Diseño Web — Tecnologías Involucradas

JQuery

info@covetel.com.ve [1]

[1] Cooperativa Venezolana de Tecnologías Libres R.S.

JQuery

JQuery es una biblioteca que contiene una amplia variedad de métodos y funciones de los cuales sera muy útil saber las categorías básicas y como utilizarlas para nuestro beneficio.

Obtención de JQuery

El sitio web oficial de JQuery siempre es el recurso más al día para el código y noticias relacionadas con la biblioteca. Para empezar necesitamos una copia de JQuery, que se puede descargar desde la página principal del sitio. Varias versiones de JQuery pueden estar disponibles en cualquier momento dado, la última versión sin comprimir será la más apropiada para nosotros.

Para usar JQuery no es necesaria su instalación sólo tenemos que colocarlo en nuestro sitio en un lugar público. Cada vez que necesitamos un metodo de JQuery simplemente lo referenciamos desde el archivo del documento HTML.

Configuración del documento HTML

Existen tres componentes en el uso de JQuery:
Componentes

- El Documento HTML.
- El Archivo CSS con los estilos.
- Los Archivos Javascript.

En este ejemplo se muestra como referenciar los componentes de JQuery.

Configuración del documento HTML

```
1  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang:"en" lang:"en">
2   <head>
3    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
4    <title>JQuery</title>
5    <link rel="stylesheet" href="jquery.css" type="text/css" />
6    <script src="jquery.js" type="text/javascript"></script>
7    <script src="appjquery.js" type="text/javascript"></script>
8    </head>
9   <body>
10  </body>
11 </html>
```

Escribiendo el Código JQuery

Nuestro código personalizado lo hemos incluido en el código HTML utilizando `<script src="appjquery.js"type="text/javascript"></script>`. A pesar de lo mucho que realiza tiene pocas lineas de código:

Escribiendo el Código JQuery

```
1  JQuery.fn.toggleNext = function() {
2   this.toggleClass('arrow-down')
3   .next().slideToggle('fast');
4  }
5
6  $(document).ready(function () {
7   $('div id="page-contents"></div>')
8   .prepend('<h3>Page Contents</h3>')
9   .append('<div></div>')
10  .prependTo('body');
```

Escribiendo el Código JQuery

```
11  $('#content h2').each(function(index) {
12  var $chapterTitle = $(this);
13  var chapterId = 'chapter~' + (index + 1);
14  chapterTitle.attr('id', chapterId);
15  $('<a></a>').text($chapterTitle.text())
16   .attr({
17    'title': 'Jump to' + $chapterTitle.text(),
18    'href': '#' + chapterId
19   })
20   .appendTo('#page-contents div');
21   });
```

Escribiendo el Código JQuery

```
22  $('#page-contents h3').click(function() {
23    $(this).toggleNext();
24  });
25
26  $('#introduction > h2 a').click(function() {
27    $('#introduction').load(this.href);
28    return false;
29  });
30 });
```

Este ejemplo crea una tabla dinámica para los usuarios.

Disección del Script

Este script ha sido elegido específicamente ya que ilustra la capacidad general de la biblioteca JQuery e identifica las categorías de los métodos utilizados.

Selector de Expresiones

Antes de que podamos actuar en un Documento HTML se debe localizar las partes pertinentes, en el script algunas veces utilizamos un método simple para buscar un elemento:

```
1 $('#introduction')
```

Esta expresión se crea un nuevo objeto JQuery que hace referencia al elemento de ID `introduction`, algunas veces se requiere un selector más complejo:

```
1 $('#introduction > h2 a')
```

Aqui se obtiene un objeto JQuery que se refiere a los elementos descendientes de <h2> y que son hijos del elemento con ID `introduction`.

Métodos de recorrido DOM

A veces tenemos un objeto JQuery que ya hace referencia a un conjunto de elementos DOM, pero tenemos que realizar una acción en uno diferente, en estos casos los métodos de recorrido DOM son útiles, esto lo podemos ver en parte de nuestro script:

```
1  this.toggleClass('arrow-down')
2   .next()
3   .slideToggle('fast');
```

Debido al contexto de esta pieza de código, la palabra reservada `this` se refiere al objeto JQuery en este caso el objeto apunta la etiqueta `<h3>`. La llamada al método `.toggleClass` manipula este elemento de partida, el método `.next()` cambia el funcionamiento del elemento y `.slideToggle` llama las acciones del elemento `<div>`

Métodos de Eventos

Aun cuando podemos modificar la página a nuestra voluntad necesitamos métodos que reaccionen a eventos causados por los usuarios en el momento adecuado:

```
1  $('#introduction > h2 a').click(function() {
2   $('#introduction').load(this.href);
3   return false;
4  });
```

En este fragmento se registra un controlador que se ejecutará cada vez que se hace click en la etiqueta de anclaje seleccionado.

