JavaScript

Sesión 6 - **jQuery**

Índice

jQuery

- Creado por *John Resig* - http://ejohn.org

6.1 Características I

- Software libre y gratuito.
- Funciona en todos los navegadores modernos
- Abstrae las prestaciones que ofrece cada navegador
  - Facilita el *Cross-Browser Development*
- Simplifica las tareas de scripts comunes como:
  - Manipular el contenido de un documento web
  - Gestionar los eventos con independencia del navegador
  - Añadir efectos y transiciones.
- Facilita los escenarios de trabajo más comunes como:
  - Gestionar las configuraciones que se realizan al cargar una página
  - Tras un evento, obtener y manipular/animar el contenido para volverlo a mostrar en la página.

Características II

- Sintaxis de selectores similar a CSS
- Trabaja con conjuntos de elementos, permitiéndonos realizar una acción sobre más de un elemento DOM de manera simultánea
- Permite realizar múltiples operaciones sobre un conjunto de elementos mediante una única línea de código, mediante el encadenamiento de sentencias (*statement chaining*).
- Extensible mediante multitud de plugins de terceros, o si queremos, mediante plugins de nuestra creación

6.2 Descarga / Versiones

http://jquery.com/download/

- 1.12.1 o 2.2.1
  - Version 2.x *(2% de uso)* deja de dar soporte a las versiones 6, 7 y 8 de *Internet Explorer.*
- 2 tipos de archivos
  - script no comprimido, permite consultar el código pero ocupa más → 287 KB
  - script comprimido (*minificado*), se usa en producción → 95 KB
- CDNs
  - jQuery: `<script src="//code.jquery.com/jquery-1.12.1.min.js"></script>`
  - Google: `<script src="//ajax.googleapis.com/ajax/libs/jquery/1.12.1/jquery.min.js"></script>`
  - CDNJS: `<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/1.12.1/jquery.min.js"></script>`
- Se ha de cargar antes que las librerías que dependan de ella:

```
<link rel="stylesheet" type="text/css" href="estilos.css" />
<script src="jquery.js" />
<script src="libQueUsaJQuery.js" />
```

6.3 Alias: `jQuery` por `$`

- Objeto `jQuery`

```
document.getElementById("miCapa").className = "resaltado";
jQuery("#miCapa").addClass("resaltado");
```

- `$` es una alias de `jQuery`

```
$("#miCapa").addClass("resaltado");
```

- Si utilizamos otras librerías *JavaScript* que también utilizan el `$`, se recomienda realizar una llamada a `$.noConflict()` para evitar conflictos de *namespace*.
- Tras la llamada, el alias `$` deja de estar disponible, obligándonos a escribir `jQuery` cada vez que escribiríamos `$`

Permitiendo otro alias

- Si queremos permitir que fuera de una *IIFE* se use el alias `$` para usar otra librería *JS* como *Mootools* o *Prototype*, podemos hacer esto:

```
(function($) {
  // código jQuery que usa el $
}) (jQuery);
```

- Otra manera es usando el evento `document.ready`
  - Se lanza cuando la página ha cargado y recibe como argumento el objeto global `jQuery`, el cual podemos renombrar:

```
jQuery( document ).ready(function( $ ) {
  // código jQuery que usa el $
});
```

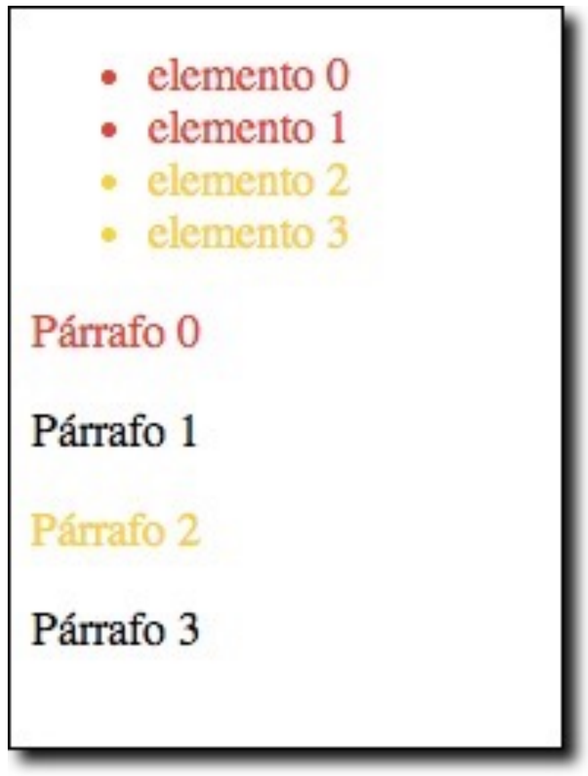
6.4 Seleccionando contenido

- Mediante selectores y filtros
- http://api.jquery.com/category/selectors/

- El resultado de aplicar un selector siempre va a ser un array de objetos
  - No son objetos *DOM*, sino objetos *jQuery* que envuelven a los objetos *DOM* añadiéndoles funcionalidad extra

- Posteriormente, podemos aplicar uno o más filtros sobre un selector para refinar el array de resultados que devuelve.

Página ejemplos




```html
<!DOCTYPE html>
<html>
<head lang="es">
  <meta charset="UTF-8">
  <title>Selectores</title>
  <style>
  .a { color: red; }
  .b { color: gold; }
  </style>
</head>
<body>
  <ul id="listado">
    <li class="a">elemento 0</li>
    <li class="a">elemento 1</li>
    <li class="b">elemento 2</li>
    <li class="b">elemento 3</li>
  </ul>
  <p id="pa0" class="a" lang="es-AR">Párrafo 0</p>
  <p>Párrafo 1</p>
  <p id="pa2" class="b">Párrafo 2</p>
  <p id="pa3" lang="es-ES">Párrafo 3</p>
</body>
</html>
```

6.4.1 Selectores - Básicos

- Sintaxis *CSS* → Similar a `querySelector`
- `$("selector")`
- `$("selector", contexto)` → restringe el selector al contexto

```
$('p:even');
$('p:even', $('.importante'));
```

