FUNDAMENTOS DE JQUERY

por Javier Guede

www.guede.me

ÍNDICE

INTRODUCCIÓN A JQUERY

El lema de jQuery es "escribir menos, hacer más", se trata de una biblioteca, o un conjunto de complementos útiles para el lenguaje de programación JavaScript.

El propósito de jQuery es hacer mucho más fácil el uso de JavaScript en un sitio web.

jQuery lleva a cabo muchas tareas típicas que requieren muchas líneas de JavaScript para lograrlo y que los envuelve en métodos que se pueden llamar con una sola línea de código.

jQuery también simplifica muchas cosas complicadas de JavaScript, como las llamadas AJAX y la manipulación de DOM.

La librería de jQuery hace sencillo:

- Encontrar elementos en el documento HTML.
- Cambiar contenido HTML.
- Observar lo que hace un usario y actuar en consecuencia.
- Animar contenido de una página.
- Recuperar contenido.

Lo que se traduce en las siguientes características:

- Manipulación de HTML/DOM
- Manipulación de CSS
- Métodos de eventos HTML
- Efectos y animaciones
- AJAX
- Utilidades

Además, jQuery tiene tareas para casi cualquier tarea.

Para aprovechar jQuery al máximo tenemos que repasar cómo se compone una página HTML.

Un documento de HTML está estructurado de acuerdo con el **Document Object Model**, es decir, el Modelo de Objetos del Documento, o **DOM**. jQuery puede acceder y modificar el HTML interactuando con el DOM.

El DOM es una estructura de árbol creada por los navegadores y está compuesto de todos los elementos en la página, dispuestos en una jerarquía que refleja la forma en que el documento HTML está ordenado. Igual que en un documento de HTML, los elementos en el DOM pueden tener elementos padres, hijos y hermanos.

El DOM es una plataforma e interfaz de lenguaje neutral que permite a los programas y scripts, acceder dinámicamente y actualizar el contenido, estructura y estilo de un documento.

INSTALACIÓN JQUERY

Hay varias maneras de empezar a utilizar jQuery en un sitio web. Se puede:

- Descargar la librería jQuery desde www.jquery.com
- Incluir jQuery desde un CDN, como Google.

DESCARGARNDO JQUERY

Hay dos versiones de jQuery disponibles para descargar:

- Versión de producción: esta es la versión para subir al servidor porque el archivo está minificado y comprimido.
- Versión de desarrollo: esta es la versión para pruebas y desarrollo (descomprimido y código legible).

JQUERY CDN

Si no quieres descargar y vincular jQuery por ti mismo, se puede incluir desde un CDN (Content Delivery Network). Tanto Google como jQuery hospedan jQuery, se puede utilizar a través de los siguientes enlaces:

Google CDN:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
```

Microsoft CDN:

```
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.3.min.js"></script>
```

SINTAXIS JQUERY

Con jQuery se seleccionan (contenidos (query)) elemento HTML y se les realizas unas determinadas "acciones".

La sintaxis de jQuery está hecha a medida para la **selección** de elementos HTML y llevar a cabo algunas **acciones** sobre dichos elementos.

El "**;**" al final de cada sentencia es importante, es la manera en que jQuery sabe que le estamos dando una orden.

Sintaxis:

```
$(selector).action()
```

- Un signo $ para definir y acceder a jQuery.
- Un selector de elementos o contenidos de HTML.
- Una acción jQuery para aplicar al elemento.

Ejemplos:

```
$(this).hide() // Oculta el elemento actual
$("p").hide() // Oculta todos los elementos "p"
$(".test").hide() // Oculta todos los elementos con la clase test
$("#test").hide() // Oculta todos los elementos con el id test
```

EL EVENTO "DOCUMENT READY"

Este evento previene que cualquier código jQuery se ejecute antes de que la página web este totalmente cargada (esté preparada).

Si ponemos *document* entre paréntesis (no va entre comillas) se sabe que se va utilizar jQuery en el documento HTML donde se ha vinculado el archivo JavaScript. Mientras que *.ready()* es una función que se ejecuta una vez esté listo el documento HTML.

- *$(document)* es un objeto de jQuery. La parte de *$()* es en realidad una función disfrazada; convierte a *document* en un objeto de jQuery.
- *.ready()* es una especie de función; imagínate que es como un ayudante que ejecuta el código que tiene entre sus paréntesis, en cuanto el documento HTML está listo.
- *function(){}* es la acción que ejecutará *.ready()* tan pronto como se cargue el documento de HTML.

Sintaxis:

```
$(document).ready(function(){

   // los métodos de jQuery van aquí

});
```

Esto es una buena práctica de esperar a que el documento esté totalmente cargado y listo antes de trabajar con el. Esto también permite tener el código JavaScript antes del *body* de tu documento, en la sección *head*, sin necesidad de definir todos los *scripts* al final del *body*.

Existe un método abreviado del evento “document ready”.

Sintaxis:

```
$(function(){

   // jQuery methods go here...

});
```

SELECTORES JQUERY

Los selectores de jQuery son una de las partes más importantes de la librería jQuery.

Los selectores nos permiten seleccionar y manipular elementos HTML. Se utilizan para “encontrar” (o seleccionar) elementos HTML basados en sus id, clases, tipos, atributos, valores de atributos y mucho más. Se basa en los existentes selectores de CSS, y además tienen algunos selectores personalizados propios.

SELECTOR DE ELEMENTOS

El selector de elementos jQuery selecciona elementos basados en el nombre del elemento.

Ejemplo:

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});

/* Cuando el usuario pulsa el botón, todos los elementos p se
ocultarán */
```

SELECTOR COMPUESTO

Se pueden seleccionar varios elementos separando estos por comas.

Ejemplo:

```
$(document).ready(function(){
  $(‘p, li’).fadeTo(‘slow’, 0);
});
```

SELECTOR DE #ID

El selector de #id utiliza el atributo id de una etiqueta HTML para encontrar el elemento específico. Un id debe ser único en una página web, por tanto, se debe usar el selector #id cuando se quiere encontrar un elemento único.

Ejemplo:

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});

/* Cuando el usuario pulsa el botón, el elemento id=”test” se
ocultará */
```

SELECTOR DE CLASES

El selector de calases encuentra los elementos con una clase específica.

Ejemplo:

