAll()`

El método `.wrapAll()` envolverá todos los elementos incluidos en la selección con un único elemento, que pasará a ser el elemento padre común de todos ellos.

<code>\$(selector).wrapAll(html)</code>		
Objeto jQuery (selección) sobre el que aplicar el método	Método <code>.wrapAll()</code>	Código HTML con el que envolver todos los elementos contenidos en la selección

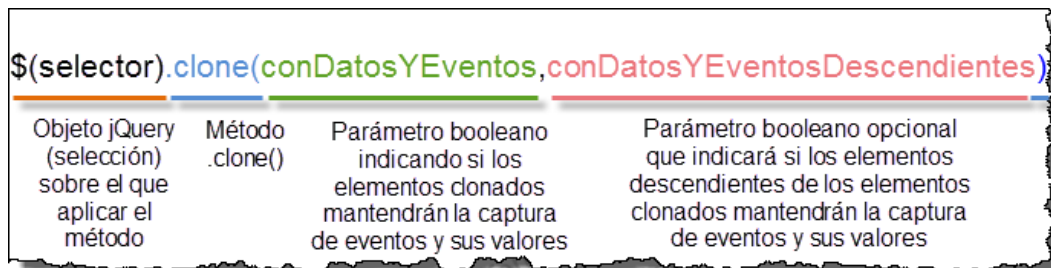
Método `.wrapInner()`

El método `.wrapInner()` nos permitirá "envolver" a todos los hijos de cada uno de los elementos incluidos en la selección sobre la que se apliquen dentro de un nuevo elemento, cuyo HTML se proporcionará como parámetro al método. De este modo, los elementos existentes en la selección sobre la que se aplique pasarán a contener un único hijo, dentro del que se establecerán todos sus hijos anteriores.



Método `.clone()`

El método `.clone()` creará copias exactas de los elementos incluidos en la selección sobre la que se aplique el método, incluyendo en cada una de las copias todos los elementos descendientes de los elementos copiados.



En referencia a los campos de los formularios, el método `.clone()` duplicará los elementos a partir de su HTML original, es decir, los elementos clonados no dispondrán de los valores de los campos modificados: Si con el método `.clone()` obtenemos una copia de un `input[type="text"]`, el atributo "value" contendrá su valor original en lugar del valor actual del elemento.



Al llamar al método `.clone()` los nuevos elementos no son insertados automáticamente en el DOM, sino que están pendientes de ser posicionados con métodos como `.append()` o `.insertBefore()`. Además, los elementos copiados podrán ser modificados antes de su inserción en el DOM

La llamada por defecto al método `.clone()` solamente duplicará los elementos, pero no así la captura de eventos existentes en los mismos. Por ello, el método aceptará opcionalmente dos parámetros con los que poder definir el comportamiento del método frente a la captura de eventos:

El primer parámetro, definirá si se reasignarán los eventos sobre los elementos duplicados, mientras que el segundo definirá el reasignado de la captura de eventos sobre los descendientes.



Debemos tener precaución al utilizar el método `.clone()` sobre elementos con el atributo ID asignado, ya que este atributo debe ser único para todos los elementos de una página. Antes de insertar en el DOM los elementos copiados, elimina de los mismos el atributo ID para evitar este inconveniente.

Eliminación de elementos del DOM

Métodos específicos para la eliminación de elementos de la página

Método `.detach()`

Con el método `.detach()` "desenlazaremos" del DOM los elementos incluidos en la selección sobre la que se aplique. Como resultado, este método devolverá un objeto jQuery conteniendo los elementos desenlazados.

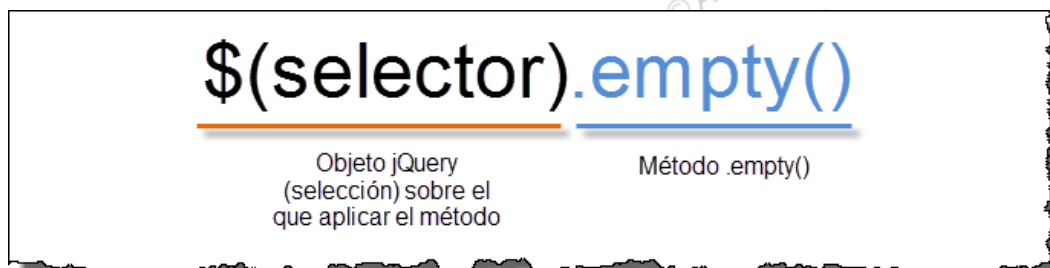
Este método es de gran utilidad para remover elementos y posteriormente insertarlos en otra posición del DOM ya que los eventos de éstos se mantendrán activos.