Selector	Propósito	Ejemplo
etiqueta	Encuentra todos los elementos de *etiqueta*	`p`
#identificador	Elemento cuyo `id` es *identificador*	`#contenido`
.nombreClase	Elementos cuyo `class` es *nombreClase*	`.anuncio`
etiqueta.nombreClase	Elementos de tipo *etiqueta* cuyo `class` es *nombreClase*	`div.anuncio`
etiqueta#ident.nombreClase	Elemento de tipo etiqueta cuyo `id` es *ident* y su `class` es *nombreClase*	`div#banner.anuncio`
*	Todos los elementos de la página	`*`

Ejemplos selectores básicos

- Obtener todos los párrafos

```
document.getElementsByTagName("p");
$("p");
```

- Obtener la etiqueta cuyo id sea `listado`

```
document.getElementById("listado");
$("#listado");
```

- Obtener las etiquetas `li` cuya clase sea `a`:

```
var todos = document.getElementsByTagName("li");
var liClaseA = [];
for (var i = 0, tam = todos.length; i < tam; i++) {
  if (todos[i].className === "a") {
    liClaseA.push(todos[i]);
  }
}
$("li.a");  // jQuery
```

Selectores jerárquicos

Selector	Propósito	Ejemplo
***selector1*, *selector2*, ...**	Encuentra todos los selectores especificados	`p, div`
.clase1.clase2	Elementos cuya `class` son *clase1* **y** *clase2*	`.anuncio.vip`
padre > hijo	Todos los elementos *hijo* que son hijos directos del tipo *padre*	`div > img`
ascendiente descendiente	Todos los elementos *descendiente* contenidos dentro del tipo *ascendiente* (hijos, nietos, etc…)	`div img`
anterior + posterior	Todos los elementos *posterior* que están después de *anterior*	`img + p`
anterior ~ hermanos	Todos los hermanos (*siblings*) que están tras *anterior* y que cumplen el selector de *hermanos*	`div.carrusel ~ img`

Ejemplos selectores jerárquicos

http://jsbin.com/modiko/6/edit?html,css,js,output

- Párrafos y los elementos de lista cuya clase sea `b`:

```
$("p, li.b");
```

- Elementos de lista cuya clase sea `a` y que sean descendientes de una lista desordenada

```
$("ul li.a");
```

- Primer párrafo que está tras una lista desordenada:

```
$("ul + p");  // párrafo 0
```

- Párrafos que son hermanos posteriores del id `listado`

```
$("#listado ~ p"); // párrafo 0
```

6.4.2 Filtros básicos

- Se aplican sobre los selectores para refinar la selección.
- Comienzan con `:`
- Se pueden encadenar → `#noticias tr:has(td):not(:contains("Java"))`

Filtro	Propósito
:first	Selecciona sólo la primera instancia del conjunto devuelto por el selector
:last	Selecciona sólo la última instancia del conjunto devuelto por el selector
:even	Selecciona sólo los elementos pares del conjunto devuelto por el selector
:odd	Selecciona sólo los elementos impares del conjunto devuelto por el selector
:eq(*n*)	Selecciona los elementos situados en el índice *n (0-index)*
:gt(*n*)	Selecciona los elementos situados detrás del índice *n*
:lt(*n*)	Selecciona los elementos situados antes del índice *n*
:header	Selecciona todos los elementos cabeceras (`h1`, `h2`, `h3` ,...)
:animated	Selecciona todos los elementos que están actualmente animados
:not(*selector*)	Selecciona los elementos que no cumplen el *selector*

Ejemplo filtros básicos

http://jsbin.com/hufido/1/edit?html,css,js,output

- Obtener el primer párrafo

```
$("p:first");
```

- Obtener el último elemento cuya clase sea `a`

```
$(".a:last");
```

- Obtener los párrafos pares *(el primer párrafo es el 0, con lo cual es par)*

```
$("p:even");
```

- Obtener todos los párrafos menos los dos primeros

```
$("p:gt(1)");
```

- Obtener todos los párrafos menos el segundo

```
$("p:not(p:eq(1))");
```

Filtros basados en atributos

Filtro	Propósito
[*atrib*]	Incluye los elementos que tienen el atributo *atrib*
[*atrib*=*valor*]	Incluye los elementos que tienen el atributo *atrib* con el valor *valor*
[*atrib*!=*valor*]	Incluye los elementos que tienen el atributo *atrib* y no tiene el valor *valor*
[*atrib*^=*valor*]	Incluye los elementos que tienen el atributo *atrib* y su valor comienza por
[*atrib*$=*valor*]	Incluye los elementos que tienen el atributo *atrib* y su valor termina con *valor*
[*atrib**=*valor*]	Incluye los elementos que tienen el atributo *atrib* y su valor contiene *valor*
[*filtroAtrib1*][*filtroAtrib2*]	Incluye los elementos que cumplen todos los filtros especificados, es decir, *filtroAtrib1* y *filtroAtrib2*

Ejemplo filtros basados en atributos

http://jsbin.com/ziraqe/1/edit?html,css,js,output

- Obtener los párrafos que tienen alguna clase

```
$("p[class]");
```

- Obtener el párrafo cuyo id sea `pa2`

```
$("p[id=pa2]");
```

- Obtener los párrafos cuyo id comience por `pa`

```
$("p[id^=pa]");
```

- Obtener los párrafos cuyo *id* comience por `pa` y que su atributo `lang` contenga `es`

```
$("p[id^=pa][lang*=es]");
```

Filtros basados en el contenido

http://jsbin.com/runojo/1/edit?html,css,js,output

Filtro	Propósito
:contains(*texto*)	Incluye los elementos que contienen la cadena *texto*
:empty	Incluye los elementos vacíos, es decir, sin contenido
:has(*selector*)	Incluye los elementos que contienen al menos uno que cumple el *selector*
:parent	Incluye los elementos que son padres, es decir, que contienen al menos otro elemento, incluido texto

- Obtener los párrafos que contienen el número 3
```
$("p:contains(3)");
```
- Obtener todos los elementos que contienen el número 3
```
$(":contains(3)");
$("p:contains(3),li:contains(3)");
```
- Obtener los párrafos que son padres
```
$("p:parents");
```
- Obtener una lista desordenada donde haya algún elemento que sea de la clase b
```
$("ul:has(li[class=b])");
```

Filtros basados en la visibilidad

- Propiedad `visibility`

Filtro	Propósito
:visible	Incluye los elementos visibles
:hidden	Incluye los elementos ocultos