```
$(document).ready(function(){
  $("button").click(function(){
    $(.test").hide();
  });
});
/* Cuando el usuario pulsa el botón, los elementos class=”test” se ocultarán */
```

SELECTORES MÁS UTILIZADOS

EJEMPLO	SELECCIONA
$(“*”)	Selecciona todos los elementos
$(“this”)	Selecciona el elemento HTML actual
$(“p.intro”)	Selecciona todos los elementos <p> con class=”intro”
$(“p:first”)	Selecciona el primer elemento <p>
$(“ul li:first”)	Selecciona el primer elemento del primer
$(“ul li:first-child”)	Selecciona el primer elemento de todos los
$(“[href]”)	Selecciona todos los elementos con el atributo href
$(“a[target=’_blank’]”)	Selecciona todos los elementos a con atributo target igual a “_blank”
$(“a[target!=’_blank’]”)	Selecciona todos los elementos a con atributo target no igual a “_blank”
$(“:button”)	Selecciona todos los elementos <button> y elementos <input> de type = “button”
$(“tr:even”)	Selecciona todos los elementos <tr> pares
$(“tr:odd”)	Seleccionan todos los elementos <tr> impares

A través del siguiente enlace se pueden ver todos los selectores jQuery:

http://www.w3schools.com/jquery/jquery_ref_selectors.asp

SELECTORES JQUERY

La siguiente tabla lista todos los selectores jQuery disponibles:

SELECTOR	DESCRIPCIÓN	EJEMPLO
*	*$(“*”)* Selecciona todos los elementos del documento.	$(“*”)
#id	*$(“#id”)* Selecciona un elemento con un id específico (debe ser único en el documento).	$(“#lastname”)
.class	*$(“.class”)* Selecciona un elemento con una clase específica.	$(“.intro”)
.class,.class	*$(".class1,.class2,.class3,...")* Selecciona múltiples clases.	$(“.intro, .demo, .end”)
element	*$("element")* Selecciona todos los elementos con el nombre de un elemento específico.	$(“p”)
el1,el2,el3	*$("element1,element2,element3,...")* Selecciona múltiples elementos.	$(“h2, div, span”)
:first	*$(“:first”)* Selecciona el primer elemento.	$(“p:first”)
:last	*$(“:last”)* Selecciona el último elemento.	$(“p:last”)
:even	*$(“:even”)* Selecciona cada elemento con un índice par.	$(“tr:even”)
:odd	*$(“:odd”)* Selecciona cada elemento con un índice impar.	$(“tr:odd”)
:first-child	*$(“:frist-child”)* Selecciona todos los elementos que son el primer hijo de su padre.	$(“p:first-child”)
:first-of-type	*$(“:first-of-type”)* Selecciona todos los elementos que son el primer hijo , de un tipo particular, de su padre.	$(“p:first-of-type”)
:last-child	*$(“:last-child”)*	$(“p:last-child”)

SELECTOR	DESCRIPCIÓN	EJEMPLO
	Selecciona todos los elementos que son el último hijo de su padre.	
:last-of-type	*$(":last-of-type")* Selecciona todos los elementos que son el último hijo, de un tipo particular, de su padre.	$("p:last-of-type")
:nth-child(n)	*$(":nth-child(n|even|odd|formula)")* Selecciona todos los elementos que son el n índice hijo de su padre, independientemente del tipo.	$("p:nth-child(3)")
:nth-last-child(n)	*$(":nth-last-child(n|even|odd|formula)")* Selecciona todos los elementos que son el n índice hijo de su padre contando desde el último hijo, independientemente del tipo.	$("p:nth-last-child(3)")
:nth-of-type(n)	*$(":nth-last-child(n)")* Selecciona todos los elementos que son el n índice hijo de su, de un tipo particular.	$("p:nth-of-type(3)")
:nth-last-of-type(n)	*$(":nth-last-of-type(n|even|odd|formula)")* Selecciona todos los elementos que son el n índice hijo de su padre contando desde el último hijo, de un tipo particular.	$("p:nth-last-of-type(3)")
:only-child	*$(":only-child")* Selecciona todos los elementos que son el único hijo de su padre.	$("p:only-child")
:only-of-type	*$(":only-of-type")* Selecciona todos los elementos que son el único hijo de su tipo de su padre.	$("p:only-of-type")
parent > child	*$("parent > child")* Selecciona todos los elementos que son un hijo directo de un elemento específico.	$("div > span")
parent decendant	*$("parent descendant")* Selecciona todos los elementos que son descendientes de un elemento específico.	$("div span")
element + next	*$("element + next")* Selecciona el elemento siguiente de un elemento específico.	$("div + p")
element ~ siblings	*$("element ~ next")*	$("div ~ p")

SELECTOR	DESCRIPCIÓN	EJEMPLO
	Selecciona todos los elementos que son hermanos de un elemento específico. Ambos elementos deben compartir el mismo padre.	
:eq(index)	*$(":eq(index)")* Selecciona un elemento con un específico índice numérico.	$("p:eq(1)")
:gt(no)	*$(":gt(index)")* Selecciona elementos con un índice numérico mayor que un número específico.	$("tr:gt(3)")
:lt(no)	*$(":lt(index)")* Selecciona elementos con un índice numérico menor que un número específico.	$("ul li:lt(3)")
:not(selector)	*$(":not(selector)")* Selecciona todos los elementos excepto el elemento especificado.	$("input:not(:empty)")
:header	*$(":header")* Selecciona todos los elementos header (<h1> to < <h6>)	$(":header")
:animated	*$(":animated")* Selecciona todos los elementos que se encuentran actualmente animados.	$(":animated")
:focus	*$(":focus")* Selecciona todos los elementos que se encuentran actualmente enfocados.	$(":focus")
:contains(text)	*$(":contains(text)")* Selecciona todos los elementos que contienen una cadena específica. Puede estar contenida directamente como texto o en un elemento hijo.	$("p:contains(is)")
:has(selector)	*$(":has(selector)")* Selecciona todos los elementos que tienen uno o más elementos dentro de ellos, que coinciden con un selector específico.	$(":p:has(span)")
:empty	*$(":empty")* Selecciona todos los elementos vacíos.	$(":empty")
:parent	*$(":parent")*	$("td:parent")

SELECTOR	DESCRIPCIÓN	EJEMPLO
	Selecciona todos los elementos que son padre de otro elemento, incluyendo texto.	
:hidden	*$(":hidden")* Selecciona todos los elementos que son: - Establecidos como display:none. - Elementos de formulario con type="hidden" - Ancho y altura establecidad a cero. - Un elemento padre oculto.	$(":hidden").show()
:visible	*$(":visible")* Selecciona todos los elementos que son actualmente visibles.	$("p:visible")
:root	*$(":root")* Selecciona el elmento raíz del documento. En HTML, el elemento raíz es siempre <html>	$(":root").css("background-color", "yellow");
:lang(language)	*$(":lang(language)")* Selecciona todos los elementos con el atributo de idioma con un especifico valor.	$("p:lang(it)")
[attribute]	*$("[attribute]")* Selecciona cada elemento con un atributo específico.	$("[id]");
[attribute=value]	*$("[attribute='value']")* Selecciona cada elemento con un atributo y valor específicos.	$("[id='choose']");
[attribute!=value]	*$("[attribute!='value']")* Selecciona cada elemento diferente de un atributo y valor específicos.	$("p[class!='intro']");
[attribute$=value]	*$("[attribute$='value']")* Selecciona cada elemento con un atríbuto específico, con un valor que es el final de una cadena específica.	$("a[href$='.org']")
[attribute|=value]	*$("[attribute|='value']")* Selecciona cada elemento con un específico atributo, con un valor igual a una cadena específica o comenzando por esa cadena y seguida por un guion.	$("p[title|='Tomorrow']")
[attribute^=value]	*$("[attribute^='value']")*	$("input[name^='nation']")

SELECTOR	DESCRIPCIÓN	EJEMPLO
	Selecciona cada elemento con un atributo específico, con un valor que es el inicio de una cadena específica.	
[attribute~=value]	*$("[attribute~='value']")* Selecciona cada elemento con un atributo específico, con un valor que contiene una cadena específica.	$("input[name~='nation']")
[attribute*=value]	*$("[attribute*='value']")* Selecciona cada elemento con un atributo específico, con un valor que contiene una cadena o parte de ella.	$("input[name*='nation']")
:input	*$(":input")* Selecciona todos los elementos input de un formulario.	$(":input")
:text	*$(":text")* Selecciona todos los elementos input con el type=text.	$(":text")
:password	*$(":password")* Selecciona todos los elementos input con el type=password.	$(":password")
:radio	*$(":radio")* Selecciona todos los elementos input con el type=radio.	$(":radio")
:checkbox	*$(":checkbox")* Selecciona todos los elementos input con el type=checkbox.	$(":checkbox")
:submit	*$(":submit")* Selecciona todos los botones y elementos input con el type=submit.	$(":submit")
:reset	*$(":reset")* Selecciona todos los botones y elementos input con el type=reset.	$(":reset")
:button	*$(":button")* Selecciona todos los botones y elementos input con el type=button.	$(":button")
:image	*$(":image")*	$(":image")

SELECTOR	DESCRIPCIÓN	EJEMPLO
	Selecciona todos elementos input con el type=image.	
:file	*$(":file")* Selecciona todos los elementos input con el type=file.	$(":file")
:enabled	*$(":enabled")* Selecciona todos los elementos habilitados de un formulario.	$(":enabled")
:disabled	*$(":disabled")* Selecciona todos los elementos deshabilitados de un formulario.	$(":disabled")
:selected	*$(":selected")* Selecciona todos los elementos de opción que son preseleccionados.	$(":selected")
:checked	*$(":checked")* Selecciona todos los checkboxes o radio buttons chequeados.	$(":checked")

MÉTODOS DE EVENTOS JQUERY

jQuery está hecho a medida para responder a eventos en una página HTML.

Todas las diferentes acciones de los visitantes que una página web puede responder son conocidas como eventos. Algunos ejemplos son:

- Mover el cursor encima de un elemento
- Seleccionar un botón
- Clicar en un elemento

El término “disparar” es normalmente utilizado con eventos. Por ejemplo, el evento *keypress* dispara el momento en que tu pulsas una tecla.

Algunos eventos comunes del DOM:

RATÓN	TECLADO	FORMULARIO	VENTANA
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

En jQuery la mayoría de los eventos DOM tienen un método equivalente jQuery. Después de asignar un evento a un selector, hay que definir que sucederá cuando el evento sea disparado. Se debe pasar una función al evento.

Sintaxis:

```
$(selector).click(function(){
  // action goes here!!
});
```

MÉTODOS DE EVENTOS JQUERY COMUNMENTE UTILIZADOS

$(document).ready()

Nos permite ejecutar una función cuando el documento está totalmente cargado (ya explicado anteriormente).

click()

El método *click*() atribuye una función controlador de eventos a un elemento HTML. La función es ejecutada cuando el usuario clica sobre el elemento HTML.

Ejemplo:

```
$("p").click(function(){
    $(this).hide();
});
```

El ejemplo anterior dice que cuando un evento "*click*" se dispara sobe un elemento <p>, y se oculta el elemento actual.

dblclick()

El método *dblclick()* atribuye una función controlador de eventos a un elemento HTML. La función es ejecutada cuando el usuario hace doble clic sobre el elemento HTML.

Ejemplo:

```
$("p").dblclick(function(){
    $(this).hide();
});
```

mouseenter()

El método *mouseenter()* atribuye una función controlador de eventos a un elemento HTML. La función es ejecutada cuando el puntero del ratón pasa por encima del elemento HTML.

Ejemplo:

```
$("#p1").mouseenter(function(){
    alert("You entered p1!")
});
```

mouseleave()

El método *mouseleave()* atribuye una función controlador de eventos a un elemento HTML. La función es ejecutada cuando el puntero del ratón deja de estar encima del elemento HTML.

Ejemplo:

```
$("#p1").mouseleave(function(){
    alert(“Bye! You now leave p1!”)
});
```

mousedown()

El método *mousedown()* atribuye una función controlador de eventos a un elemento HTML. La función es ejecutada cuando el botón izquierdo, el del medio o el botón derecho es pulsado, mientras el ratón esta encima del elemento HTML.

Ejemplo:

```
$("#p1").mousedown(function(){
    alert("Mouse down over p1!");
});
```

mouseup()

El método *mouseup()* atribuye una función controlador de eventos a un elemento HTML. La función es ejecutada cuando el botón izquierdo, el del medio o el botón derecho es liberado, mientras el ratón esta encima del elemento HTML.

Ejemplo:

```
$("#p1").mouseup(function(){
    alert("Mouse up over p1!");
});
```

hover()

El método *hover()* puede asumir dos funciones y es una combinación de los métodos *mouseenter()* y *mouseleave().* La primera función es ejecutada cuando el cursor está encima del elemento HTML, y la segunda función es ejecutada cuando el puntero deja de estar encima del elemento HTML. Las dos funciones están separadas por una coma.

Ejemplo:

```
$("#p1").hover(function(){
    alert("You entered p1!");
},
function(){
    alert("Bye! You now leave p1!");
});
```

focus()

El método focus() atribuye una función controlador de eventos a un campo de un formulario HTML. La función es ejecutada cuando el campo del formulario está activo.

Ejemplo:

```
$("input").focus(function(){
    $(this).css("background-color", "#cccccc");
});
```

blur()

El método *blur()* atribuye una función controlador de eventos a un campo de un formulario HTML. La función es ejecutada cuando el campo del formulario está desactivo.

Ejemplo:

```
$("input").blur(function(){
    $(this).css("background-color", "#ffffff");
});
```

EL MÉTODO ON()

on()

El método *on()* atribuye una o más funciones controlador de eventos para los elementos seleccionados.

El método *on()* surgió con la versión de jQuery 1.7, cuyo objetivo es reemplazar a los antiguos *blind*, *live* y *delegate*. Al reemplazar a *live()* y *delegate()*, este comportamiento será heredado por cualquier otro elemento que se inserte en el DOM de forma dinámica aún después de haber sido ejecutada la función anterior. Esto quiere decir que funcionará para los elementos actuales y otros futuros (como un nuevo elemento creado por script).

Para eliminar los controladores de eventos, se utiliza el método *off().*

Sintaxis:

```
$(selector).on(events,childSelector,data,function,map)
```

- *events:* es un parámetro requerido, especifica uno o más eventos a agregar a los elementos seleccionados. Varios valores de eventos están separados por espacios (debe ser un evento válido).
- *childSelector:* es un parámetro opcional y especifica que el controlador de evento debe ser agregado únicamente a los elementos hijos indicados (y no al selector en si mismo).
- *data:* es un parámetro opcional que especifica cualquier tipo de datos que se necesite pasar a la función (suele corresponderse con un objeto jQuery).
- *function:* es un parámetro obligatorio yse corresponde con el callback o acción a realizar después de que el evento se dispare.
- *map:* especifica un evento *map* del tipo ({evento:function, event:function,…}) contiene uno o más eventos para agregar a los elementos seleccionados y funciones a ejecutar cuando el evento ocurre.

El método *on()* es realmente recomendable a usar en casi todas las asociaciones a eventos en jQuery ya que su rendimiento es algo más elevado.

Por ejemplo:

```
$(function(){
    $('.borrar').click(function() {
        // …
    });
});
```

Es menos óptimo que usar:

```
$(function(){
    $('.borrar').on('click',function() {
        // …
    });
});
```

Esto es porque jQuery usa *on()* internamente si usamos la primera forma, por lo que la segunda es la ruta más directa y por lo tanto, más óptima.

Los métodos "***blur, focus, focusin, focusout, load, resize, scroll, unload, click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select, submit, keydown, keypress, keyup, error*** *y* ***contextmenu***" utilizan internamente "*this.on".*

Se puede utilizar la siguiente sintaxis para asociar varios eventos a los mismos elementos:

Ejemplo:

```
$("p").on({
    mouseenter: function(){
        $(this).css("background-color", "lightgray");
    },
    mouseleave: function(){
        $(this).css("background-color", "lightblue");
    },
    click: function(){
        $(this).css("background-color", "yellow");
    }
}, "a.enlace");
```

EL MÉTODO OFF()

off()

Para la eliminación de eventos, contábamos con varios métodos como *unbind()*, *die()* o *undelegate()*. De nuevo, el objetivo principal de la nueva instrucción *off()* es reemplazarlos a todos de un modo consistente.

Sintaxis:

```
$(elements).off( [ events ] [, selector] [, handler] );
```

Con *off()*, todos los parámetros son opcionales. Cuando se utiliza en su forma más simple, *$(elements).off()*, se eliminan todos los eventos asociados al conjunto seleccionado.

MÉTODOS DE EVENTOS JQUERY

Disparador de métodos de eventos o adjuntar una función a un controlador de eventos para los elementos seleccionados. La siguiente tabla lista todos los métodos jQuery utilizados en controlador de eventos:

MÉTODOS	DESCRIPCIÓN
bind()	*$(selector).bind(event,data,function,map)* Atribuye uno o más controladores de eventos a los elementos seleccionados y especifica una función cuando esto sucede. Desde la versión 1.7 se utiliza el método *on*.
blur()	*$(selector).blur(function)* Dispara el evento *blur* que hace que un elemento se desenfoque, o agrega una función (opcional) que se ejecuta cuando el evento sucede. Se utiliza normalmente con el método *focus*.
change()	*$(selector).change(function)* Dispara el evento *change* que sucede cuando el valor de un elemento ha sido cambiado (sólo funciona en elementos <input>, <textarea> y <select>).
click()	*$(selector).click(function)* Dispara el evento *click* que sucede cuando un elemento ha sido clicado., o agrega una función (opcional) que se ejecuta cuando el evento sucede.
dblclick()	*$(selector).dblclick(function)* Dispara el evento *dblclick* que sucede cuando en un elemento se hace doble clic., o agrega una función (opcional) que se ejecuta cuando el evento sucede.
delegate()	*$(selector).delegate(childSelector,event,data,function)* Agrega uno o más controladores de eventos para los elementos específicos que son hijos de los elementos seleccionados, y especifica una función que se ejecuta cuando el evento sucede. También funciona con elementos creados dinámicamente. Desde la versión 1.7 se utiliza el método *on*.
die()	*$(selector).die(event,function)* Elimina todos los controladores de eventos añadidos con el método *live* , para los elementos seleccionados. Está obsoleto desde la versión 1.7 y eliminado en la versión 1.9.
error()	*$(selector).error(function)* Dispara el evento *error* que sucede cuando el elemento encuentra un error (si el elemento no es cargado correctamente), o agrega una función (opcional) que se ejecuta cuando el evento sucede. Está obsoleto desde la versión 1.8 de jQuery.
event.currentTarget	La propiedad *event.currentTarget* se corresponde con el elemento DOM actual dentro de la fase de propagación del evento, y normalmente es igual a *this.*

MÉTODOS	DESCRIPCIÓN
event.data	La propiedad *event.data* contiene los datos opcionales pasados a un método de eventos cuando el manejador de ejecución actual está ligado.
event.delegateTarget	La propiedad *event.delegateTarget* devuelve el elemento donde la llamada actual del controlador del evento de jQuery fue agregado. Esta propiedad es útil para eventos delegados agregados por el método *on*, donde el controlador de eventos es agregado a un antecedente del elemento que está siendo procesado.
event.isDefaultPrevented()	Comprueba si el método *preventDefault* fue llamado por el evento. Devuelve verdadero o falso.
event.isImmediatePropagationStopped()	Comprueba si el método *event.stopImmediatePropagation* fue llamado por el evento. Devuelve verdadero o falso
event.isPropagationStopped()	Comprueba si el método *event.stopPropagation* fue llamado por el evento. Devuelve verdadero o falso.
event.namespace	Devuelve el espacio de nombres cuando el evento fue disparado.
event.pageX	Devuelve la posición relativa con respecto al borde izquierdo del documento.
event.pageY	Devuelve la posición relativa con respecto al borde superior del documento.
event.preventDefault()	Detiene o anula la acción predeterminada de un evento que sucedería al elemento seleccionado.
event.relatedTarget	Devuelve que elemento está siendo excitado con el movimiento del ratón.
event.result	Contiene el último valor devuelto por un controlador de eventos disparado por un evento específico.
event.stopInmediatePropagation()	Detiene que el resto de controlador de eventos sean ejecutados.
event.stopPropagation()	Detiene el burbujeo de un evento para elementos padre, evitando cualquier controlador de eventos padre sea ejecutado.
event.target	Devuelve que elemento DOM dispara el evento.
event.timeStamp	Devuelve el número de milisegundos desde el 1 de Enero de 1970, cuando el evento es disparado.
event.type	Devuelve que tipo de evento fue disparado.

MÉTODOS	DESCRIPCIÓN
event.which	Devuelve que tecla del teclado o botón del ratón fue pulsado por el evento.
focus()	*$(selector).focus(function)* Dispara el evento *focus* que sucede cuando un elemento se vuelve *focus* (seleccionado con un clic o navegando con el tabulador), o agrega una función (opcional) que se ejecuta cuando el evento sucede.
focusin()	*$(selector).focusin(function)* Dispara el evento *focusin* que sucede cuando un elemento se vuelve *focus* (o cualquiera de los elementos dentro de el) (seleccionado con un clic o navegando con el tabulador), o agrega una función (opcional) que se ejecuta cuando el evento sucede.
focusout()	*$(selector).focusout(function)* Dispara el evento *focusin* que sucede cuando un elemento deja de ser *focus* (o cualquiera de los elementos dentro de el) (seleccionado con un clic o navegando con el tabulador), o agrega una función (opcional) que se ejecuta cuando el evento sucede.
hover()	*$(selector).hover(inFunction,outFunction)* Ejecuta dos funciones cuando el puntero del ratón hace *hover* (por encima y deja de estar encima) sobre los elementos seleccionados.
keydown()	*$(selector).keydown(function)* Dispara el evento *keydown* que sucede cuando una tecla es presionada, o agrega una función (opcional) que se ejecuta cuando el evento sucede.
keypress()	*$(selector).keypress(function)* Dispara el evento *keypress* que sucede cuando una tecla es presionada (pero no para todas las teclas, tales como Alt, Ctrl, Esc, Shift), o agrega una función (opcional) que se ejecuta cuando el evento sucede.
keyup()	*$(selector).keypress(function)* Dispara el evento *keyup* que sucede cuando una tecla es liberada, o agrega una función (opcional) que se ejecuta cuando el evento sucede.
live()	Eliminada desde la versión 1.9
load()	*$(selector).load(function)* Obsoleto desde la versión 1.8. Sucede cuando un elemento ha sido cargado.
mousedown()	*$(selector).mousedown(function)* Dispara el evento *mousedown* que sucede cuando el botón izquierdo del ratón es presionado, o agrega una función (opcional) que se ejecuta cuando el evento sucede.
mouseenter()	*$(selector).mouseenter(function)*

MÉTODOS	DESCRIPCIÓN
	Dispara el evento *mouseenter* que sucede cuando el puntero del ratón está encima del elemento seleccionado, o agrega una función (opcional) que se ejecuta cuando el evento sucede.
mouseleave()	*$(selector).mouseleave(function)* Dispara el evento *mouseleave* que sucede cuando el puntero del ratón deja de estar encima del elemento seleccionado, o agrega una función (opcional) que se ejecuta cuando el evento sucede.
mousemove()	*$(selector).mousemove(function)* Dispara el evento *mousemove* que sucede cuando el puntero del ratón se mueve dentro del elemento seleccionado, o agrega una función (opcional) que se ejecuta cuando el evento sucede.
mouseout()	*$(selector).mouseout(function)* Dispara el evento *mouseout* que sucede cuando el puntero del ratón deja de estar encima del elemento seleccionado (incluyendo sus elementos hijos), o agrega una función (opcional) que se ejecuta cuando el evento sucede.
mouseover()	*$(selector).mouseover(function)* Dispara el evento *mouseover* que sucede cuando el puntero del ratón está encima del elemento seleccionado (incluyendo sus elementos hijos), o agrega una función (opcional) que se ejecuta cuando el evento sucede.
mouseup()	*$(selector).mouseup(function)* Dispara el evento *mouseup* que sucede cuando el botón izquierdo del ratón es liberado, o agrega una función (opcional) que se ejecuta cuando el evento sucede.
off()	*$(selector).off(event,selector,function(eventObj),map)* Elimina controladores de eventos agregados con el método *on*. A partir de la versión 1.7 es el substituto de *unbind*.
on()	*$(selector).on(event,childSelector,data,function,map)* Agrega uno o más controladores de eventos a los elementos seleccionados y sus hijos. Desde la versión 1.7, substituye a los elementos *bind*, *live* y *delegate*.
one()	*$(selector).one(event,data,function)* Agrega uno o más controlador de eventos para los elementos seleccionados, y especifica una función a ejecutar cuando el evento sucede. Sólo se ejecuta una vez para cada elemento.
$.proxy()	*$(selector).proxy(function,context)* Toma una función existente y devuelve una nueva con un particular contexto.
ready()	*$(document).ready(function)* Dispara el evento *ready* que sucede cuando el DOM ha sido totalmente cargado. Es un buen lugar para tener todos los eventos y funciones jQuery.

MÉTODOS	DESCRIPCIÓN
resize()	*$(selector).resize(function)* Dispara el evento *resize* que sucede cuando la ventana del navegador cambia de tamaño, o agrega una función (opcional) que se ejecuta cuando el evento sucede.
scroll()	*$(selector).scroll(function)* Dispara el evento *scroll* que sucede cuando el usuario hace scroll en un elemento específico, o agrega una función (opcional) que se ejecuta cuando el evento sucede.
select()	*$(selector).select(function)* Dispara el evento *select* que sucede cuando un texto es seleccionado en un área ó campo de texto, o agrega una función (opcional) que se ejecuta cuando el evento sucede.
submit()	*$(selector).submit(function)* Dispara el evento *submit* que sucede cuando un formulario es enviado, o agrega una función (opcional) que se ejecuta cuando el evento sucede.
toggle()	Eliminado en la versión 1.9.
trigger()	*$(selector).trigger(event,eventObj,param1,param2,...)* Dispara un evento específico y el comportamiento por defecto de un evento para los elementos seleccionados.
triggerHandler()	*$(selector).triggerHandler(event,param1,param2,...)* Dispara un evento específico para los elementos seleccionados.
unbind()	*$(selector).unbind(event,function,eventObj)* Elimina controladores de eventos de los elementos seleccionados. Desde la versión 1.7, se utilizan los métodos *on* y *off.*
undelegate()	*$(selector).undelegate(selector,event,function)* Elimina uno o más controladores de eventos, añadidos con el método *delegate*.
unload()	*$(selector).unload(function)* Obsoleto desde la versión 1.7. Dispara el evento *unload* que ucede cuando el usuario navega para abandonar la página, en acciones tales como: - se hace clic en un link que abandona la página. - una nueva url es escrita en la barra de direcciones. - son utilizados los botones hacia delante y hacia atrás del navegador. - la ventana del navegador es cerrada. - la página es recargada.

FUNCIONES JQUERY

Las funciones son la unidad básica de acción de jQuery. Una función está compuesta de tres partes:

- La palabra clave *function.*
- Los *argumentos* que la función toma que van entre () y se separan por comas si se trata más de uno.
- Las acciones que la función debe ejecutar entre {}.

Sintaxis:

```
function(argumento1, argumento2, etc.) {
  Realizá una acción
  Realizá otra acción
  ¡Realizá otra acción más!
}
```

Una de las ventajas de jQuery es que el argumento de una función puede ser casi cualquier cosa, incluso puede ser otra función.

VARIABLES

Las variables nos sirven para guardar información que se quiere usar más adelante. Las variables pueden guardar cualquier tipo de información con la que trabajar.

El signo = se usa para asignar valores.

Ejemplo:

```
var lucky = 7;
var name = “Codecademy”;
var $p = $(‘p’);
```

Nuestro tercer ejemplo de variables es el resultado de un selector jQuery asignado a una variable.

Como se puede observar el identificador de la variable empieza por $, simplemente se trata de una cuestión de convenio para indicar que esa variable contiene un objeto de jQuery, pero realmente se puede nombrar de cualquier manera.

Sin embargo, $() es una función disfrazada que crea objetos jQuery.

OBJETO THIS

La palabra clave *this* se refiere al objeto jQuery con el que estamos trabajando en este momento.

Si se utiliza un controlador de eventos en un elemento, se puede llamar al evento que ocurre en $(this), y el evento solamente afectará al elemento con el que haces algo en ese momento (No lleva comillas).

Ejemplo 1:

```
$(document).ready(function() {
  $('div').mouseenter(function() {
    $('div').hide();
    });
});
```

Ejemplo 2:

```
$(document).ready(function() {
  $('div').mouseenter(function() {
    $(this).hide();
    });
});
```

En el ejemplo 1, al pasar el ratón por encima de algún elemento , se ocultarán todos los elementos , mientras que en el segundo ejemplo sólo se ocultará el elemento por el cual pasemos el cursor por encima.

EFECTOS JQUERY

MÉTODOS JQUERY HIDE() Y SHOW()

hide() y show()

Con jQuery se puede ocultar y mostrar elementos HTML con los métodos *hide()* y *show()*.

Sintaxis:

```
$(selector).hide(speed,callback);

$(selector).show(speed,callback);
```

Ejemplo:

```
$("#hide").click(function(){
    $("p").hide();
});

$("#show").click(function(){
    $("p").show();
});
```

El parámetro de velocidad opcional especifica la velocidad de ocultar/mostrar, y puede tomar los siguientes valores: "*slow*", "*fast*" o milisegundos.

El parámetro de "llamada de vuelta" (callback) opcional es una función para ser ejecutada después de que los métodos *hide()* y *show()* se hayan completado.

Ejemplo:

```
$("button").click(function(){
    $("p").hide(1000);
});
```

toggle ()

Con jQuery se puede intercambiar entre los métodos *hide()* y *show()* con el método *toggle()*.

Sintaxis:

```
$(selector).toggle(speed,callback);
```

Ejemplo:

```
$("button").click(function(){
    $("p").toggle();
});
```

MÉTODOS JQUERY "FADING"

Con jQuery se puede desvanecer un elemento dentro y fuera de la visibilidad. jQuery tiene los siguientes métodos de desvanecer:

- fadeIn()
- fadeOut()
- fadeToggle()
- fadeTo()

fadeIn()

El método *fadeIn()* es utilizado para hacer aparecer gradualmente un elemento oculto.

Sintaxis:

```
$(selector).fadeIn(speed,callback);
```

El parámetro de velocidad opcional especifica la duración del efecto. Puede tomar los siguientes valores: "*slow*", "*fast*" o milisegundos.

El parámetro de "llamada de vuelta" (callback) opcional es una función para ser ejecutada después de que el desvanecimiento se haya completado.

Ejemplo:

```
$("button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
});
```

fadeOut()

El método *fadeOut()* es utilizado para hacer desaparecer gradualmente un elemento visible.

Sintaxis:

```
$(selector).fadeOut(speed,callback);
```

Ejemplo:

```
$("button").click(function(){
    $("#div1").fadeOut();
    $("#div2").fadeOut("slow");
    $("#div3").fadeOut(3000);
});
```

fadeToggle()

El método *fadeToggle()* intercambia entre los métodos *fadeIn()* y *fadeOut()*.

Sintaxis:

```
$(selector).fadeToggle(speed,callback);
```

Ejemplo:

```
$("button").click(function(){
    $("#div1").fadeToggle();
    $("#div2").fadeToggle("slow");
    $("#div3").fadeToggle(3000);
});
```

fadeTo()

El método *fadeTo()* permite desvanecer un valor de opacidad dado (entre 0 y 1).

Sintaxis:

```
$(selector).fadeTo(speed,opacity,callback);
```

Los parámetros de velocidad y opacidad son obligatorios.

Ejemplo:

```
$("button").click(function(){
    $("#div1").fadeTo("slow", 0.15);
    $("#div2").fadeTo("slow", 0.4);
    $("#div3").fadeTo("slow", 0.7);
});
```

MÉTODOS JQUERY "SLIDING"

Con jQuery se pueden crear efectos de deslizamiento sobre los elementos. jQuery tiene los siguientes métodos de deslizamiento:

- slideDown()
- slideUp()
- slideToggle()

slideDown()

El método *slideDown()* es utilizado para deslizar hacia abajo un elemento.

Sintaxis:

```
$(selector).slideDown(speed,callback);
```

El parámetro de velocidad opcional especifica la duración del efecto. Puede tomar los siguientes valores: "*slow*", "*fast*" o milisegundos.

El parámetro de "llamada de vuelta" (callback) opcional es una función para ser ejecutada después de que el deslizamiento se haya completado.

Ejemplo:

```
$("#flip").click(function(){
    $("#panel").slideDown();
});
```

slideUp()

El método *slideUp()* es usado para deslizar hacia arriba un elemento.

Sintaxis:

```
$(selector).slideUp(speed,callback);
```

Ejemplo:

```
$("#flip").click(function(){
    $("#panel").slideUp();
});
```

slideToggle()

El método *slideToggle()* intercambia entre los métodos *slideDown()* y *slideUp().*

Sintaxis:

```
$(selector).slideToggle(speed,callback);
```

Ejemplo:

```
$("#flip").click(function(){
    $("#panel").slideToggle();
});
```

MÉTODOS JQUERY "ANIMATIONS"

animate()

El método *animate()* es utilizado para crear animaciones personalizadas.

Sintaxis:

```
$(selector).animate({params},speed,callback);
```

Los parámetros *params* requeridos definen las propiedades CSS para ser animados.

El parámetro de velocidad opcional especifica la duración del efecto. Puede tomar los siguientes valores: "*slow*", "*fast*" o milisegundos.

El parámetro de "llamada de vuelta"(callback) opcional es una función para ser ejecutada después de que la animación se haya completado.

Ejemplo:

```
$("button").click(function(){
    $("div").animate({left: '250px'});
});
```

animate() - Multiples propiedades

Se pueden animar múltiples propiedades al mismo tiempo.

Ejemplo:

```
$("button").click(function(){
    $("div").animate({
        left: '250px',
        opacity: '0.5',
        height: '150px',
        width: '150px'
    });
});
```

Casi todas las propiedades CSS pueden ser manipuladas, sin embargo todas las propiedades deben ser llamadas con sus nombres correspondientes cuando se utiliza el método *animate()*. Por ejemplo: *paddingLeft* en vez de *padding-left*, *marginRight* en vez de *margin.right*, etc.

La animación de colores no está incluida en el núcleo de la librería jQuery. Para animar color se necesita descargar un plugin para animaciones de color desde jquery.com (http://plugins.jquery.com/).

animate() - Utilizando valores relativos

También se pueden definir valores relativos (los valores son relativos a los valores actuales de los elementos). Esto se hace poniendo += or -= delante del valor.

Ejemplo:

```
$("button").click(function(){
    $("div").animate({
        left: '250px',
        height: '+=150px',
        width: '+=150px'
    });
});
```

animate() - Utilizando valores predefinidos

Se puede especificar los valores de las propiedades de la animación: “show”, “hide”, or “toggle”.

Ejemplo:

```
$("button").click(function(){
    $("div").animate({
        height: 'toggle'
    });
});
```

animate() - Utilizando funcionalidades en cola

Por defecto, jQuery viene con funcionalidad de colas para animaciones. Esto significa que si tu escribes múltiples llamadas *anímate()* una después de la otra, jQuery creo una cola interna con esas llamadas. Entonces se ejecutarán las llamadas una a una.

Ejemplo:

```
$("button").click(function(){
    var div = $("div");
    div.animate({height: '300px', opacity: '0.4'}, "slow");
    div.animate({width: '300px', opacity: '0.8'}, "slow");
    div.animate({height: '100px', opacity: '0.4'}, "slow");
    div.animate({width: '100px', opacity: '0.8'}, "slow");
});
```

A continuación se muestran las propiedades que pueden ser animadas:

PROPIEDADES CSS			
backgroundPositionX	marginLeft	width	letterSpacing
backgroundPositionY	marginRight	maxHeight	wordSpacing
borderWith	marginTop	maxWidth	lineHeight
borderBottomWidth	outlineWidth	minHeight	textIndent
borderLeftWidth	padding	minWidth	

borderRightWidth	paddingBottom	fontSize	
borderTopWidth	paddingLeft	bottom	
borderSpacing	paddingRight	left	
Margin	paddingTop	right	
marginBottom	height	top	

MÉTODOS JQUERY "STOP ANIMATIONS"

stop()

El método *stop()* es utilizado para parar una animación o efecto antes de que esta haya terminado.

El método *stop()* trabaja para todos los efectos, incluyendo desvanecimiento, deslizamiento y animaciones personalizadas.

Sintaxis:

```
$(selector).stop(stopAll,goToEnd);
```

El parámetro *stopAll* opcional si la cola de animación también debe ser limpiada o no. Por defecto es *false*, el cual significa que sólo se parará la animación en curso, permitiendo a cualquier animación en cola ser realizada posteriormente.

El parámetro *goToEnd* especifica si o no completar la animación actual inmediatamente. Por defecto es *false*.

Por lo tanto, por defecto, el método *stop()* termina con la animación actual siendo realizada sobre el elemento seleccionado.

Ejemplo:

```
$("#stop").click(function(){
    $("#panel").stop();
});
```

FUNCIONES JQUERY "CALLBACK"

Las sentencias de JavaScript son ejecutadas línea por línea. Sin embargo, con efectos, la siguiente línea de un código puede ser ejecutada, aunque el efecto no haya terminado. Esto puede crear errores.

Para prevenir esto, se puede crear una función *callback*. Una función *callback* es ejecutada después de que el efecto actual haya terminado.

Sintaxis:

```
$(selector).hide(speed,callback);
```

Ejemplo:

```
$("button").click(function(){
    $("p").hide("slow", function(){
        alert("The paragraph is now hidden");
    });
});
```

TÉCNICA JQUERY "CHAINING"

En jQuery hay una técnica llamada "chaining" (encadenar) que permite ejecutar múltiples comandos, uno después del otro, sobre el mismo elemento. De este modo, los navegadores no tienen que encontrar el mismo elemento más de una vez.

Para encadenar una acción, simplemente se añade la acción después de otra acción.

Ejemplo:

```
$("#p1").css("color", "red").slideUp(2000).slideDown(2000);
```

Cuando encadenamos, la línea de código puede ser bastante larga. Para evitar esto, en jQuery se pueden incluir saltos de línea.

Ejemplo:

```
$("#p1").css("color", "red")
  .slideUp(2000)
  .slideDown(2000);
```

MÉTODOS DE EFECTOS JQUERY

La siguiente tabla lista todos los métodos jQuery utilizados para la creación de efectos de animación:

MÉTODOS	DESCRIPCIÓN
animate()	*$ (selector).animate({styles},speed,easing,callback)* Ejecuta una animación personalizable con propiedades CSS sobre los elementos seleccionados. Sólo se pueden animar valores numéricos.
clearQueue()	*$(selector).clearQueue(queueName)* Elimina todos los elementos de la cola que no han sido ejecutados todavía. Cuando una función a empezado a ejecutarse, lo hace hasta que se haya completado.
delay()	*$(selector).delay(speed,queueName)*

MÉTODOS	DESCRIPCIÓN
	Establece un temporizador para retrasar la ejecución del siguiente elemento de la cola (acepta milisegundos, *slow* y *fast* como argumentos).
dequeue()	*$(selector).dequeue(queueName)* Elimina la siguiente función de la cola, y a continuación ejecuta la función.
fadeIn()	*$(selector).fadeIn(speed,easing,callback)* Cambia gradualmente la opacidad de oculto a visible, de los elementos seleccionados.
fadeOut()	*$(selector).fadeOut(speed,easing,callback)* Cambia gradualmente la opacidad de visible a oculto, de los elementos seleccionados.
fadeTo()	*$(selector).fadeTo(speed,opacity,easing,callback)* Cambia gradualmente la opacidad a un valor específico, de los elementos seleccionados.
fadeToggle()	*$(selector).fadeToggle(speed,easing,callback)* Intercambia entre los métodos *fadeIn* y *fadeOut*.
finish()	*$(selector).finish(queueName)* Este método detiene las animaciones en curso, elimina todas las animaciones en la cola, y completa el resultado final de la animación.
hide()	*$(selector).hide(speed,easing,callback)* Oculta los elementos seleccionados. Es similar a la propiedad CSS display: none.
queue()	*$(selector).queue(queueName)* Muestra la cola de funciones a ser ejecutada sobre los elementos seleccionados. Puede ser una o más funciones esperando a ser ejecutadas.
show()	*$(selector).show(speed,easing,callback)* Muestra los elementos ocultos seleccionados.
slideDown()	*$(selector).slideDown(speed,easing,callback)* Desliza hacia abajo los elementos seleccionados.
slideToggle()	*$(selector).slideToggle(speed,easing,callback)* Intercambia entre los métodos *slideDown* y *slideUp*.
slideUp()	*$(selector).slideUp(speed,easing,callback)* Desliza hacia arriba los elementos seleccionados.
stop()	*$(selector).stop(stopAll,goToEnd)* Para la animación en curso para los elementos seleccionados.

MÉTODOS	DESCRIPCIÓN
toggle()	*$(selector).toggle(speed,easing,callback)* Intercambia entre los métodos *show* y *hide*.

MANIPULACIÓN DE ELEMENTOS Y ATRIBUTOS HTML CON JQUERY

Una parte importante de jQuery es la posibilidad de manipular el DOM. jQuery viene con un montón de métodos DOM relacionados que hacen que sea fácil de acceder y manipular elementos y atributos.

MÉTODOS JQUERY GET

Get Content - text(), html() y val()

Tres simples pero útiles métodos jQuery para manipular el DOM que son:

- *text()*: Establece o devuelve el contenido del texto de los elementos seleccionados.
- *html()*:Establece o devuelve el contenido de los elementos seleccionados (incluyendo el formato HTML).
- *val()*: Establece o devuelve el valor de los campos del formulario.

Ejemplo:

```
$("#btn1").click(function(){
    alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
    alert("HTML: " + $("#test").html());
});
```

Get Attributes - attr()

El método *attr()* es utilizado para conseguir valores de atributos:

Ejemplo:

```
$("button").click(function(){
    alert($("#w3s").attr("href"));
});
```

MÉTODOS JQUERY SET

Set Content - text(), html() y val()

Se utilizarán los mismos tres métodos para establecer contenidos:

- *text()*: Establece o devuelve el contenido del texto de los elementos seleccionados.
- *html()*:Establece o devuelve el contenido de los elementos seleccionados (incluyendo el formato HTML).

- *val()*: Establece o devuelve el valor de los campos del formulario.

Ejemplo:

```
$("#btn1").click(function(){
    $("#test1").text("Hello world!");
});
$("#btn2").click(function(){
    $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
    $("#test3").val("Dolly Duck");
});
```

Una función callback para text(), html() y val()

Todos los tres elementos: *text(), html()* y *val(),* vienen con una función callback. Esta función tiene dos parámetros: el índice del elemento actual en la lista de elementos seleccionados y el valor original (viejo). Entonces devuelve la cadena que se desea utilizar como el nuevo de la función.

Ejemplo:

```
$("#btn1").click(function(){
    $("#test1").text(function(i, origText){
        return "Old text: " + origText + " New text: Hello world!
        (index: " + i + ")";
    });
});

$("#btn2").click(function(){
    $("#test2").html(function(i, origText){
        return "Old html: " + origText + " New html: Hello
<b>world!</b>
        (index: " + i + ")";
    });
});
```

Set Attributes - attr()

El método *attr()* también es utilizado para establecer/cambiar valores de atributos.

Ejemplo:

```
$("button").click(function(){
    $("#w3s").attr("href", "http://www.w3schools.com/jquery");
});
```

También permite poner varios atributos al mismo tiempo.

Ejemplo:

```
$("button").click(function(){
    $("#w3s").attr({
        "href" : "http://www.w3schools.com/jquery",
        "title" : "W3Schools jQuery Tutorial"
    });
});
```

Una función callback para attr()

El método *attr()* también viene con una función *callback*. Esta función tiene dos parámetros: el índice del elemento actual en la lista de elementos seleccionados y el valor original (viejo). Entonces devuelve la cadena que se desea utilizar como el nuevo de la función.

Ejemplo:

```
$("button").click(function(){
    $("#w3s").attr("href", function(i, origValue){
        return origValue + "/jquery";
    });
});
```

MÉTODOS JQUERY ADD

A continuación, se detallan los cuatro métodos de jQuery que se utilizan para añadir nuevo contenido:

- *append()*: Inserta contenido al final de los elementos seleccionados.
- *preprend()*: Inserta contenido al principio de los elementos seleccionados.
- *afeter()*: Inserta contenido después de los elementos seleccionados.
- *before()*: Inserta contenido antes de los elementos seleccionados.

Método append()

El método *append()* inserta contenido al final de los elementos HTML seleccionando.

Ejemplo:

```
$("p").append("Some appended text.");
```

Método prepend()

El método *prepend()* inserta contenido al principio de los elementos HTML seleccionando.

Ejemplo:

```
$("p").prepend("Some prepended text.");
```

Añadir varios nuevos elementos con append() y prepend()

Ambos métodos *append()* y *prepend()* pueden llevar un número infinito de nuevos elementos y parámetros. Los nuevos elementos pueden ser generados con el texto HTML, con jQuery o con código JavaScript y elementos DOM.

Ejemplo:

```
unction appendText() {
    var txt1 = "<p>Text.</p>";               // Create element
with HTML
    var txt2 = $("<p></p>").text("Text.");   // Create with jQuery
    var txt3 = document.createElement("p");  // Create with DOM
    txt3.innerHTML = "Text.";
    $("p").append(txt1, txt2, txt3);         // Append the new
elements
}
```

Método after() y before()

El método *after()* inserta contenido después de los elementos HTML seleccionados.

El método *before()* inserta contenido antes de los elementos HTML seleccionados.

Ejemplo:

```
$("img").after("Some text after");

$("img").before("Some text before");
```

Añadir varios nuevos elementos con after() y before()

Ambos métodos *after()* y *before()* pueden llevar un número infinito de nuevos elementos como parámetros. Los nuevos elementos pueden ser generados con texto HTML, con jQuery o código JavaScript y elementos DOM.

Ejemplo:

```
function afterText() {
    var txt1 = "<b>I </b>";                  // Create element
with HTML
    var txt2 = $("<i></i>").text("love ");   // Create with
jQuery
    var txt3 = document.createElement("b");  // Create with DOM
    txt3.innerHTML = "jQuery!";
    $("img").after(txt1, txt2, txt3);        // Insert new
elements after <img>
}
```

MÉTODOS JQUERY REMOVE

A continuación, se detallan los cuatro métodos de jQuery que se utilizan para quitar contenidos y elementos:

- *remove()*: Elimina los elementos seleccionados (y sus elementos hijos).
- *empty()*: Elimina los elementos hijos desde el elemento seleccionado

Método remove()

El método *remove()* elimina los elementos seleccionando y sus elementos hijos.

Ejemplo:

```
$("#div1").remove();
```

El método *remove()* también acepta un parámetro, el cual permite filtrar los elementos a ser eliminados.

El siguiente ejemplo elimina los elementos <p> con las clases “test” y “demo”.

Ejemplo:

```
$("p").remove(".test, .demo");
```

Método empty()

El método *empty()* elimina los elementos hijos de los elementos seleccionados.

Ejemplo:

```
$("#div1").empty();
```

MANIPULANDO CSS | JQUERY CSS CLASSES

jQuery tiene varios métodos para la manipulación de CSS, tales como:

- *addClass()*:Añade una o más clases a los elementos seleccionados.
- *removeClass()*: Elimina una o más clases de los elementos seleccionados.
- *toggleClass()*: Intercambia entre añadir/eliminar clases de los elementos seleccionados.
- *css()*: Establece o devuelve el atributo de estilo.

Método addClass()

El método *addClass()* añade una o más clases CSS a los elementos seleccionados (puede ser más de un elemento seleccionado).

Ejemplo:

```
$("button").click(function(){
    $("h1, h2, p").addClass("blue");
    $("div").addClass("important");
});
```

También se pueden especificar varias clases dentro del método *addClass()*:

Ejemplo:

```
$("button").click(function(){
    $("#div1").addClass("important blue");
});
```

Método removeClass()

El método *removeClass()* elimina una o más clases CSS a los elementos seleccionados (puede ser más de un elemento seleccionado).

Ejemplo:

```
$("button").click(function(){
    $("h1, h2, p").removeClass("blue");
});
```

Método toggleClass()

El método *toggleClass()* intercambia entre los métodos *addClass()* y *removeClass()*.

Ejemplo:

```
$("button").click(function(){
    $("h1, h2, p").toggleClass("blue");
});
```

Método css()

El método *css()* tiene dos usos:

1. Devolver el valor de una propiedad CSS específica. Devuelve el valor del primer elemento en juego.

   Sintaxis:

   ```
   css("propertyname");
   ```

   Ejemplo:

   ```
   $("p").css("background-color");
   ```

2. Establecer una propiedad CSS.

   Sintaxis:

```
css("propertyname","value");
```

   Ejemplo:

```
$("p").css("background-color", "yellow");
```

También se pueden establecer varias propiedades CSS.

Sintaxis:

```
css({"propertyname":"value","propertyname":"value",...});
```

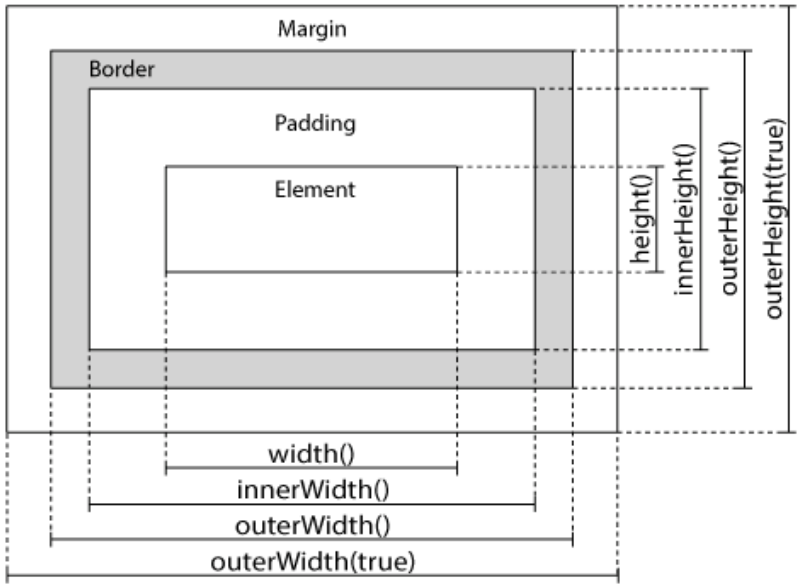
Ejemplo:

```
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

MÉTODOS JQUERY DIMENSIONS

Resulta sencillo trabajar con las dimensiones de elementos y la ventana del navegador. jQuery tiene varios métodos para trabajar con ellas:

- *width()*
- *height()*
- *innerWidth()*
- *innerHeight()*
- *outerWidth()*
- *outerHeight()*



Método width() y height()

El método *width()* establece o devuelve el ancho de un elemento (excluyendo el padding, borde y margen).

El método *height()* establece o devuelve el alto de un elemento (excluyendo el padding, borde y margen).

Ejemplo:

```
$("button").click(function(){
    var txt = "";
    txt += "Width: " + $("#div1").width() + "</br>";
    txt += "Height: " + $("#div1").height();
    $("#div1").html(txt);
});
```

El siguiente ejemplo devuelve el ancho y altura de un documento y ventana del navegador.

Ejemplo:

```
$("button").click(function(){
    var txt = "";
    txt += "Document width/height: " + $(document).width();
    txt += "x" + $(document).height() + "\n";
    txt += "Window width/height: " + $(window).width();
    txt += "x" + $(window).height();
    alert(txt);
});
```

El siguiente ejemplo establece el ancho y altura de un elemento concreto.

Ejemplo:

```
$("button").click(function(){
    $("#div1").width(500).height(500);
});
```

Método innerWidth() y innerHeight()

El método innerW*idth()* devuelve el ancho de un elemento incluyendo el padding.

El método innerH*eight()* devuelve el alto de un elemento incluyendo el padding.

Ejemplo:

```
$("button").click(function(){
    var txt = "";
    txt += "Inner width: " + $("#div1").innerWidth() + "</br>";
    txt += "Inner height: " + $("#div1").innerHeight();
    $("#div1").html(txt);
});
```

Método outerWidth() y outerHeight()

El método *outerWidth()* devuelve el ancho de un elemento incluyendo el padding y el borde.

El método *outerHeight()* devuelve el alto de un elemento incluyendo el padding y el borde.

Ejemplo:

```
$("button").click(function(){
    var txt = "";
    txt += "Outer width: " + $("#div1").outerWidth() + "</br>";
    txt += "Outer height: " + $("#div1").outerHeight();
    $("#div1").html(txt);
});
```

El método *outerWidth(true)* devuelve el ancho de un elemento incluyendo el padding, el borde y el margin.

El método *outerHeight(true)* devuelve el alto de un elemento incluyendo el padding, el borde y el margin.

Ejemplo:

```
$("button").click(function(){
    var txt = "";
    txt += "Outer width (+margin): " + $("#div1").outerWidth(true)
+ "</br>";
    txt += "Outer height (+margin): " +
$("#div1").outerHeight(true);
    $("#div1").html(txt);
});
```

MÉTODOS HTML/CSS JQUERY

La siguiente tabla lista todos los métodos utilizados para manipular HTML y CSS. Los siguientes métodos trabajan tanto para documentos HTML como para XML, a excepción de html().

MÉTODOS	DESCRIPCIÓN
addClass()	*$(selector).addClass(classname,function(index,oldclass))* Añade una o más clases a los elementos seleccionados.
after()	*$(selector).after(content,function(index))* Inserta contenido después del elemento seleccionado.
append()	*$(selector).append(content,function(index,html))* Inserta contenido al final de los elementos seleccionados.
appendTo()	*$(content).appendTo(selector)* Inserta elementos HTML al final de los elementos seleccionados.
attr()	*$(selector).attr(attribute)* *$(selector).attr(attribute,value)* *$(selector).attr({attribute:value, attribute:value,...})* Devuelve o establece valores de atributos de los elementos seleccionados.
before()	*$(selector).before(content,function(index))* Inserta contenido antes de los elementos seleccionados.
clone()	*$(selector).clone(true|false)* Hace una copia de los elementos seleccionados.
css()	*$(selector).css(property)* *$(selector).css(property,value)* *$(selector).css({property:value, property:value, ...})* Devuelve o establece una o más propiedades de estilos para los elementos seleccionados.
detach()	*$(selector).detach()* Elimina los elementos seleccionados incluyendo sus hijos (mantiene datos y eventos)
empty()	*$(selector).empty()* Elimina todos los elementos hijos y su contenido de los elementos seleccionados.
hasClass()	*$(selector).hasClass(classname)* Comprueba si alguno de los elementos seleccionados tiene una clase específica. Devuelve "true" en caso afirmativo.

MÉTODOS	DESCRIPCIÓN
height()	*$(selector).height()* *$(selector).height(value)* Establece o devuelve la altura de los elementos seleccionados.
html()	*$(selector).html()* *$(selector).html(content)* Establece o devuelve el contenido interior de los elementos seleccionados.
innerHeight()	*$(selector).innerHeight()* Devuelve la altura de un elemento (incluido el padding, pero no el borde).
innerWidth()	*$(selector).innerWidth()* Devuelve el ancho de un elemento (incluido el padding, pero no el borde).
insertAftert()	*$(content).insertAfter(selector)* Inserta elementos HTML después de los elementos seleccionados.
insertBefore()	*$(content).insertBefore(selector)* Inserta elementos HMTL antes de los elementos seleccionados.
offset()	*$(selector).offset()* *$(selector).offset({top:value,left:value})* Devuelve o establece las coordenadas (relativas al documentos) para los elementos seleccionados.
offsetParent()	*$(selector).offsetParent()* Devuelve el primer elemento padre posicionado.
outerHeight()	*$(selector).outerHeight(includeMargin)* Devuelve la altura de un elemento (incluyendo el padding y el borde).
outerWidth()	*$(selector).outerWidth(includeMargin)* Devuelve el ancho de un elemento (incluyendo el padding y el borde).
position()	*$(selector).position()* Devuelve la posición de un elemento, relativa al elemento padre.
prepend()	*$(selector).prepend(content,function(index,html))* Inserta contenido al principio de los elementos seleccionados.
prependTo()	*$(content).prependTo(selector)* Inserta elementos HTML al principio de los elementos seleccionados.

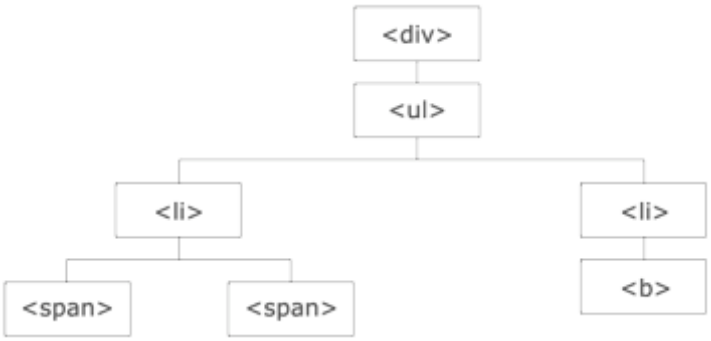
MÉTODOS	DESCRIPCIÓN
prop()	*$(selector).prop(property)* *$(selector).prop(property,value)* *$(selector).prop({property:value, property:value,...})* Establece o devuelve valor de propiedades de los elementos seleccionados.
remove()	*$(selector).remove(selector)* Elimina los elementos seleccionados (incluyendo datos y eventos).
remoreAttr()	*$(selector).removeAttr(attribute)* Elimina uno o más atributos de los elementos seleccionados.
removeClass()	*$(selector).removeClass(classname,function(index,currentclass))* Elimina una o más clases de los elementos seleccionados.
removeProp()	*$(selector).removeProp(property)* Elimina una propiedad establecida con el método *prop*.
replaceAll()	*$(content).replaceAll(selector)* Reemplaza elementos seleccionados con nuevos elementos HTML.
replaceWith()	*$(selector).replaceWith(content,function(index))* Reemplaza elementos seleccionados con nuevo contenido.
scrollLeft()	*$(selector).scrollLeft()* *$(selector).scrollLeft(position)* Devuelve o establece la posición de la barra de scroll horizontal de los elementos seleccionados.
scrollTop()	*$(selector).scrollTop()* *$(selector).scrollTop(position)* Devuelve o establece la posición de la barra de scroll vertical de los elementos seleccionados.
text()	*$(selector).text()* *$(selector).text(content)* Devuelve o establece contenido de texto de los elementos seleccionados.
toggleClass()	*$(selector).toggleClass(classname,function(index,currentclass),switch)* Intercambia entre los métodos *addClass* y *removeClass* de los elementos seleccionados.
unwrap()	*$(selector).unwrap()*

MÉTODOS	DESCRIPCIÓN
	Elimina el elemento padre de los elementos seleccionados.
val()	*$(selector).val()* *$(selector).val(value)* Establece o devuelve el valor de atributo de los elementos seleccionados de un formulario.
width()	*$(selector).width()* *$(selector).width(value)* Establece o devuelve el ancho de los elementos seleccionados.
wrap()	*$(selector).wrap(wrappingElement,function(index))* Envuelve elementos HTML alrededor de cada elemento seleccionado.
wrapAll()	*$(selector).wrapAll(wrappingElement)* Envuelve elementos HTML alrededor de todos los elementos seleccionados.
wrapInner()	*$(selector).wrapInner(wrappingElement,function(index))* Envuelve elementos HTML alrededor del contenido interior de cada elemento.