Métodos de Eventos

```
1  $(document).ready(function() {
2    //..
3  });
```

Este método nos permite registrar comportamiento inmediatamente después de que la estructura del DOM este disponible en nuestro código

Métodos de Efectos

Los métodos de evento nos permiten reaccionar a las entradas del usuario. En los métodos de efecto se hace con efecto, en lugar de ocultar y mostrar inmediatamente elementos lo hace con animación.

```
1  this.toggleClass('arrow-down')
2   .next()
3   .slideToggle('fast');
```

Este método realiza una rápida transición de deslizamiento en el elemento, permitiendo ocultar y mostrar alternativamente con cada invocación.

Métodos AJAX

Muchos sitios web modernos emplean técnicas para carga el contenido cuando no se ah solicitado una actualización de la página. JQuery permite lograr esto con facilidad, los métodos de AJAX inician la solicitud de contenido y nos permite monitorear su progreso.

```
1  $('#introduction > h2 a').click(function() {
2   $('#introduction').load(this.href);
3   return false;
```

En la línea 2 el método `.load` nos permite obtener otro documento HTML desde el servidor e insertarlo en el documento actual, todo ello con una línea de código.

Elemento: T

Todos los elementos que tienen un nombre de etiqueta para T. **Ejemplo**

- $('div') *: selecciona todos los elementos con un nombre de etiqueta* `div` *en el documento.*
- $('em') *: selecciona todos los elementos con un nombre de etiqueta* `em` *en el documento.*

JQuery utiliza JavaScript `getElementByTagName()` para los selectores de nombre de etiqueta.

ID: #myid

El único elemento con un ID igual a `myid`.

Ejemplo

- $('#myid') *: selecciona el unico elemento con* `id = 'myid'`*, independientemente de su nombre de etiqueta.*
- $('p#myid') *: selecciona un solo párrafo con* `id` *de* `'myid'`*, el único elemento* `<p id = 'myid'>`*.*

Cada nombre asignado a un `id` debe ser utilizado sólo una vez en un documento.

Para un selector `id` JQuery utiliza el `getElementById()`.

Clase: .myclass

Todos los elementos que tienen una class `myclass`. **Ejemplo**

- $('.myclass') *: selecciona todos los elementos con una clase `myclass`.*
- $('p.myclass') *: selecciona todos los párrafos con una clase `myclass`.*
- $('.myclass.otherclass') *: selecciona todos los elementos con una clase `myclass` y `otherclass`*

En términos de velocidad, el ejemplo 2 es preferible al 1, ya que el primero utiliza el nativo `getElementsByTagName()` función de JavaScript para filtrar la búsqueda, y las miradas de la clase dentro del subconjunto emparejado de elementos DOM.

Descendiente: E F

Todos los elementos encontrados por F que son descendientes de uno de los elementos coincidentes por E.

Ejemplo

- $('#container p') *: selecciona todos los elementos encontrados por <p> que son descendientes de un elemento que tiene como id container.*
- $('a img') *: selecciona todos los elementos encontrados por <img> que son descendientes de un elemento coincidente con <a>.*

Un descendiente de un elemento puede ser un hijo, nieto, bisnieto y así sucesivamente. Por ejemplo en el siguiente código HTML, el elemento `<img>` es descendiente de los elementos `<span>`, `<p>`, `<div id="inner">` y `<div id="container">`:

```
1  <div id="container">
2    <div id="inner">
3      <p>
4        <span><img src="example.jpg" alt="" /></span>
5      </p>
6    </div>
7  </div>
```

Hijo: E ¿ F

Todos los elementos encontrados por F que son hijos de un elemento acompañado por E. **Ejemplo**

- $('li¿ul') *: selecciona todos los elementos que comienzan por <ul> que son hijo de un elemento acompañado por *
- $('p¿code') *: selecciona todos los elementos encontrados por <code> que son hijos de un elemento acompañado por <p>*

El combinador de hijo puede ser pensado de una forma mas especifica para obtener los descendientes seleccionando sólo el descendiente de primer nivel, en el siguiente código HTML el elemento `<img> es un hijo único del elemento <span>`

```
1  <div id="container">
2    <div id="inner">
3      <p>
4        <span><img src="example.jpg" alt="" /></span>
5      </p>
6    </div>
7  </div>
```

Hermanos adyacentes: E + F

Todos los elementos encontrados por F seguidos inmediatamente por un Elemento E y que tengan un mismo padre. **Ejemplo**

- $('ul + p') *: selecciona todos los elementos por <p> que están inmediatamente después de un elemento <ul>.*
- $('strong + em') *: selecciona todos los elementos encontrados por que están después de un elemento .*