- Obtener todos los párrafos que están ocultos

```
$("p:hidden");
```

Filtros basados en los hijos

Filtro	Propósito
:nth-child(*índice*)	Incluye los hijos número *índice*, numerados de *1* a *n*.
:nth-child(even)	Incluye los hijos pares
:nth-child(odd)	Incluye los hijos impares
:nth-child(*ecuación*)	Incluye los hijos cuya posición cumple la *ecuación*, por ejemplo, `2n` o `3n+1`
:first-child	Incluye los elementos que son el primer hijo
:last-child	Incluye los elementos que son el último hijo
:only-child	Incluye los elementos que son hijos únicos, es decir, no tienen hermanos

- Obtener el tercer elemento de lista de una lista desordenada

```
$("ul li:nth-child(3)");
```

- Obtener el último elemento de lista de una lista desordenada

```
$("ul li:last-child“);
```

Filtros basados en los elementos de formulario

Selector	Propósito
:input	Encuentra todos los *input*, *select*, *textarea* y elementos *button*
:text	Encuentra todos los elementos de dicho tipo
:password	
:radio	
:checkbox	
:submit	
:reset	
:image	
:button	
:file	

```
$("form :input");
```

```
$("form :text:enabled");
```

```
$("form :checkbox:checked");
```

Filtro	Propósito
:enabled	Incluye los elementos que están habilitados
:disabled	Incluye los elementos que están deshabilitados
:checked	Incluye los elementos que están marcados (*radio* y *checkbox*)
:selected	Incluye los elementos que están seleccionados (*select*)

6.4.3 Objeto `jQuery` I

- Tras seleccionar la información mediante selectores y filtros obtendremos un objeto `jQuery`
- Envuelve a uno o más elementos HTML.
- Permite acceder al resultado, iterar sobre él o encadenar nuevas sentencias.

Propiedad / Método	Propósito	Devuelve
context	Devuelve el conjunto de elementos utilizado como contexto	`HTMLElement`
each(*función*)	Invoca la *función* para cada uno de los elementos buscados	`jQuery`
find(*selector*)	Obtiene elementos descendientes que cumplen el *selector*	`jQuery`
get() / toArray()	Devuelve un array con todos los elementos DOM del conjunto de resultado. Se usa cuando necesitamos trabajar con los objetos DOM en vez de los objetos propios de jQuery	`HTMLElement[]`
get(*índice*)	Obtiene un único elemento DOM que ocupa la posición *índice* del conjunto de resultados	`HTMLElement`

Ejemplo Objeto `jQuery` I

- Obtener el primer elemento de la lista

```
$("li").get()[0];  // object HTMLLIElement -> objeto DOM
$("li").get(0); // object HTMLLIElement -> objeto DOM
$("li:first-child"); // object Object -> objeto jQuery
```

- Obtener los elementos de lista que tienen la clase `b`

```
$("ul").find("li.b");
$("ul li.b");
```

- Realizar una acción sobre todos los párrafos

```
$("p").each(function() {
  // Por ejemplo, ponerles un borde
  $(this).css("border","3px solid red");
});
```

Objeto `jQuery` II

Propiedad / Método	Propósito	Devuelve
index(*HTMLElement*)	Obtiene el *indice* del *HTMLElement* especificado	`number`
index(*jQuery*)	Obtiene el *indice* del primer elemento del objeto *jQuery*	`number`
index(*selector*)	Obtiene el *indice* del primer elemento del objeto *jQuery* dentro del conjunto de elementos encontrados por el *selector*. Realiza la operación inversa a las anteriores	`number`
length / size()	Número de elementos del conjunto de resultados	`number`
selector	Devuelve el selector empleado	`string`

Ejemplos objeto `jQuery` II

- Obtener el selector empleado

```
var parrafosImpares = $("p:odd");
console.log(parrafosImpares.selector); // "p:odd"
```

- Obtener la cantidad de párrafos del documento

```
var numParrafos = $("p").length;
```

- Obtener la posición del elemento cuyo id es *contenido*

```
$("body *").index(document.getElementById("contenido"));
$("body *").index($("#contenido"));
$("#contenido").index("body *");
```

De `HTMLElement` a `jQuery`

- Para crear un objeto `jQuery` desde un objeto DOM, podemos hacerlo pasándolo como argumento a la función `$`.
- Permite incluir código que no usa *jQuery* e integrarlo de un modo sencillo.

```
var contenido = document.getElementById("contenido");
var contenidoJQuery = $(contenido);
```

6.4.4 Modificando la selección - Encadenando sentencias

- Tras seleccionar la información mediante selectores y filtros, podemos modificarla añadiendo nuevos elementos al conjunto de resultados o restringiendo los resultados a un subconjunto.
- *jQuery* facilita **encadenar sentencias** de manera conjunta para realizar varias operaciones en una única línea de código.
- Tras usar el selector, mediante el operador `.` podemos añadir métodos que se irán encadenando unos detrás de otros:

```
$(selector).funcion1().funcion2().funcion3();
```

- Al encadenar las sentencias, los métodos se ejecutan sobre el elemento modificado.
- Para detener el *chaining* y volver a trabajar con la selección inicial → `end()`

```
$(this).siblings('button').remoteAttr('disabled').end().attr('disabled','disabled');
```

Ampliando la selección

- Método `add`

Método	Propósito
add(*selector*)	Añade todos los elementos que cumplen el *selector*
add(*selector*, *contexto*)	Añade todos los elementos que cumplen el *selector* dentro del *contexto*
add(*HTMLElement*)	Añade un elemento o un array de *HTMLElements*
add(*HTMLElement[]*)	
add(*jQuery*)	Añade los contenidos del objeto *jQuery*

```
var imgFlickr = $("img[src*=flickr]");
$("img:even").add("img[src*=twitpic]").add(imgFlickr);
```

Reduciendo la selección

Método	Propósito
eq(*índice*)	Elimina todos los elementos a excepción al que ocupa la posición *índice*
first()	Elimina todos los elementos excepto el primero
has(*selector*)	Elimina los elementos que no tengan un descendiente que cumpla el *selector* o el objeto *jQuery* o aquellos descendientes que no incluyan los objetos *HTMLElement* especificados
has(*jQuery*)	
has(*HTMLElement*)	
has(*HTMLElement[]*)	
last()	Elimina todos los elementos excepto el último
slice(*inicio*,*fin*)	Elimina los elementos fuera del rango indicado por los valores de *inicio* y *fin*

```
$("li").first();
$("li").last();
$("li").eq(1); // segundo elemento
$("li").eq(-1); // penúltimo elemento
$("li").slice(0,2) // primeros dos elementos
```

