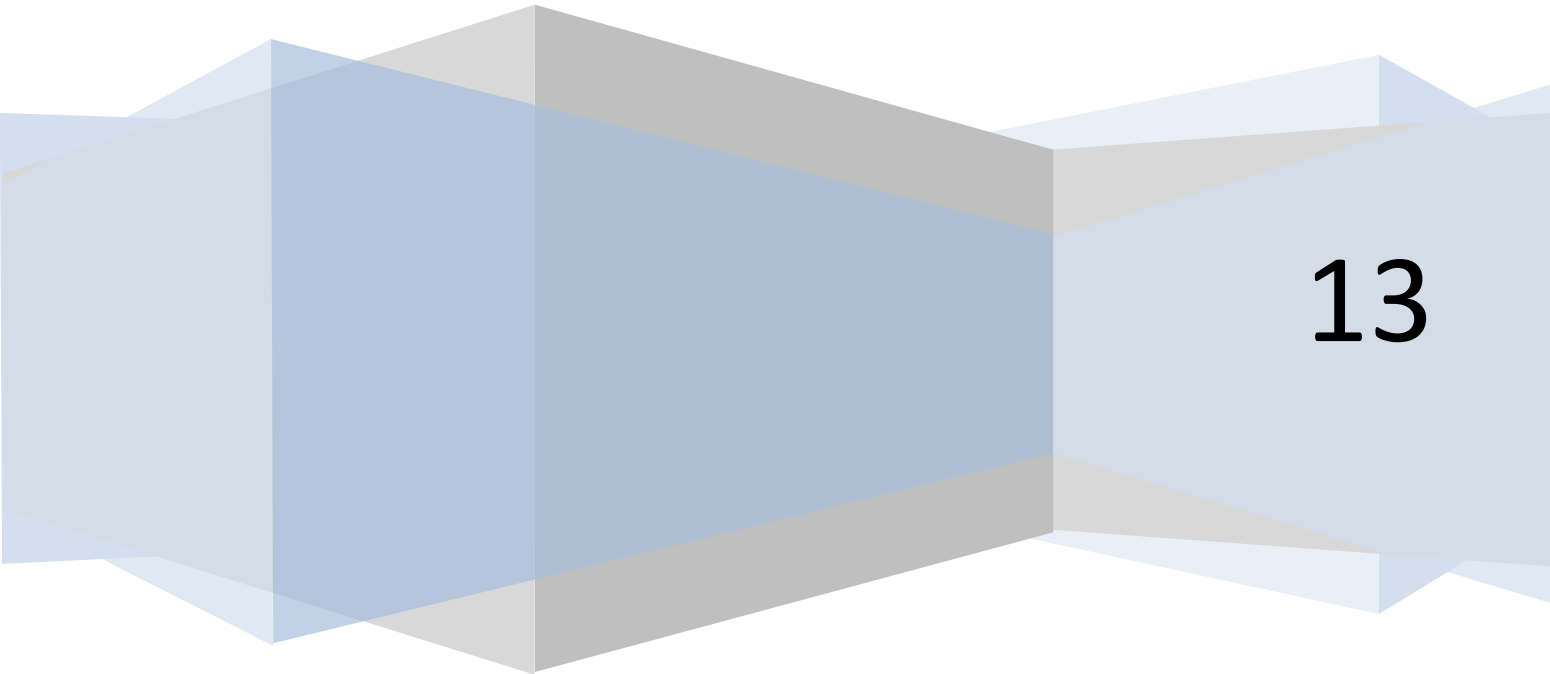


Manual básico de jQuery

http://mundosica.github.io/tutorial_hispano_jQuery/

Editado por José Luis Comesaña



13

ÍNDICE

Introducción a jQuery.....	3	Efectos y modificaciones sobre el DOM.....	15
Descripción de este manual.....	3	insertbefore.....	15
Breve referencia HTML.....	3	before.....	15
Concepto DOM.....	3	insertafter.....	15
Breve referencia CSS.....	4	after.....	15
Anatomía CSS.....	4	append.....	15
Software a usar.....	5	appendto.....	15
1. Editor de texto:.....	6	fadeln.....	15
2. Navegador web:.....	6	Animate.....	15
3. La librería jQuery.....	7	animate sobre un color.....	16
Evolución de jQuery.....	7	Código HTML.....	16
Cambios entre 1.7 y 1.8.....	7	Código CSS.....	16
Cambios en la futura versión 2.0.....	8	Código jQuery.....	16
Agregando jQuery.....	8	show.....	16
4. Servidor web.....	8	hide.....	16
Selectores con jQuery.....	8	Lecturas recomendadas.....	16
Transversatilidad.....	8	Ajax.....	16
Enlaces externos recomendables:.....	9	Ajax y JQuery.....	17
Características principales de jQuery.....	9	Ejemplo básico.....	17
constructor jQuery.....	9	Quitando la opción asincrónica.....	17
\$ es una alias de jQuery.....	9	Ejemplo Manejando el error.....	17
Ejemplo - jQuery Alias.....	9	Otras funciones Ajax.....	18
Código HTML.....	9	getScript.....	18
Código jQuery.....	9	Ejemplo - getScript.....	18
Trabaja por grupos(<i>Iteración implícita</i>).....	10	Código JS.....	18
Ejemplo - Comportamiento trabajo por grupo.....	10	Código CSS.....	18
Código HTML.....	10	Código HTML.....	18
Código jQuery.....	10	getJSON.....	18
Diseñado para realizar consultas a través del DOM.....	11	Ejemplo - getJson y la API de Flickr.....	19
Es un sistema modular.....	11	Código HTML.....	19
Chaining.....	11	Código CSS:.....	19
Enlaces externos.....	11	Código jQuery.....	19
Manejo de eventos.....	11	Ejemplo load.....	19
Introducción a los eventos.....	11	Lecturas recomendadas.....	20
Evento click.....	12	Organización de código.....	20
Ejemplo - Click.....	12	Organizando el código en archivos.....	20
Código HTML.....	12	Ejemplo - tabla de contenidos.....	20
Código JS.....	12	JS.....	20
Comportamiento en cola.....	12	jQuery Plugins.....	23
Evento hover.....	12	Uso.....	23
Ejemplo - Hover.....	13	Ejemplo plugin jquery.....	24
código HTML:.....	13	Código JS.....	24
CSS.....	13	Uso.....	24