Un punto importante a considerar tanto con el combinar + como con el es que solo seleccionan los hermanos.

```
1  <div id="container">
2    <ul>
3      <li></li>
4      <li></li>
5    </ul>
6      <p>
7        <img />
8      </p>
9  </div>
```

Hermanos adyacentes: E + F

Del ejemplo anterior:

- $('ul + p') *: selecciona <p> ya que esta inmediatamente después de <ul> y los dos elementos comparten el mismo padre .*
- $('ul + img') *: no selecciona nada porque <ul> es un nivel mas alto que <img> en el árbol DOM.*
- $('li + img') *: no selecciona nada porque <li> y <img> estan en el mismo nivel en el árbol DOM.*

Hermanos en General: E F

Todos los elementos encontrados por F que están después de un elemento E y tienen el mismo padre. **Ejemplo**

- $('p ul') *: selecciona todos los elementos encontrados por <ul> que están después de un elemento <p>.*
- $('code code') *: selecciona todos los elementos encontrados por <code> después de <code>.*

La diferencia ente el combinador + y el es el alcance. mientras que el combinador + alcanza los elementos inmediatamente siguientes el combinador se extiende a todos los elementos de un mismo nivel.

```
1   <ul>
2     <li class="first"></li>
3     <li class="second"></li>
4     <li class ="third"></li>
5   </ul>
6   <ul>
7     <li class="fourth"></li>
8     <li class="fifth"></li>
9     <li class="sixth"></li>
10  </ul>
```

Hermanos adyacentes: E + F

Del ejemplo anterior:

- $('li.first li') *: selecciona <li class="second"> and <li class="third">.*
- $('li.first + li') *: selecciona <li class="second">.*

Multiple Elementos: E, F, G

Selecciona todas los elementos encontrados por el selector de expresiones E, F o G.

- $('code, em, strong') *: selecciona todos los elementos encontrados por <code> or or .*
- $('p strong, .myclass') *: selecciona todos los elementos por que son descendientes de un elemento encontrado <p> y tienen una clase .myclass.*

El combinador coma `(,)` es eficiente para seleccionar elementos dispares.

Enésimo Hijo (:nth-child(n))

Todos los elementos que son enésimos hijos de sus padres. **Ejemplo**

- $('li:nth-child(2)') *: selecciona todos los elementos encontrados por <li> que son segundo hijo del padre.*
- $('p:nth-child(5)') *: selecciona todos los elementos encontrados por <p> que son quinto hijo de su padre.*

```
1  <div>
2    <h2></h2>
3    <p></p>
4    <h2></h2>
5    <p></p>
6    <p></p>
7  </div>
```

- $('p:nth(1)') *: selecciona el segundo <p> porque la numeración para :nth(n) comienza en 0.*
- $('p:nth-child(1)') *: no selecciona nada porque el primer elemento hijo del padre no es <p>.*
- $('p:nth(2)') *: selecciona el primer <p> porque es el segundo hijo del padre.*

Hijos

- Primer Hijo (:first-child) *: Todos los elementos que son primer hijo de su padres.*
  Ejemplo
  - $('li:first-child') *: selecciona todos los elementos encontrados por <li> que son el primer hijo del padre.*
- Ultimo Hijo (:last-child) *: Todos los elementos que son el último hijo del padre.*
  Ejemplo
  - $('li:first-last') *: selecciona todos los elementos encontrados por <li> que son el último hijo del padre.*
- Único Hijo (:only-child) *: Todos los elementos que son único hijo del padre.*
  Ejemplo
  - $('li:first-last') *: selecciona todos los elementos encontrados por <li> que son único hijo del padre.*

Diversos Selectores

- No (:not(s)) *: Todos los elementos que no coinciden con el selector* `s`.
  Ejemplo
  - $('li:not(.myclass)') *: Selecciona todos los elemntos encontrados por* `<li>` *que no tienen* `class="myclass"`.
- Vacio (:empty) *: Todos los elementos que no tienen hijos (incluyendo los nodos de texto).*
  Ejemplo
  - $('p:empty') *: Selecciona todos los elementos encontrados por* `<p>` *que no tienen hijos.*
- Universal (:*)*: Todos los elementos.*
  Ejemplo
  - $('*') *: Selecciona todos los elementos en el documento.*

Descendientes: E//F

Todos los elementos encontrados por F que son descendientes de un elemento encontrado por E. **Ejemplo**

- $('div//code'): *: selecciona todos los elementos encontrados do <code> que son descendientes de un elemento encontrado por *

Este selector de descendientes de XPath funciona igual que el selector de descendiente de CSS, salvo que en la version XPath se puede especificar que inicie desde la raíz del documento.

Hijos: E/F

Todos los elementos encontrados por F que son hijos de un elemento conincidente con E. **Ejemplo**

- $('/docroot/el') *: Selecciona todos los elementos encontrados por <el> que son hijos de un elemento coincidente con <docroot>.*

El selector de XPath hijo es una alternativa para el selector CSS hijo. Si el selector de expresión comienza con un slash como es el caso del ejemplo, el selector inmediatamente después del slash debe estar en la raíz del documento.