Reduciendo la selección - `filter()` y `not()`

Método	Propósito
filter(*selector*)	Elimina todos los elementos que no cumplen el selector
filter(*HTMLElement*)	Elimina todos los elementos menos el *HTMLElement*
filter(*jQuery*)	Elimina todos los elementos que no están contenidos en el objeto *jQuery*
filter(*función(índice)*)	La función se invoca por cada elemento; aquellos en los que la función devuelva `false` se eliminarán
not(*selector*)	Elimina todos los elementos que no cumplen el selector
not(*HTMLElement[]*)	Elimina todos los elementos menos el *HTMLElement*
not(*HTMLElement*)	Elimina todos los elementos menos el *HTMLElement*
not(*jQuery*)	Elimina todos los elementos que no están contenidos en el objeto *jQuery*
not(*función(índice)*)	La función se invoca por cada elemento; aquellos en los que la función devuelva `false` se eliminarán

Ejemplos `filter()` y `not()`

- Obtener los párrafos que tiene clase o que no la tienen

```
$("p").filter("[class]")
$("p").not("[class]");
```

- Obtener los párrafos cuyo id comience por `pa` o los que no comiencen por `pa`

```
var pa = $("[id^=pa]");
$("p").filter(pa);
$("p").not(pa);
```

- Obtener los párrafos cuya clase es `a` o el párrafo que ocupa la tercera posición

```
$("p").filter(function(indice) {
  return this.getAttribute("class") == "a" || indice == 2;
});
```

6.5 Navegando por la selección

- Una vez seleccionado los elementos, podemos navegar por la selección (*traversing*).
- Mediante relaciones DOM

- **Navegación Descendente**

Método	Propósito
children()	Selecciona los hijos (descendientes inmediatos) de todos los elementos del conjunto de resultados
children(*selector*)	Selecciona todos los elementos que cumplen el selector y que son hijos del conjunto de resultados
contents()	Devuelve los hijos, incluyendo los contenidos de texto u nodos de comentarios de todos los elementos

- Uso conjunto con `find()` para filtrar los resultados.

Navegación ascendente I

Método	Propósito
parent()	Selecciona el padre de cada elemento del objeto *jQuery*, pudiéndose filtrar por el *selector*
parent(*selector*)	
parents()	Selecciona los ascendentes de cada elemento del objeto *jQuery*, pudiéndose filtrar por el *selector*
parents(*selector*)	
parentsUntil(*selector1*)	Selecciona los ascendentes de cada elemento del objeto *jQuery* hasta que se encuentre una ocurrencia para el *selector1*. Los resultados se pueden filtrar mediante *selector2*.
parentsUntil(*selector1*, *selector2*)	
parentsUntil(*HTMLElement*)	Selecciona los ascendentes de cada elemento del objeto *jQuery* hasta que se encuentre uno de los elementos especificados. Los resultados se pueden filtrar mediante un *selector*.
parentsUntil(*HTMLElement*, *selector*)	
parentsUntil(*HTMLElement[]*)	
parentsUntil(*HTMLElement[]*, *selector*)	

Ejemplos navegación ascendente I

- Obtener el padre de todos los enlaces
- Obtener aquellos padres de enlaces que tienen el atributo *lang*

```
$("a").parent();
$("a").parent("[lang]");
```

- Obtener los ascendentes (padre, abuelo, etc..) de los elementos cuya clase sea *a*

```
$(".a").parents();
```

- Obtener los ascendentes hasta llegar a la etiqueta *form* de los elementos cuya clase sea *a* y que contenga el valor *es* en el atributo *lang*

```
$(".a").parentsUntil('form','[lang*=es]');
```

Navegación ascendente II

Método	Propósito
closest(*selector*)	Selecciona el ascendente más cercano de cada elemento del objeto *jQuery* y realiza la intersección con el *selector* / *contexto*.
closest(*selector*,*contexto*)	
closest(*jQuery*)	Selecciona el ascendente más cercano de cada elemento del objeto *jQuery* y realiza la intersección con los elementos contenidos en el parámetro.
closest(*HTMLElement*)	
offsetParent()	Encuentra el ascendente posicionado más cercano (tiene valor `fixed`, `absolute` o `relative` en la propiedad `position`). Útil para trabajar con animaciones

- Obtener los ascendentes más cercanos cuya clase sea a, de la imagen cuya fuente referencie a *flickr*

```
$("img[src*=flickr]").closest(".a");
```

Navegación horizontal

Método	Propósito
siblings()	Selecciona todos los hermanos para cada uno de los elementos del objeto `jQuery`.
next()	Selecciona el hermano inmediatamente posterior
nextAll()	Selecciona todos los hermanos posteriores
nextUntil(*selector*)	Selecciona los hermanos anteriores para cada elemento hasta (sin incluir) un elemento que cumpla el *selector* o un elemento del objeto `jQuery` o del array `HTMLElement`.
nextUntil(*jQuery*)	
nextUntil(*HTMLElement[]*)	
prev()	Selecciona el hermano inmediatamente anterior
prevAll()	Selecciona todos los hermanos anteriores
prevUntil(*selector*)	Selecciona los hermanos anteriores para cada elemento hasta (sin incluir) un elemento que cumpla el *selector* o un elemento del objeto `jQuery` o del array `HTMLElement`.
prevUntil(*jQuery*)	
prevUntil(*HTMLElement[]*)	

6.6 Manipulando contenido - Crear contenido

- jQuery facilita métodos para crear, copiar, borrar y mover contenido
  - Permite envolver contenido dentro de otro.
- Soporte *Cross-Browser* para trabajar con CSS
  - Información sobre posicionamiento y tamaño.