Opcionalmente, podremos indicarle al método un parámetro con el que filtrar el conjunto de elementos seleccionados a desenlazar.

Método `.empty()`

El método `.empty()` eliminará, de los elementos incluidos en la selección, tanto los nodos de texto como los descendientes de los mismos.



Al llamar el método `.empty()` jQuery eliminará, además de los elementos, todos los eventos vinculados con los mismos.

Método .remove()

El método .remove() tendrá un funcionamiento muy similar al método .empty(), eliminando del DOM, además de los descendientes, los propios elementos sobre los que se aplique el método.

En este caso, el método aceptará un selector, que filtrará el conjunto de elementos de la selección a eliminar.



Método .replaceAll()

Con el método .replaceAll() podremos reemplazar los elementos que cumplan con el selector indicado como parámetro por la selección sobre la que se aplique el método.

Este método es muy útil para posicionar en el DOM elementos creados dinámicamente, e insertarlos en reemplazo de elementos ya existentes.



Método .replaceWith()

El método .replaceWith() funcionará de forma inversa a .replaceAll(), recibiendo como primer parámetro la selección a ser reemplazada, y como segundo, los elementos con los que reemplazar.



`$(selector).replaceWith($(selector))`

Objeto jQuery
(selección) sobre el
que aplicar el método

Método
.replaceWith()

Selección que
contendrá el elemento
a utilizar como
reemplazo

Método .unwrap()

Con el método .unwrap(), los elementos incluidos en la selección sobre la que se apliquen, reemplazarán a sus elementos padre. El funcionamiento del método será el contrario al conseguido con el método .wrap()

`$(selector).unwrap()`


Objeto jQuery
(selección) sobre el
que aplicar el método

Método .unwrap()

Ejercicios

Ejercicio 1: Utilidad de creación de formularios.

Duración estimada del ejercicio


30
minutos


Descarga el archivo comprimido del ejercicio y extráelo en tu directorio de trabajo. Este archivo contiene el documento "ejercicio-manipulacion-dom.htm", que dispone del código HTML necesario, y el código de captura de evento del botón, en el que tendrás que escribir el código para dotar de funcionalidad a la página.

Reemplaza los comentarios del código fuente por el código que cree, a partir de la información establecida en la utilidad, elementos del formulario.

```
$(document).ready(function(){
    $("#btnCrearCampo").on("click", function(e){
        e.preventDefault();
        //Introduce el código necesario para que esta utilidad de creación de campos funcione correctamente.
        //Como resultado, el formulario dispone de un campo.
    });
});
```

Lo necesario para comenzar

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




ejercicio-manipulacion-dom.zip

Archivo comprimido con la plantilla para realizar el ejercicio

Ejercicio 2: Utilidad de creación de formularios mejorada.

Duración estimada del ejercicio


15
minutos

Añade nuevas funcionalidades al ejercicio anterior. Añade el siguiente código HTML al final del div#actionsContainer:

```
<p>Eliminar un campo:</p>
<table border="1" cellpadding="1" cellspacing="1" style="width: 500px;">
  <tbody>
    <tr>
      <th scope="row">Indica el índice del elemento a eliminar<br />(utilizando para su selección el selector :eq(index)</th>
      <td><input name="fieldRemoveIndex" type="text" /></td>
    </tr>
    <tr>
      <th colspan="2" scope="row">
        <input type="button" id="btnDelItem" value="Eliminar elemento" /></th>
      </tr>
    </tbody>
  </table>
```

También deberás añadir el siguiente código dentro de la captura del evento "ready" del document.

```
$("#btnDelItem").bind("click", function(e){
  e.preventDefault();
  /*aquí tu código*/
});
```