Padres: E/..

Todos los elementos que son padres de un elemento que coincide con E.

Ejemplo

- $('.myclass/..') *: Selecciona los elementos padre de todos los elementos que tengan una clase* `myclass`.
- $('.myclass/../p') *: Selecciona los elementos que coinciden con <p> que son hijos de un elemento que tiene una clase* `myclass`.

```
1  <div>
2    <p id="firstp"></p>
3    <div id="subdiv"></div>
4    <p id="secondp">
5      <span class="myclass"></span>
6    </p>
7  </div>
8  <div>
9    <p></p>
10 </div>
```

Padres: E/..

En el código anterior:

- $('span.myclass/..') *: selecciona <p id="secondp"> porque este es el padre de .*
- $('#firstp/../') *: selecciona <p id="firstp">, y <p id="secondp"> porque el selector:*
  - a *: Comienza con <p id="firstp">.*
  - b *: Atraviesa un nivel en el árbol DOM.*
  - c *: Selecciona todos los hijos de .*

Contiene: [F]

Todos los elementos que coinciden con F. **Ejemplo**

- $('p[.myclass]') *: selecciona todos los elementos encontrados por <p> que contienen un elemento con una clase* `myclass`.

Valores de Atributos

- Tiene Atributos: [@foo] *Todos los elementos que tienen el atributo foo.*
  Ejemplo
  - $('p[@class]') *: selecciona todos los elementos encontrados por <p> que tiene una class como atributo.*
- Valores de Atributos no iguales: [@foo!=bar] *Todos los elementos en los que el atributo foo no tiene exactamente el valor bar.*
  Ejemplo
  - $('input[@name!=myname]') *: selecciona todos los elementos coincidentes con <input> en los que el atributo name no tiene como valor myname.*
- Comienzo del valor de los Atributos: [@foo^=bar] *Todos los elementos que tengan el atributo foo y su valor comience con el string bar.*
  Ejemplo
  - $('input[@name^=my]') *: selecciona todos los elementos coincidentes con <input> cuyo valor del atributo name comience con my.*

Valores de Atributos

- Final del valor de los Atributos: [@foo$=bar] *Todos los elementos que tengan el atributo foo y su valor termina con el string bar.*
  Ejemplo
  - $('a[@href$=index.html]') *: selecciona todos los elementos coincidentes con <a> en los que el valor del atributo* `href` *termine con* `index.html`.
- Atributos en los que el valor contiene: [@foo*=bar] *Todos los elementos en los que el atributo foo contiene en su valor el string bar.*
  Ejemplo
  - $('a[href*=example.com]') *: selecciona todos los elementos coincidentes con <a> en los que el valor del atributo* `href` *contiene el string* `example.com`.

Selectores de Formulario

Los siguientes selectores se pueden utilizar para acceder a los elementos del formulario en cualquier estado.

- Todos los elementos del formulario (`<input>`, `<select>`, `<textarea>`, `<button>`).
- Todos los campos de texto (`<input type="text">`).
- Todos los campos de password (`<input type="password">`).
- Todos los input submit y elementos button (`<input type="submit">`, `<button>`).
- Todos los input de imagen (`<input type=ïmage">`).
- Todos los elementos de interfaz de usuario que estén habilitadas.
- Todos los elementos de interfaz de usuario checkboxes and radio buttons.

Elemento Incluido, Elemento Extraño, Enésimo Elemento

- Elemento Incluido (:even) Elemento Extraño (:odd)
  Todos los elementos con indice incluido: :even Todos los elementos con un indice extraño: :odd.
  Ejemplo
  - $('li:even') *: selecciona todos los elementos que coinciden con* `li` *que tienen un valor indice par.*
  - $('li:even') *: selecciona todos los elementos que coinciden con* `<tr>` *que tienen un valor indice impar*
- Enésimo Elemento (:eq(n), :nth(n))
  El elemento con un valor de indice igual a n.
  Ejemplo
  - $('li:eq(2)') *: selecciona el tercer elemento* `<li>` .
  - $('p:nth(1)') *: selecciona el segundo elemento* `<p>` .

Mayor que, Menor que, Primero, Ultimo

- Mayor que :gt(n) *Todos los elementos con un indice mayor que N*
  Ejemplo
  - $('li:gt(1)') *: selecciona todos los elementos encontrados por <li> a parir del segundo.*
- Menor que :lt(n) *Todos los elementos con un indice menor que N*
  Ejemplo
  - $('li:lt(3)') *: selecciona todos los elementos encontrados por <li> anteriores del tercero, en otras palabras los dos primeros.*
- Primero :first *La primera instancia de un elemento*
  Ejemplo
  - $('a:first') *: selecciona el primer elemento <a>.*
- Ultimo :last *La ultima instancia de un elemento*
  Ejemplo
  - $('#container .myclass:last') *: selecciona el último elemento con una clase myclass y es descendiente de un elemento con id=çontainer".*

Padre, Que Contiene, Visible, Oculto

- Padre :parent.