- **Crear Contenido**
  - `$(cadenaHTML)`

```
var nuevoEncabezado = $("<h1>Encabezado Nivel 1</h1>");
```

Método	Propósito
html()	Devuelve el contenido *HTML* del primer elemento del conjunto de resultados
html(*nuevoContenido*)	Asigna el *nuevoContenido HTML* a todos los elementos del conjunto de resultados. Escapa los símbolos `<` y `>`
text()	Devuelve el contenido de todos los elemento del conjunto de resultados
text(*nuevoTexto*)	Asigna el *nuevoTexto* a todos los elementos del conjunto de resultados

Ejemplos creación de contenido

```
console.log($("#listado").html()); // <li class="a">elemento 0.......elemento 3</li>
console.log($("#listado").text()); // elemento 0 elemento 1 ... elemento 3
console.log($("li").html());  // elemento 0
console.log($("li").text());  // elemento 0elemento 1elemento 2elemento 3
$("#listado").html("<li>Nuevo elemento mediante jQuery</li>");
console.log($("#listado").html()); // <li>Nuevo elemento mediante jQuery</li>
$("p:first").html("Este es el primer párrafo con <br /> en medio");
$("p:last").text("Este es el último párrafo con <br /> en medio");
```

- Nuevo elemento mediante jQuery

Este es el primer párrafo con
en medio

Párrafo 1

Párrafo 2

Este es el último párrafo con
 en medio

- Al asignar nuevo contenido se elimina el contenido previo
- Mantiene los atributos de la etiqueta sobre la que se añade el contenido.

Trabajando con valores

- `val()` → Permite obtener y asignar valores a los elementos de un formulario

Método	Propósito
val()	Devuelve el valor del primer elemento del conjunto de resultados
val(*valor*)	Asigna el *valor* a todos los elementos del conjunto de resultados
val(*función*)	Asigna valores a todos los elementos del conjunto de resultados mediante la *función*

```
$("input").each(function(indice, elem) {
  console.log("Nombre: " + elem.name + " Valor: " + $(elem).val());
});
```

```
$("input").val(function(indice, valorActual) {
  return (indice + 1) * 100;
});
```

Trabajando con atributos

Método	Propósito
attr(*nombre*)	Obtiene el valor de la propiedad *nombre* del primero elemento del conjunto de resultados. Si el elemento no tiene un atributo con dicho nombre, devuelve `undefined`
attr(*objetoPropiedades*)	Asigna una serie de atributos en todos los elementos del conjunto de resultado mediante la sintaxis de notación de objeto, lo que permite asignar un gran número de propiedades de una sola vez `$("img").attr({ src:"/imgs/logo.gif", title:"JavaUA", alt: "Logo JavaUA"});`
attr(*clave*, *valor*)	Asigna *valor* a la propiedad *clave* para todos los elementos del conjunto de resultados
attr(*clave*, *función*)	Asigna una única propiedad a un valor calculado para todos los elementos del conjunto de resultados. En vez de pasar un valor mediante una cadena, la función devolverá el valor del atributo.
removeAttr(*nombre*)	Elimina el atributo *nombre* de todos los elementos del conjunto de resultados

Ejemplo atributos

```
<a href="imagenes/logo.jpg"><img src="imagenes/logo.jpg" alt="logo" /></a>
```

- Abrir el enlace en una nueva ventana

```
$("a").attr("target","_blank");
```

- Quitar el cuadrado que encubre la imagen al tratarse de un enlace

```
$("a").removeAttr("href");
```

- Cambiar la ruta de la imagen y su texto alternativo

```
$("img").attr({src:"imagenes/batman.jpg", alt:"Batman"});
```

Insertando contenido (*dentro*)

Método	Propósito
append(*contenido*)	Anexa contenido (al final) dentro de cada elemento del conjunto de resultados
appendTo(*selector*)	Traslada todos los elementos del conjunto de resultados detrás de los encontrados por el *selector*
prepend(*contenido*)	Añade contenido (al inicio) dentro de cada elemento del conjunto de resultados
prependTo(*selector*)	Traslada todos los elementos del conjunto de resultados delante de los encontrados por el *selector*

- Añadir un texto al final de todos los párrafos

```
$("p").append(" con nuevo contenido anexado");
```

- Añadir un texto al inicio de todos los párrafos

```
$("p").prepend("Nuevo contenido en el ");
```

- Trasladar el último párrafo delante del primero

```
$("p:last").prependTo("p:first");
```

Autoevaluación

- Suponiendo que tenemos una capa que contiene una capa con artículos ¿Qué realiza el siguiente código ?

```
$('#articulo').find('span.co').each(function() {
  var $this = $(this);
  $('<blockquote></blockquote>', {
    class: 'co',
    text: $this.text()
  }).prependTo( $this.closest('p') );
});
```

Insertando contenido (*delante/detrás*)

Método	Propósito
after(*contenido*)	Inserta el *contenido* detrás de cada elemento del conjunto de resultados
before(*contenido*)	Inserta el *contenido* antes de cada elemento del conjunto de resultados
insertAfter(*selector*)	Inserta todos los elementos del conjunto de resultados detrás de los encontrados por el *selector*
insertBefore(*selector*)	Inserta todos los elementos del conjunto de resultados delante de los encontrados por el *selector*

- Añadir un nuevo párrafo detrás de cada uno de los existentes

```
$("p").after("<p>Párrafo separado</p>");
$("<p>Párrafo separado</p>").insertAfter("p");
```

- Mover un elemento una posición hacia abajo

```
$("p:eq(1)").after(function() {
  return $(this).prev();
});
```

Duplicando contenido

- También podemos crear contenido duplicando contenido que ya existe.

Método	Propósito
clone()	Clona los elementos del conjunto de resultados y selecciona los clonados
clone(bool)	Clona los elementos del conjunto de resultados y todos sus manejadores de eventos y selecciona los clonados (pasándole `true`)

```
var vip2 = $("#vip").clone();
vip2.attr("id","vip2");
```

Modificando contenido - Envolver

Método	Propósito
wrap(*html*)	Envuelve cada elemento del conjunto de resultados con el contenido *html* especificado
wrap(*elemento*)	Envuelve cada elemento del conjunto de resultados con el *elemento* especificado
wrapAll(*html*)	Envuelve todos los elementos del conjunto de resultados con el contenido *html* especificado
wrapAll(*elemento*)	Envuelve todos los elementos del conjunto de resultados con un único *elemento*
wrapInner(*html*)	Envuelve los contenidos de los hijos internos de cada elemento del conjunto de resultados (incluyendo los nodos de texto) con una estructura *html*
wrapInner(*elemento*)	Envuelve los contenidos de los hijos internos de cada elemento del conjunto de resultados (incluyendo los nodos de texto) con un *elemento* DOM

- `wrapAll()` → si la selección no comparte un padre común. el nuevo elemento se inserta com padre del primer elemento seleccionado.
  - jQuery mueve el resto de elementos para que sean hermanos del primero.