JQUERY TRAVERSING

jQuery traversing significa “moverse a través”, es utilizado para encontrar o seleccionar elementos HTML basados en su relación a otros elementos. Se empieza con una selección y se mueve a través de esa selección hasta que alcanzan los elementos deseados.

La familia de abajo ilustra un ejemplo de familia en árbol:



La ilustración explica:

3. El elemento es padre (parent) de <ul>, y un antepadado (ancestor) de todas las cosas dentro de el.
4. El elemento <ul> es padre de ambos elementos <li>, y un hijo (child) de .
5. El elemento <li> de la izquierda es padre de , hijo de <ul> y descendiente (descendant) de .
6. Los dos elementos <li> son hermanos (siblings) (Ellos comparten el mismo padre).
7. El elemento <li> es padre de <b>, hijo de <ul> y un descendiente de .
8. El elemento <b> es un hijo del <li> derecho y descendiente de <ul> y .

ANCESTOR | JQUERY TRAVERSING

Un *ancestor* es un padre, abuelo, bisabuelo y así sucesivamente. Con jQuery se puede ascender a través del árbol del DOM para encontrar antepasados de un elemento.

Tres métodos jQuery útiles para ascender a través del DOM son:

- *parent()*
- *parents()*
- *parentsUntil()*

parent()

Este método devuelve el elemento padre directo del elemento seleccionado. Este método sólo asciendo un nivel en el árbol del DOM.

Ejemplo:

```
$(document).ready(function(){
    $("span").parent();
});
```

parents()

Este método devuelve todos los elementos padres del elemento seleccionado, todo el camino hasta el elemento raíz del documento (<html>).

Ejemplo:

```
$(document).ready(function(){
    $("span").parents();
});
```

Se puede utilizar un parámetro opcional para filtrar la búsqueda de antepasados.

Ejemplo:

```
$(document).ready(function(){
    $("span").parents("ul");
});
```

parentsUntil()

Este método devuelve todos los elementos antepasados entre dos argumentos.

Ejemplo:

```
$(document).ready(function(){
    $("span").parentsUntil("div");
});
```

DESCENDANTS | JQUERY TRAVERSING

Un *descendant* es un hijo, nieto, bisnieto y así sucesivamente. Con jQuery se puede descender a través del árbol del DOM para encontrar descendientes de un elemento.

Dos métodos jQuery útiles para ascender a través del DOM son:

- *children()*
- *find()*

children()

Este método devuelve el hijo directo del elemento seleccionado. Este método sólo desciende un nivel en el árbol del DOM.

Ejemplo:

```
$(document).ready(function(){
    $("div").children();
});
```

Se puede usar un parámetro opcional la búsqueda de hijos. El siguiente ejemplo devuelve todos los elementos <p> cuya clase es “first”, y que son hijos directos de :

Ejemplo:

```
$(document).ready(function(){
    $("div").children("p.first");
});
```

find()

Este método devuelve todos los elementos descendientes del elemento seleccionado, todo el camino hasta el último descendiente.

Ejemplo:

```
$(document).ready(function(){
    $("div").find("span");
});
```

El siguiente ejemplo devuelve todos los descendientes de :

Ejemplo:

```
$(document).ready(function(){
    $("div").find("*");
});
```

SIBLINGS | JQUERY TRAVERSING

Hermanos comparten el mismo padre.

En jQuery se puede recorrer hacia el lado en un árbol del DOM para encontrar elementos hermanos:

- *siblings()*
- *next()*
- *nextAll()*
- *nextUntil()*
- *prev()*

- *prevAll()*
- *prevUntil()*

siblings()

Este método devuelve todos los elementos hermanos del elemento seleccionado.

Ejemplo:

```
$(document).ready(function(){
    $("h2").siblings();
});
```

También se puede utilizar un parámetro opcional para filtrar la búsqueda de hermanos.

Ejemplo:

```
$(document).ready(function(){
    $("h2").siblings("p");
});
```

next()

Este método devuelve el siguiente elemento hermano del elemento seleccionado.

Ejemplo:

```
$(document).ready(function(){
    $("h2").next();
});
```

nextAll()

Este método devuelve todos los siguientes elementos hermanos del elemento seleccionado.

Ejemplo:

```
$(document).ready(function(){
    $("h2").nextAll();
});
```

nextUntil()

Este método devuelve todos los siguientes elementos hermanos entre dos argumentos dados.

Ejemplo:

```
$(document).ready(function(){
    $("h2").nextUntil("h6");
});
```

prev(), prevAll() y prevUntil()

Estos métodos trabajan del mismo modo que los métodos anteriores, pero funcionalmente al revés: ellos devuelven los elementos hermanos anteriores (se mueve hacia atrás a través de los elementos hermanos del DOM, en vez de hacia adelante).

FILTERING | JQUERY TRAVERSING

Los tres métodos de filtro más básicos son:

- *first()*
- *last()*
- *eq()*

Los cuales permitirán seleccionar un elemento específico basado en su posición en un grupo de elementos.

Otros métodos de filtrado, como *filter()* y *not()* permiten seleccionar los elementos que coinciden o no con un determinado criterio.

first()

Este método devuelve el primer método de los elementos seleccionados.

Ejemplo:

```
$(document).ready(function(){
    $("div p").first();
});
```

last()

Este método devuelve el último elemento de los elementos seleccionados.

Ejemplo:

```
$(document).ready(function(){
    $("div p").last();
});
```

eq()

Este método devuelve un elemento con un número específico del índice de los elementos seleccionados.

Ejemplo:

```
$(document).ready(function(){
    $("p").eq(1);
});
```

filter()

Este método permite especificar un criterio. Los elementos que no cumplen el criterio son ignorados en la selección, y se devolverán aquellos que encajan.

Ejemplo:

```
$(document).ready(function(){
    $("p").filter(".intro");
});
```

not()

Este método devuelve todos los elementos que no encajan con el criterio.

Ejemplo:

```
$(document).ready(function(){
    $("p").not(".intro");
});
```

MÉTODOS JQUERY TRAVERSING

La siguiente tabla lista todos los métodos utilizados para *traversing*.

MÉTODOS	DESCRIPCIÓN
add()	*$(selector).add(element,context)* Añade elementos a un conjunto de elementos coincidentes.
addBack()	Añade el conjunto de elementos anterior del conjunto actual.
addSelf()	Obsoleto desde la versión 1.8. Un alias para *addBack*.
children()	*$(selector).children(filter)* Devuelve todos los hijos directos de los elementos seleccionado.
closest()	*$(selector).closest(filter)* Devuelve el primer antepasado del elemento seleccionado.
contents()	*$(selector).contents()* Devuelve todos los hijos directos de los elementos seleccionados, incluyendo texto y comentarios.
each()	*$(selector).each(function(index,element))* Ejecuta cada función para cada elemento emparejado.
end()	Finaliza la operación de filtrado más reciente en la cadena actual, y devuelve el conjunto de elementos que coinciden con su estado anterior.
eq()	*$(selector).eq(index)* Devuelve un elemento con un índice numérico específico de los elementos seleccionados.
filter()	*$(selector).filter(criteria,function(index))* Reduce el conjunto de elementos coincidentes a aquellos que coinciden con el selector o aprueban la prueba de la función.
find()	*$(selector).find(filter)* Devuelve elementos descendientes del elemento seleccionado.
first()	*$(selector).first()* Devuelve el primer elemento de los elementos seleccionados.
has()	*$(selector).has(element)* Devuelve todos los elementos que tienen uno o más elementos dentro de ellos.
is()	*$(selector).is(selectorElement,function(index,element))*

MÉTODOS	DESCRIPCIÓN
	Comprueba el conjunto de elementos coincidentes con selector/elemento/objeto jQuery, y devuelve verdadero si al menos uno de esos elementos coincide con los argumentos proporcionados.
last()	*$(selector).last()* Devuelve el último elemento de los elementos seleccionados.
map()	Pasa cada elemento en el conjunto combinado a través de una función, produciendo un nuevo objeto jQuery que contiene los valores de retorno.
next()	*$(selector).next(filter)* Devuelve el siguiente elemento hermano del elemento seleccionado.
nextAll()	*$(selector).nextAll(filter)* Devuelve todos los siguientes elementos hermanos del elemento seleccionado.
nextUntil()	*selector).nextUntil(stop,filter)* Devuelve todos los siguientes elementos hermanos entre dos argumentos dados.
not()	*$(selector).not(criteria,function(index))* Elimina elementos del conjunto de elementos coincidentes.
offsetParent()	*$(selector).offsetParent()* Devuelve el primer elemento padre posicionado.
parent()	*$(selector).parent(filter)* Devuelve el elemento padre directo del elemento seleccionado.
parents()	*$(selector).parents(filter)* Devuelve todos los elementos antepasados del elemento seleccionado.
parentsUntil()	*$(selector).parentsUntil(stop,filter)* Devuelve todos los elementos antecesores del elemento seleccionado.
prev()	*$(selector).prev(filter)* Devuelve el anterior elemento hermano del elemento seleccionado.
prevAll(()	*$(selector).prevAll(filter)* Devuelve todos los anteriores elementos hermanos del elemento seleccionado.
prevUntil()	*$(selector).prevUntil(stop,filter)* Devuelve todos los anteriores elementos hermanos entre dos argumentos dados.
siblings()	*$(selector).siblings(filter)* Devuelve todos los elementos hermanos del elemento seleccionado.

MÉTODOS	DESCRIPCIÓN
slice()	*$(selector).slice(start,stop)* Reduce el conjunto de elementos coincidentes a un subconjunto especifico por rango de índices.

JQUERY AJAX

AJAX = JavaScript y XML asíncrono.

En resumen, AJAX trata sobre la carga de datos en segundo plano y lo muestra en la web, sin necesidad de recargar la página.

jQuery proporciona varios métodos para las funcionalidades de AJAX. Con los métodos AJAX de jQuery se puede solicitar texto, HTML, XML o JSON desde un servidor remoto utilizando tanto *get* como *post*. Y se pueden cargar directamente los datos externos en elementos HTML seleccionado de un sitio web.

Escribir código AJAX sin jQuery puede ser un poco difícil, porque los diferentes navegadores utilizan diferente sintaxis para la implementación de AJAX. Esto significa que habrá que escribir código extra para probar los diferentes navegadores, sin embargo, jQuery ha tenido esto en cuenta y permite escribir funcionalidades AJAX con una sola línea de código.

METODOS AJAX | JQUERY AJAX

Se estudiarán los métodos AJAX más importantes.