  Todos los elementos que son padres de otro elemento.
  Ejemplo
  - $('td:parent') *: selecciona todos los elementos encontrados por* `<td>` *que son padres de otro elemento.*
- Que Contiene :contains(text).

  Todos los elementos que contienen el texto especificado.
  Ejemplo
  - $('li:contains(second)') *: selecciona todos los elementos encontrados por* `<li>` *que son contienen el texto second.*

Padre, Que Contiene, Visible, Oculto

- Visible :visible.

  Todos los elementos que son Visibles.
  Ejemplo
  - $('td:visible') *: selecciona todos los elementos encontrados por <td> que son visibles.*

- Oculto :hidden.

  Todos los elementos que son ocultos.
  Ejemplo
  - $('input:hidden') *: selecciona todos los elementos encontrados por <input> que están ocultos.*

$()

Crea un nuevo objeto JQuery encontrando elementos en el DOM.

```
1  $(selector[, context])
2  $(element)
3  $(elementArray)
4  $(Object)
5  $(html)
```

Ejemplo

- selector *: una cadena que contiene una expresión de selección.*
- context *: parte del árbol DOM dentro de la cual se busca.*
- element *: un elemento DOM para envolver en un objeto JQuery.*
- elementArray *: un array contiene un conjunto de elementos DOM para envolver en un objeto JQuery.*
- object *: un objeto JQuery existente para clonar.*
- html *: una cadena que contiene un código HTMK que describen el nuevo elemento DOM a crear.*

.filter()

Reduce el conjunto de elementos encontrados con aquellos que coincidan con el selector. **Ejemplo**

- $('.filter(selector)') *selector= una cadena que contiene una expresión de selección que coincida con los elementos.*
- $('.filter(function)') *function= una función usada como prueba para cada elemento del conjunto.*

```
1  <ul>
2    <li><strong> list </strong> item 1 - one strong </li>
3    <li><strong> list </strong> item <strong>2</strong> -
4    two<span>strongs</span></li>
5    <li>list item 3</li>
6    <li>list item 4</li>
7    <li>list item 5</li>
8    <li>list item 6</li>
9  </ul>
```

.filter()

```
1  $('li').filter(function(index) {
2    return \$("strong",  this).length ==1;
3  })
```

El resultado de esta expresión será el primer elemento de la lista, ya que contiene exactamente una etiqueta `<strong>` dentro de la función de filtro. `this` se refiere a cada elemento DOM a su vez. El parámetro pasado a la función nos dice que es el elemento DOM a ser comparado.

.not()

- .not(selector) *selector= una cadena que contiene una expresión para que coincida con los elementos.*
- .not(elements) *elements= uno o mas elementos DOM para eliminar del conjunto.*

```
1  <ul>
2    <li>list item 1</li>
3    <li>list item 2</li>
4    <li id="notli">list item 3</li>
5    <li>list item 4</li>
6    <li>list item 5</li>
7  </ul>
```

```
1  $('li').not(document.getElementById('notli'))
```

Esta expresión obtiene un objeto JQuery con los items 1, 2, 4 y 5.

.find()

- .find(selector) *selector= un string que contiene un selector que coincida con los elementos.*

```
<ul class="level-1">
  <li class="item-i">I</li>
  <li class="item-ii">II
    <ul class="level-2">
      <li class="item-a">A</li>
      <li class="item-b">B
        <ul class="level-3">
          <li class="item-1">1</li>
          <li class="item-2">2</li>
          <li class="item-3">3</li>
        </ul>
      </li>
      <li class="item-c">C</li>
    </ul>
  </li>
  <li class="item-iii">III</li>
</ul>
```

```
$('li.item-ii').find('li')
```

El resultado de esta llamada es un objeto JQuery que contiene los items A, B, 1, 2, 3 and C.

.children()

- .children([selector]) *selector=un string que contiene un selector que coincida con los elementos*

```
<ul class="level-1">
  <li class="item-i">I</li>
  <li class="item-ii">II
    <ul class="level-2">
      <li class="item-a">A</li>
      <li class="item-b">B
        <ul class="level-3">
          <li class="item-1">1</li>
          <li class="item-2">2</li>
          <li class="item-3">3</li>
        </ul>
      </li>
      <li class="item-c">C</li>
    </ul>
  </li>
  <li class="item-iii">III</li>
</ul>
```

```
$('ul.level-2').children()
```

El resultado de esta llamada es un objeto JQuery que contiene los items A, B y C.