Ejemplos envolver

- Envuelve cada párrafo con una capa de color rojo (tantas capas como párrafos)

```
$("p").wrap("<div style='color:red' />");
// <div style='color:red'><p>Párrafo 1</p></div>
// <div style='color:red'><p>Párrafo 2</p></div>
```

- Envuelve todos los párrafos con una capa de color rojo (una capa que envuelve a todos los párrafos)

```
$("p").wrapAll("<div style='color:red' />");
// <div style='color:red'><p>Párrafo 1</p><p>Párrafo 2</p>...</div>
```

- Envuelve el contenido de todos los párrafos con una capa de color rojo

```
$("p").wrapInner("<div style='color:red' />");
// <p><div style='color:red'>Párrafo 1</div></p>
// <p><div style='color:red'>Párrafo 2</div></p>
```

Sustituyendo/Borrando contenido

Método	Propósito
replaceWith(*contenido*)	Sustituye todos los elementos del conjunto de resultados con el *contenido* especificado, ya sea HTML o un elemento DOM
replaceAll(*selector*)	Sustituye los elementos encontrados por el *selector* con los del conjunto de resultados, es decir, lo contrario a `replaceWith`.
empty()	Elimina todos los nodos hijos del conjunto de resultados
remove()	Elimina todos los elementos del conjunto de resultados del DOM
detach()	Igual que `remove()` pero devuelve los elementos eliminados. Esto permite volver a insertarlos en otra parte del documento
detach(selector)	
unwrap()	Elimina el padre de cada uno de los elementos del conjunto de resultados, de modo que los elementos se convierten en hijos de sus abuelos.

Ejemplos sustitución/borrado

- Sustituye los párrafos (y su contenido) por capas

```
$("p").replaceWith("<div>Capa Nueva</div>");
$("<div>Capa Nueva</div>").replaceAll("p");
```

- Vacía la lista desordenada (la etiqueta *ul* sigue en el documento)

```
$("ul").empty();
```

- Elimina la lista desordenada (la etiqueta *ul* desaparece)

```
$("ul").remove();
var listaBorrada = $("ul").detach();
```

CSS - Estilos

Método	Propósito
css(*nombre*)	Devuelve el valor de la propiedad CSS *nombre* del primer elemento del conjunto de resultados
css(*propiedades*)	Asigna las propiedades de cada elemento del conjunto de resultados mediante la sintaxis de objeto: `var cssObj = {'background-color':'blue';'font-weight':'bold'}; $(this).set(cssObj);`
css(*propiedad*, *valor*)	Asigna un *valor* a una única *propiedad* CSS. Si se pasa un número, se convertirá automáticamente a un valor de píxel, a excepción de: `z-index`, `font-weight`, `opacity`, `zoom` y `line-height`.

```
var tamAnt = $("p").css("font-size");
$(“p").css("font-size", "1.5em");
$(“p").css("font-size", "+=5"); // 5 píxeles más
```

```
var valoresCSS = {
  "font-size": "1.5em",
  "color": "blue"
};
$("p").css("font-size", valoresCSS);
```

CSS - Clases

Método	Propósito
addClass(*clase1* [*clase2*])	Añade la *clase* especificada a cada elemento del conjunto de resultados
hasClass(*clase*)	Devueve `true` si está presenta la *clase* en al menos uno de los elementos del conjunto de resultados
removeClass(*clase* [*clase2*])	Elimina la *clase* de todos los elementos del conjunto de resultados
toggleClass(*clase* [*clase2*])	Si no está presente, añade la *clase*. Si está presente, la elimina.
toggleClass(*clase*, *cambio*)	Si *cambio* es `true` añade la *clase*, pero la elimina en el caso de que *cambio* sea `false`.

http://jsbin.com/tujuni/3/edit?html,css,js,console,output

```
$("p").addClass("a");
$("p:even").removeClass("b").addClass("a");

console.log("En algún elemento:" + $("p").hasClass("a"));
$("p").each(function(indice, elem) {
  console.log("Elemento: " + $(elem).text() + " -> " + $(elem).hasClass("a"));
});

$("p").toggleClass("a");
```

CSS - Posicionamiento

Método	Propósito
offset()	Obtiene el desplazamiento actual (en píxeles) del primer elemento del conjunto de resultados, respecto al documento
offsetParent()	Devuelve una colección *jQuery* con el padre posicionado del primer elemento
position()	Obtiene la posiciones superior e izquierda de un elemento relativo al desplazamiento de su padre (propiedades `top` y `left`)
scrollTop()	Obtiene el desplazamiento del scroll superior del primer elemento
scrollTop(*valor*)	Asigna el *valor* del desplazamiento del scroll superior a todos los elementos
scrollLeft()	Obtiene el desplazamiento del scroll superior del primer elemento
scrollLeft(*valor*)	Asigna el *valor* del desplazamiento del scroll izquierdo a todos los elementos

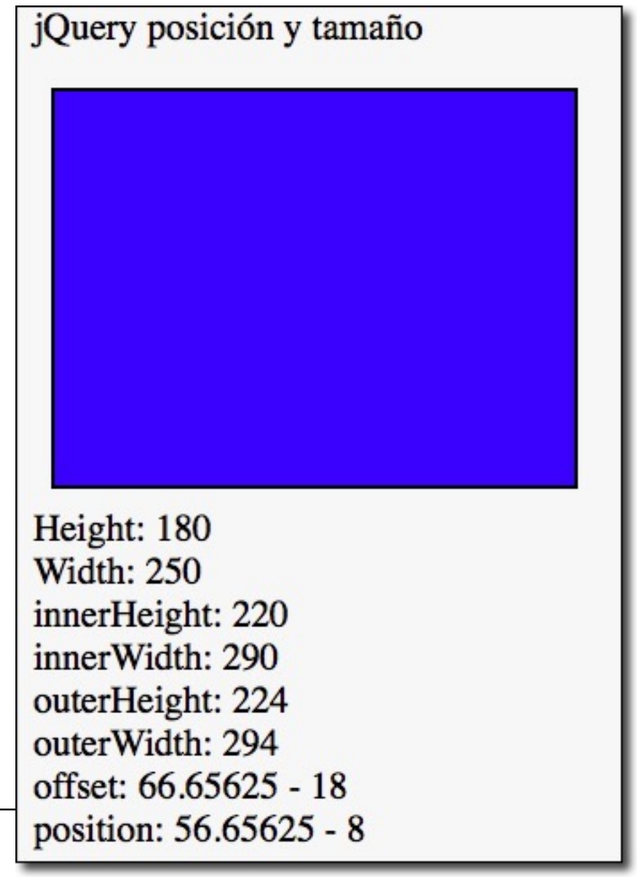
```
var pos = $("img").position();
console.log("Arriba:" + pos.top + " Izq:" + pos.left);
```