- *load()*
- *get() / post()*

load()

Este método es simple pero muy poderoso. Carga datos desde un servidor y devuelve datos dentro del elementos seleccionado.

Sintaxis:

```
$(selector).load(URL,data,callback);
```

- El parámetro *URL* obligatorio especifica la *url* que se desea cargar.
- El parámetro *data* opcional especifica un conjunto de consulta clave/valor para enviar junto con la solicitud.
- El parámetro *callback* opcional es el nombre de una función para ser ejecutada después de que el método *load()* se haya completado.

Ejemplo:

El contenido de “demo_test.txt” es:

```
<h2>jQuery and AJAX is FUN!!!</h2>
<p id="p1">This is some text in a paragraph.</p>
```

El siguiente ejemplo devuelve el contenido de “demo_test.txt” dentro de un específico.

```
$("#div1").load("demo_test.txt");
```

También es posible añadir un selector jQuery al parámetro *URL*.

Ejemplo:

El siguiente ejemplo carga el contenido del elemento con id=”p1”, dentro del fichero “demo_test.txt”.

```
$("#div1").load("demo_test.txt #p1");
```

El parámetro opcional *callback* especifica una función callback que se ejecuta cuando el método *load()* se ha completado. La función callback puede tener diferentes parámetros:

- *responseTxt*: contiene el contenido resultante si la llamada se hace correctamente.
- *statusTxt*: contiene el estado de la llamada.
- *xhr*: contiene el objeto XMLHttpRequest.

Ejemplo:

```
$("button").click(function(){
    $("#div1").load("demo_test.txt", function(responseTxt,
statusTxt, xhr){
        if(statusTxt == "success")
            alert("External content loaded successfully!");
        if(statusTxt == "error")
            alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
})
```

HTTP Request: GET vs POST

Dos métodos comúnmente utilizados para solicitar una respuesta entre cliente-servidor son: GET y POST.

- GET: Solicita datos de un recurso específico.
- POST: Subministra datos para ser procesados por un recurso específico.

GET es utilizado básicamente para conseguir (recuperar) datos del servidor. El método GET puede devolver datos en caché. POST también puede ser utilizado para conseguir algunos datos del servidor. Sin embargo, el método POST nunca almacena datos en caché, y normalmente se utiliza para enviar datos junto con la solicitud.

$.get()

Este método solicita datos del servidor con una petición HTTP GET.

Sintaxis:

```
$.get(URL,callback);
```

- El parámetro *URL* obligatorio especifica la *url* a la que se desea solicitar.
- El parámetro *callback* opcional es el nombre de una función para ser ejecutada si la solicitud tiene éxito.

Ejemplo:

```
$("button").click(function(){
    $.get("demo_test.asp", function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```

"demo_test.asp":

```
<%
response.write("This is some text from an external ASP file.")
%>
```

$.post()

Este método solicita datos del servidor con una petición HTTP POST.

Sintaxis:

```
$.post(URL,data,callback);
```

- El parámetro *URL* obligatorio especifica la *url* a la que se desea solicitar.
- El parámetro *data* especifica algún dato a enviar junto con la solicitud.
- El parámetro *callback* opcional es el nombre de una función para ser ejecutada si la solicitud tiene éxito.

Ejemplo:

```
$("button").click(function(){
    $.post("demo_test_post.asp",
    {
        name: "Donald Duck",
        city: "Duckburg"
    },
    function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```

Archivo “demo_test_post.asp”:

```
<%
dim fname,city
fname=Request.Form("name")
city=Request.Form("city")
Response.Write("Dear " & fname & ". ")
Response.Write("Hope you live well in " & city & ".")
%>
```

MÉTODOS JQUERY AJAX

AJAX es el arte de intercambiar datos con un servidor, y actualizar partes de una web, sin recargar la página completa. La siguiente tabla lista todos los métodos utilizados para AJAX.

MÉTODOS	DESCRIPCIÓN
$.ajax()	*$.ajax({name:value, name:value, ... })* Realiza una petición asíncrona AJAX.
$.ajaxPrefilter()	Opciones AJAX personalizables manualmente o edición de opciones existentes antes de que cada petición sea enviada y procesada por $.ajax().
$.ajaxSetup()	*$.ajaxSetup({name:value, name:value, ... })* Establece los valores por defecto para las futuras peticiones AJAX.
$.ajaxTransport()	Crea un objeto que se encarga de la transmisión real de datos AJAX.
$.get()	*$.get("test.php");* Carga datos desde un servidor usando el método de petición GET.
$.getJSON()	*$(selector).getJSON(url,data,success(data,status,xhr))* Carga datos codificados en JSON desde un servidor usando el método de petición GET.
$.getScript()	*$(selector).getScript(url,success(response,status))* Carga (y ejecuta) un JavaScript desde el un servidor usando el método de petición GET.
$.param()	*$.param(object,trad)* Crea una representación serializada de un array u objeto.
$.post()	*$(selector).post(URL,data,function(data,status,xhr),dataType)* Carga datos desde un servidor usado petición el método de petición POST.
ajaxComplete()	*$(document).ajaxComplete(function(event,xhr,options))* Especifica una función a ser ejecutada cuando la petición AJAX ha sido completada.
ajaxError()	*$(document).ajaxError(function(event,xhr,options,exc))* Especifica una función a ser ejecutada cuando la petición AJAX ha sido completada con un error.
ajaxSend()	*$(document).ajaxSend(function(event,xhr,options))* Especifica una función a ser ejecutada antes de enviar una petición AJAX.
ajaxStart()	*$(document).ajaxStart(function())* Especifica una función a ser ejecutada cuando la primera petición AJAX empieza.
ajaxStop()	*$(document).ajaxStop(function())*

MÉTODOS	DESCRIPCIÓN
	Especifica una función a ser ejecutada cuando todas las peticiones AJAX se han completado.
ajaxSuccess()	*$(document).ajaxSuccess(function(event,xhr,options))* Especifica una función a ser ejecutada cuando todas las peticiones AJAX han sido completadas satisfactoriamente.
load()	*$(selector).load(url,data,function(response,status,xhr))* Carga datos desde un servidor y pone los datos devueltos en un elemento seleccionado.
serialize()	*$(selector).serialize()* Codifica un conjunto de elementos de un formulario como una cadena.
serializeArray()	*$(selector).serializeArray()* Codifica un conjunto de elementos de un formulario como un array de nombres y valores.

OTRAS FUNCIONALIDADES DE JQUERY

Si se desea utilizar otros frameworks en un sitio web, mientras todavía se utiliza jQuery.

JQUERY Y OTROS FRAMEWORKS JAVASCRIPT | JQUERY MISC

jQuery utiliza el signo $ como acceso directo para jQuery.

Hay muchos frameworks JavaScript como: Angular, Backbone, Ember, Knockout, etc. Si dos frameworks diferentes utilizan el mismo acceso directo, uno de ellos podría parar de funcionar. jQuery ha pensado en esto y hay implementado el método *noConflict().*

noConflict()

Este método libera el mantenimiento del identificador de acceso directo $ y por lo tanto otros scripts pueden utilizarlo.

De este modo se puede seguir utilizando jQuery, simplemente escribiendo la palabra completa *jQuery* en vez del acceso directo.

Ejemplo:

```
$.noConflict();
jQuery(document).ready(function(){
    jQuery("button").click(function(){
        jQuery("p").text("jQuery is still working!");
    });
});
```

También podemos crear nuestro propio acceso directo de manera muy sencilla. El método *noConflict()* devuelve una referencia a jQuery, que se puede guardar en una variable para usarla posteriormente.

Ejemplo:

```
var jq = $.noConflict();
jq(document).ready(function(){
    jq("button").click(function(){
        jq("p").text("jQuery is still working!");
    });
});
```

Si tiene un bloque de código jQuery el cual usa el acceso directo *$* y no quieres cambiarlo todo, se puede pasar el signo *$* como un parámetro del método *ready*. Esto permite acceder a jQuery usando *$*, dentro de esta función. Fuera de ella se puede utilizar "*jQuery*".

Ejemplo:

```
$.noConflict();
jQuery(document).ready(function($){
    $("button").click(function(){
        $("p").text("jQuery is still working!");
    });
});
```

MÉTODOS JQUERY VARIOS

A continuación, se listan varios métodos jQuery:

MÉTODOS	DESCRIPCIÓN
data()	*$(selector).data(name,value)* Agrega y obtiene datos de los elementos seleccionados. - El parámetro opcional *name* especifica el nombre de los datos a recuperar (sólo para obtener). - El parámetro *value* especifica el valor de datos a establecer (en el caso de agregar tanto *name* como *value* son obligatorios)
each()	*$(selector).each(function(index,element))* Especifica una función a ejecutar para cada elemento seleccionado. - La función con los dos parámetros es obligatoria y se ejecuta para cada elemento que cumpla la condición. (*Index* es la posición del selector. *Element es el elemento actual*)
get()	*$(selector).get(index)* Consigue el elemento DOM especificado por el selector. - El parámetro opcional *index* específica cuál de los elementos obtener por el índice.
index()	*$(selector).index(element)* Devuelve el índice de la posición de elementos específicos relativos así mismos u otros elementos seleccionados. Si el elemento no es encontrado devuelve -1. - El parámetro opcional *element* especifica el elemento a conseguir la posición.
$.noConflict()	*$.noConflict(removeAll)* Libera el control de jQuery desde la variable $. También se puede utilizar para especificar un nombre personalizado para la variable de jQuery. - El parámetro opcional *removeAll* es un valor booleano que especifica si o no liberar el control de todas las variables jQuery (incluyendo “jQuery”).
$.param()	*$.param(object,trad)* Crea una representación serializada de un array o un objeto. Los valores serializados pueden ser utilizados en la cadena de consulta URL cuando se hace petición AJAX. - El parámetro obligatorio *object* especifica un array u objeto a serializar. - El parámetro opcional *trad* es un valor booleano que especifica si o no utilizar el habitual estilo de parámetro de serialización.
removeData()	*$(selector).removeData(name)* Elimina datos previamente establecidos con el método *data()*. - El parámetro opcional *name* especifica el nombre de datos a eliminar. Si no es especificado elimina todos los datos almacenados de los elementos seleccionados.
size()	*$(selector).size()*

MÉTODOS	DESCRIPCIÓN
	Devuelve el número de elementos que encajan con el selector de jQuery. Este método está obsoleto, se puede utilizar el método *lenght* en su lugar.
toArray()	*$(selector).toArray()* Devuelve los elementos que coinciden con el selector envueltos en un array.

REFERENCIAS

- **jQuery**: John Resig. “API Documentation”. Disponible en la web https://jquery.com/
- **w3school**: “jQuery Tutorial”. Disponible en la web http://www.w3schools.com/jquery/default.asp
- **LibrosWeb**: Rebecca Murphey. “Fundamentos de jQuery”.Disponible en la web https://librosweb.es/libro/fundamentos_jquery/
- **DesarrolloWeb**: Miguel Ángel Álvarez. “Manual de jQuery”. Disponible en la web http://www.desarrolloweb.com/manuales/manual-jquery.html

www.guede.me