.parents()

- .parents([selector]) *selector=un string que contiene un selector que coincida con los elementos*

```
<ul class="level-1">
  <li class="item-i">I</li>
  <li class="item-ii">II
    <ul class="level-2">
      <li class="item-a">A</li>
      <li class="item-b">B
        <ul class="level-3">
          <li class="item-1">1</li>
          <li class="item-2">2</li>
          <li class="item-3">3</li>
        </ul>
      </li>
      <li class="item-c">C</li>
    </ul>
  </li>
  <li class="item-iii">III</li>
</ul>
```

```
$('li.item-a').parents()
```

El resultado de esta llamada es un objeto JQuery que contiene la lista level-2, item ii y la lista level-1.

.prev()

- .prev([selector]) *selector=un string que contiene un selector que coincida con los elementos*

```
1  <ul>
2    <li>list item 1</li>
3    <li>list item 2</li>
4    <li class="third-item">list item 3</li>
5    <li>list item 4</li>
6    <li>list item 5</li>
7  </ul>
```

```
1  $('li.third-item').prev()
```

El resultado de esta llamada es un objeto JQuery que contiene item 2.

.add()

- .add(selector) *selector=un string que contiene un selector que coincida con los elementos adicionales*
- .add(elements) *elements=uno o mas elementos para agregar al conjunto de elementos encontrados.*
- .add(html) *html=un fragmento HTML para agregar al conjunto de elementos encontrados.*

```
1  <ul>
2    <li>list item 1</li>
3    <li>list item 2</li>
4    <li>list item 3</li>
5  </ul>
6  <p>un parrafo</p>
```

```
1  $('li').add('p')
```

El resultado de esta llamada es un objeto JQuery que contiene los 4 elementos.

.is()

- .is(selector) *selector=un string que contiene un selector que coincida con los elementos adicionales*

```
<ul>
  <li>list <strong> item 1 </strong></li>
  <li><span>list item 2</span></li>
  <li>list item 3</li>
</ul>
```

```
$('ul').click(function(event) {
  if ($(event.target).is('li') ) {
    $(even.target).remove();
  }
});
```

Ahora cuando el usuario presione click en la palabra list en el primer item o donde sea en el tercer item, el elemento que hace click en la lista se eliminara del documento.

.attr(attribute)

- .attr(attribute) *attribute= el nombre del atributo a obtener.*
- .attr(attribute, value) *value= un valor establecido para el atributo.*
- .attr(map) *map= un mapa de pares atributo-valor para establecer.*
- .attr(attribute, function) *function= una función que devuelve el valor a establecer.*

```
1  <img id="greatphoto" src="brush-seller.jpg" alt="brush seller" />
```

```
1  $('#greatphoto').attr('alt', 'Beijing Brush Seller');
```

Cambia el atributo alt señalando el atributo alt y estableciendo el nuevo valor después de la coma dentro del paréntesis.

.attr()

Mediante el uso de una función para establecer los atributos, podemos concatenar un nuevo valor con un valor existente.

```
1  $('#greatphoto').attr({alt: function() {return 'Beijing' + this.alt}, title:
2  function() {return 'Beijing' + this.alt + ' - photo by Kelly Clarck'}});
```

El uso de una función puede ser aún mas útil cuando se aplican los atributos de elementos multiples.

.removeAttr()

- .removeAttr(attribute) *attribute= un atributo.*

El método .removeAttr utiliza la función JavaScript removeAttribute, pero tiene la ventaja de poder ser encadenado a una expresión de selección JQuery.

.css()

- .css(property, value) *property= un nombre de propiedad CSS. value= un valor establecido para la propiedad.*
- .css(map) *map= un mapa de pares propiedad-valor para establecer.*
- .css(property, function) *function= una función que devuelve el valor a establecer.*

```
1  $('div.example').css('width', function(index) {
2    return index * 50;
3  });
```

Este ejemplo establece la anchura de los elementos que coinciden con lo valores progresivamente más grandes.

Dimensiones

- .height(value) *value= un entero representando el número de pixeles o un entero junto con una unidad opcional de medida.*
- .width(value) *value= un entero representado el número de o un entero junto con una unidad opcional de medida.*

Atributos de Clase

- .addClass(class) *class= uno o mas nombres de clase que se agrega al atributo de clase o a cada elemento emparejado.*
- .removeClass(class) *class= uno o mas nombres de clase que se eliminan del atributo de clase o de cada elemento emparejado.*

```
1 $('p').removeClass('myclass noclass').addClass('yourclass')
```

Aquí las clases `myclass` y `noclass` se eliminan de los párrafos, mientras `yourclass` es añadida.