CSS - Cross Browser

Método	Propósito
height()	Obtiene la altura calculada en píxeles del primer elemento del conjunto de resultados
height(*valor*)	Asigna la altura CSS para cada elemento
width()	Obtiene la anchura calculada en píxeles del primer elemento
width(*valor*)	Asigna la anchura CSS para cada elemento del conjunto de resultados
innerHeight()	Obtiene la altura interna (excluye el borde e incluye el *padding*) para el primer elemento del conjunto de resultados
innerWidth()	Obtiene la anchura interna (excluye el borde e incluye el *padding*) para el primer elemento del conjunto de resultados
outerHeight(*margen*)	Obtiene la altura externa (incluye el borde y el *padding*) para el primer elemento del conjunto de resultados. Si el *margen* es `true`, también se incluyen los valores del margen.
outerWidth(*margen*)	Obtiene la anchura externa (incluye el borde y el *padding*) para el primer elemento del conjunto de resultados. Si el *margen* es `true`, también se incluyen los valores del margen.

Ejemplo Posición y Tamaño

```css
div#capa {
  width: 250px;
  height: 180px;
  margin: 10px;
  padding: 20px;
  background: blue;
  border: 2px solid black;
  cursor: pointer;
}
p, span {
  font-size: 16pt;
}
```




```
$("#height").html($("#capa").height());
$("#width").html($("#capa").width());
$("#innerH").html($("#capa").innerHeight());
$("#innerW").html($("#capa").innerWidth());
$("#outerH").html($("#capa").outerHeight());
$("#outerW").html($("#capa").outerWidth());
$("#offset").html($("#capa").offset().top + " - " + $("#capa").offset().left);
$("#position").html($("#capa").position().top + " - " + $("#capa").position().left);
```

Asociando datos

- HTML 5 → atributos con prefijo `data-`
- Cuidado que estos atributo no se renderizan como atributos del DOM → no los podemos inspeccionar mediante las *DevTools*

Método	Propósito
data(*clave*)	Obtiene el atributo `data-`*`clave`* del primer elemento del conjunto de resultados
data(*clave*, *valor*)	Almacena el *valor* en el atributo `data-`*`clave`* en cada elemento del conjunto de resultados
removeData()	Elimina todos los atributos de datos de cada elemento del conjunto de resultados
removeData(*clave*)	Elimina el atributo `data-`*`clave`* de cada elemento del conjunto de resultados

```
$("#listado").data("tipo","tutorial");
$("#listado").data("codigo", 123);
```

```
<ul id="#listado" data-tipo="tutorial" data-codigo="123">
```

```
$("#listado").removeData("tipo");
```

6.7 Eventos

- *jQuery* simplifica el tratamiento de los eventos
- Permite asociar un manejador a un grupo de elementos obtenido mediante selectores y filtros.
- Permite conectarse a y desconectarse de los eventos de manera *Cross-Browser* y dar soporte a IE8 y anteriores.
- Ofrece un objeto de evento que expone la mayoría de las propiedades comunes de manera *Cross-Browser*.
- Ofrece funciones que encapsulan las funcionalidades de eventos más comunes y funciones auxiliares con código *Cross-Browser*.
- http://api.jquery.com/category/events/

`document.ready`

- Similar a `window.onload`

```
$(document).ready(function() {
  // código jQuery
}
```

- Permite usar varias funciones sobre el mismo evento

```
$(document).ready(function() {
  // código alfa
}

$(document).ready(function() {
  // código beta
}
```

- `window.onload` → espera a que haya cargado toda la página, incluidas las imágenes.
- `$(document).ready()` → se lanza tras cargar el DOM, **sin esperar** a descargar todas las imágenes.
- Forma simplificada:

```
$(function() {
  // código jQuery
});
```

Adjuntando manejadores

- `on(eventos [,selector] [,datos] ,manejador(objetoEvento))`

```
$(selector).on("nombreDelEvento", function() {
  // código del manejador
})
```

- Antes de jQuery 1.7 en vez de `on()`, se usaba `bind()` con jQuery 1.0, `live()` con la versión 1.3 y `delegate()` a partir de la versión 1.4.2

```
$("p").on("click", function() {
  console.log("Click sobre un párrafo");
})
```

```
$("p").on("click mouseenter", function() {
  console.log("Click o mouseenter sobre un párrafo");
})
```

Delegación de eventos - Desconexión

- Al aplicar el evento `on`, en vez de seleccionar todos los elementos de un tipo que hay en DOM, es mejor elegir el padre del que estamos interesado y luego filtrar dentro del método.

```
$("dt").on("mouseenter", function() {
```

```
$("dl").on("mouseenter","dt",function() {
```

- Para desconectar un manejador de un evento:
  - `off(eventos [,selector] ,manejador(objetoEvento))`

Ejemplo eventos

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <style type="text/css">
    .normal {
      width:300px; height:200px;
      background-color:yellow;
      font-size:18pt;
    }
    .resaltado {
      background-color:red;
    }
  </style>
</head>
<body>
<h1>Eventos con jQuery</h1>
<div id="destinoEvento" class="normal">Pasar el ratón para ver el efecto. Click para
quitar/añadir el manejador.</div>
</body>
</html>
```

```
$(document).ready(function() {
  var dest = $("#destinoEvento");
  dest.on("mouseover mouseleave", function(evt) {
    dest.toggleClass("resaltado");
  });
  dest.on("click", function(evt) {
    dest.off("mouseover mouseleave");
    $("#destinoEvento").text("Manejadores eliminados");
  });
});
```

Métodos auxiliares

- Simplifican el uso de los eventos más importantes

- Ratón: **`click`**`()`, `dblclick()`, `focusout()`, **`hover`**`()`, `mousedown()`, `mouseenter()`, `mouseleave()`, `mousemove()`, `mouseout()`, `mouseover()`, `mouseup()`.
- Teclado: `focusout()`, `keydown()`, `keypress()`, `keyup()`.
- Formulario: `blur()`, `change()`, `focus()`, `focusin()`, `select()`, `submit()`.
- Navegador: `error()`, `resize()`, `scroll()`.