- .toggleClass(class) *function= un nombre de clase para ser activado en el atributo de clase de cada elemento en el conjunto combinado.*

```
1 <div class="tumble">Some text</div>
```

```
1 $('div.tumble').toggleClass('bounce')
```

```
1 <div class="tumble bounce">Some text</div>
```

Reemplazo DOM

- .html(HTML) *HTML= una cadena HTML para establecer el contenido de cada elemento encontrado.*

```
1    <div class="demo-container">
2      <div class"demo-box">Demostration Box</div>
3    </div>
```

```
1    $('div.demo-container').html('<p>All new content. <em>You bet!</em></p>
         '));
```

```
1    <div class="demo-container">
2      <p>All new content. <em>You Bet!</em></p>
3      <div class"demo-box">Demostration Box</div>
4    </div>
```

Métodos

- .text(text) *text= una cadena de texto para establecer el contenido de cada elemento encontrado.*

```
<div class="demo-container">
  <div class"demo-box">Demostration Box</div>
  <ul>
    <li>list item1</li>
    <li>list <strong>item</strong>2</li>
  </ul>
</div>
```

```
$('div.demo-container').text('<p>This is a Test.</p>')
```

```
<div class="demo-container">
 <p>This is a test</p>
</div>
```

- .val(value) *value= una cadena de texto para establecer el valor de la propiedad de cada elemento encontrado.*

Inserciones DOM, Dentro

- .prepend(content) *content= un elemento, cadena HTML u objeto jQuery para insertar al principio del elemento establecido encontrado.*

```
<div class="demo-container">
  <div class"demo-box">Demostration Box</div>
</div>
```

```
$('div.demo-box').prepend('<div class="insertion">This text
was<strong>inserted</strong></div>');
```

- .prependTo(target) *target= un selector, elemento, cadena HTML, cadena u objeto JQuery; el conjunto combinado de elementos se insertan al principio del elemento especificado por este parámetro.*

```
<div class="demo-container">
  <div class"demo-box">Demostration Box</div>
</div>
```

```
$('<div class="insertion">This text was<strong>inserted</strong></div>'
    ).prependTo('div.demo-box');
```

Inserciones DOM, Dentro

- .append(content) *content= un elemento, cadena HTML u objeto jQuery para insertar al final del elemento establecido encontrado.*

```
<div class="demo-container">
  <div class"demo-box">Demostration Box</div>
</div>
```

```
$('div.demo-box').append('<div class="insertion">This text
was<strong>inserted</strong></div>');
```

- .appendTo(target) *target= un selector, elemento, cadena HTML, cadena u objeto JQuery; el conjunto combinado de elementos se insertan al final del elemento especificado por este parámetro.*

```
<div class="demo-container">
  <div class"demo-box">Demostration Box</div>
</div>
```

```
$('<div class="insertion">This text was<strong>inserted</strong></div>'
    ).appendTo('div.demo-box');
```

Inserciones DOM, Fuera

- .insertBefore(content) *content= un selector o elemento que se insertara antes del conjunto encontrado.*

```
1    <div class="demo-container">
2      <div class"demo-box">Demostration Box</div>
3    </div>
```

```
1    $('<div class="insertion">This text was<strong>inserted</strong></div>'
         ).insertBefore('div.demo-box');
```

- .insertAfter(content) *content= un selector o elemento que se insertara después del conjunto encontrado.*

```
1    <div class="demo-container">
2      <div class"demo-box">Demostration Box</div>
3    </div>
```

```
1    $('<div class="insertion">This text was<strong>inserted</strong></div>'
         ).insertAfter('div.demo-box');
```

DOM Copia, DOM Eliminación

- .clone([deep]) *deep= Un booleano. Predeterminado verdadero. Si se establece falso el método copia sólo los elementos coincidentes con si mismo con exclusión de cualquier hijo o descendiente.*

```
1    <div class="demo-container">
2      <div class"demo-box">Demostration Box</div>
3    </div>
```

```
1    $('div.demo-box:last').clone().insertAfter('div.demo-box:last');
```

- .remove([selector]) *selector= Un selector que filtra los elementos encontrado a ser removidos.*

```
1    <div class="demo-container">
2      <div class"demo-box">Demostration Box</div>
3    </div>
```

```
1    $('div.demo-box').remove('#temporary-demo-box')
```

Controlador adjunto de eventos

- .bind(evenType[, eventData, handler]) *evenType= Una cadena que contiene un tipo de evento JavaScript, sea click o submit.*
  eventData= Un mapa de datos que pueden ser pasados por un manejador de eventos.
  handler= Una función que se ejecutará cada vez que se dispare el evento.

```
1    $('#foo').bind('click', function() {
2      alert('User clicked on foo');
3    });
```

- .trigger(evenType [, extraParameters]) *extraParameters= Un array de parámetros adicionales pasados a través de un manejador de eventos.*

```
1    $('#foo').bind('click' function() {
2      alert($(this).text());
3    });
4    $('#foo').trigger('click');
```