```
$("p").click(function() {
  // Manejador
});

$("p").on("click", function() {
  // Manejador
});
```

hover()

- Evento de Ratón
- Acepta dos manejadores, los cuales se ejecutarán cuando el ratón entre y salga del elemento respectivamente

```
$(function() {
  $("#destino").hover(resaltar, resaltar);

  $("#destino").click(fnClick1);
  $("#destino").dblclick(fnClick2);
});

function resaltar(evt) {
  $("#destino").toggleClass("resaltado");
}
function fnClick1() {
  $("#destino").html("Click!");
}
function fnClick2() {
  $("#destino").html("Doble Click!");
```

Eventos especiales

Método	Propósito
one(*tipo*, *datos*, *manejador*)	Permite capturar el evento *tipo* una sola vez para cada elemento del conjunto de resultado
trigger(*evento*, *datos*)	Lanza un evento para cada elemento del conjunto de resultados, lo que también provoca que se ejecute la acción predeterminada por el navegador. Por ejemplo, si le pasamos un *evento* `click`, el navegador actuará como si se hubiese *clickado* dicho elemento.
triggerHandler(*evento*, *datos*)	Dispara todos los manejadores asociados al *evento* sin ejecutar la acción predeterminada o burbujero del navegador. Sólo funciona para el primer elemento del conjunto de resultados del selector

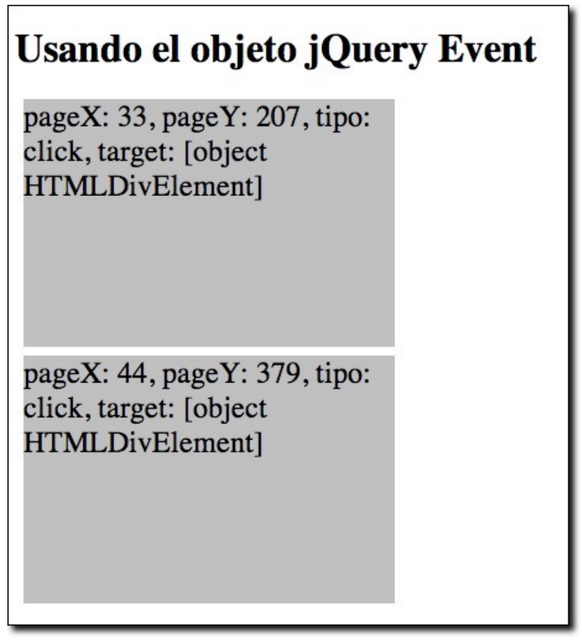
```
$(function() {
  $("div").one("click", function(evt) {
    $(this).css({
      background: "red", cursor: "auto"
    });
    console.log("Evento " + evt.type + " en div " + $(this).attr("id"));
  });
});
```

Objeto *jQuery* `Event`

Propiedad / Método	Propósito
type	Tipo del evento, por ejemplo, `click`
target	Elemento que lanzó el evento
data	Datos pasados al manejador
pageX, pageY	Coordenadas relativas al documento del ratón en el momento de lanzarse el
result	Valor devuelto por el último manejador
timestamp	Tiempo en el que se lanzó el evento
preventDefault()	Previene el comportamiento por defecto del navegador
idDefaultPrevented()	Averigua si se ha detenido el comportamiento por defecto
stopPropagation()	Detiene el burbujeo del evento hacia los elementos superiores
isPropagationStopped()	Averigua si se ha detenido el burbujeo del evento

Ejemplo objeto `jQuery Event`

```
<h1>Usando el objeto jQuery Event</h1>
<div id="div1" class="normal">Click para ver la información del evento</div>
<div id="div2" class="normal">Click para ver la información del evento</div>
```




```
$(function() {
  $("div").click(function(evt) {
      $(this).html("pageX: " + evt.pageX +
                   ", pageY: " + evt.pageY +
                   ", tipo: " + evt.type +
                   ", target: " + evt.target);
  });
});
```

Eventos personalizados

- Mediante `trigger()` / `triggerHandler()` podemos lanzar un evento personalizado

```
$('body').on('batClick', function() {
  console.log("Capturado el evento personalizado batClick");
});
$('body').trigger('batClick');
```

- Modelo de Subscripción

```
$.getJSON('http://www.omdbapi.com/?s=batman&callback=?', function(datos) {
  $(document).trigger('obd/datos', datos);  // publicador
});

$(document).on('obd/datos', function(evt, datos) {
  console.log('Subscriptor 1 recibe los datos');   // subscriptor 1
});

$(document).on('obd/datos', function(evt, datos) {
  console.log('Subscriptor 2 recibe los datos');   // subscriptor 2
});
```

6.8 `this`

- Al cargar una página, `this` referencia a `window`
- En una función, `this` referencia al padre

```
<button id="clickeame">Click aquí</button>
```

```
$("#clickeame").click(function(evt) {
  console.log(this); // HTMLButtonElement
  console.log(this.type); // "submit"
})
```

- Es común convertir `this` a un objeto jQuery

```
var $this = $(this);
```

Modificando `this` - `$.proxy()`

- `$.proxy(nombreDeLaFuncion, objetoQueSeraThis);`

```
var manejador =  {
  type: "Mi manejador de eventos",
  manejadorClick: function(evt) {
    console.log(this.type);
  }
};

$(function() {
  $("#clickeame").click(manejador.manejadorClick);
});
```

```
$(function() {
  $("#clickeame").click($.proxy(manejador.manejadorClick, manejador));
});
```

http://jsbin.com/budipe/9/edit?html,js,console,output

¿Preguntas?