Carga de Documentos

- .load(handler) *handler= Una función que se ejecutará cada vez que se dispare el evento.*

```
1       <img class="target" src="hat.gif" width="80" height="54" />
```

```
1   $('.target').load(function() {
2     $(this).log('Load event was triggered.');
3   });
```

- .error(handler) *handler= Una función que se ejecutará cada vez que se dispare el evento.*

```
1       <img class="target" src="missing.gif" width="80" height="54" />
```

```
1   $('.target').error(function() {
2     $(this).log('Error event was triggered.');
3   });
```

Eventos del Mouse

- .click(handler) *handler= Una función que se ejecutará cada vez que se dispare el evento.*

```
1    $('.target').click(function() {
2      $(this).log('Click event was triggered.');
3    });
```

- .mouseover(handler) *handler= Una función que se ejecutará cada vez que se dispare el evento.*

```
1    $('.target').mouseover(function() {
2      $(this).log('Mouseover event was triggered.');
3    });
```

- .hover(handlerIn, handlerOut) *handlerIn= Una función que se ejecutará cuando el puntero del mouse entra en el elemento.*
  handlerOut= Una función que se ejecutará cuando el puntero del mouse sale del elemento.

```
1    $('.target').hover(function() {
2      $(this).log('Hover event was triggered entering');
3    }function() {
4      $(this).log('Hover event was triggered leaving');
5    });
```

Eventos del Mouse

- .mousemove().
- .dblclick().
- .mousemove().
- .toggle().
- .mouseup(.)
- .mousedown().

Eventos del Formulario

- .focus(handler) *handler= Una función que se ejecutará cada vez que se dispare el evento.*

```
1    $('.target').focus(function() {
2      $(this).log('Focus event was triggered.');
3    });
```

- .blur().
- .change().
- .select().
- .submit(handler) *handler= Una función que se ejecutará cada vez que se dispare el evento.*

```
1    <form class="target" action="foo.html">
2      <input type="text" /><input type="submit" value="Go" />
3    </form>
4    <div class="trigger button">Button</div>
```

```
1    $('.trigger').click(function() {
2      $('.target').submit();
3    });
```

Eventos del Teclado

- .keypress(handler) *handler= Una función que se ejecutará cada vez que se dispare el evento.*

```
1     $('.target').keypress(function() {
2       $(this).log('Keypress event was triggered.');
3     });
```

- .keydown().
- .keyup().

Eventos del Navegador

- .resize(handler) *handler= Una función que se ejecutará cada vez que se dispare el evento.*

```
1    $(window).resize(function() {
2      $(this).log('Resize event was triggered.');
3    });
```

- .scroll(handler) *handler= Una función que se ejecutará cada vez que se dispare el evento.*

```
1    $('.trigger').click(function() {
2      $('.target').scroll();
3    });
```

Efectos Pre-empaquetados

- .show([speed][, callback]) *speed= una cadena o un número para determinar cuánto tiempo se ejecutara la animación.*
  callback= una función para saber cuando ha finalizado la animación.

```
<div class="content">
  <div class="trigger button">Trigger</div>
  <div class="target"><img src="hat.gif" width="80" height="54"></div>
  <div class="long"></div>
</div>
```

```
$('.trigger').click(function() {
  $('.target').show('slow',  function() {
    $(this).log('Effect Complete.');
  });
});
```

- .hide().
- .toggle().
- .slideDown() .slideUp() .slideToggle()
- .fadeIn() .fadeOut() .fadeTo()

Efectos Personalizados

- .animate(properties [, speed][, easing][, callback]) *properties= un mapa de propiedades CSS por los que se moverá la animación.*
  speed= una cadena o un número para determinar el tiempo que se ejecutara la animación.
  easing= una cadena facilita el uso de la función en la transición.
  callback= una función para saber cuando ha finalizado la animación.

```
1    <div class="content">
2      <div class="trigger button">Trigger</div>
3      <div class="target"><img src="hat.gif" width="80" height="54"></div>
4      <div class="long"></div>
5    </div>
```

```
1    $('.trigger').click(function () {
2      $('.target').animate({
3      'width': 300,
4      'left': 100,
5      'opacity': 0.25
6      }, 'slow', function() {
7        $(this).log('Effect Complete');
8      });
9    });
```

Interfaz de Bajo Nivel

- .ajax(settings) *settings= un mapa que contiene las siguientes opciones para loas solicitudes:*
  -

```
1    $('.target').focus(function() {
2      $(this).log('Focus event was triggered.');
3    });
```

- .ajaxSetup(settings) *settings= .*

```
1    <form class="target" action="foo.html">
2      <input type="text" /><input type="submit" value="Go" />
3    </form>
4    <div class="trigger button">Button</div>
```

```
1    $('.trigger').click(function() {
2      $('.target').submit();
3    });
```