código JS:.....	13	Anexo I - Etiquetas en la cabecera.....	25
Eventos del teclado.....	13	Inserción de código javascript.....	25
focusout.....	13	Inserción de código CSS.....	25
keypress.....	13	Tipo de caracteres.....	25
keyup.....	13	Descripción de la página.....	25
keydown.....	13	Title.....	25
Ejemplo de eventos de teclado:.....	13	Descripción (Description).....	25
Ejemplo - Evento Teclado.....	14	Robots.....	25
código HTML:.....	14	Parablas clave Keywords:.....	26
código CSS:.....	14	Enlaces recomendados:.....	26
Código jQuery:.....	14		
Borrando eventos función off.....	14		
Enlaces recomendados.....	15		

Introducción a jQuery

Descripción de este manual

Este es un pequeño manual *básico de jQuery*, el cual consta de 5 sesiones, cada una diseñada para una clase de 2 horas.

El propósito de este manual es ser una ayuda en la comprensión de los conceptos básicos de esta librería, al concluir la lectura de este manual se espera que el lector sea capaz: de comprender casi cualquier código en **jQuery** y continuar su aprendizaje por su cuenta.

Breve referencia HTML

HTML, siglas de *HyperText Markup Language* («lenguaje de marcado de hipertexto»), hace referencia al [lenguaje de marcado](#) para la elaboración de páginas web. Se utiliza para describir la estructura y el contenido en forma de texto. El **HTML** se escribe en forma de «etiquetas», rodeadas por *corchetes angulares*(<,>). **HTML** también puede describir, hasta un cierto punto, la apariencia de un documento aunque resulta una practica más recomendable incluir hojas de estilos(**CSS**) para este propósito, también suele incluir scripts (como [JavaScript](#)), el cual puede afectar el comportamiento de navegadores web. *fuentes:* <http://es.wikipedia.org/wiki/HTML>

Un documento **Html** tiene mas o menos el siguiente aspecto:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Titulo</title>
  </head>
  <body>
    <h1>Documento de ejemplo</h1>
    <div>
      <h1>Contenido dentro del div</h1>
      <h2>sub título</h2>
      <p>parrafo 1</p>
      <p>parrafo 2</p>
    </div>
  </body>
</html>
```

Las primeras dos líneas describen el tipo de archivo en este caso es **HTML 5** y la lengua a utilizar en este caso español.

```
<!DOCTYPE html>
<html lang="es">
```

A lo largo de este manual se usara **HTML 5** al menos que se diga lo contrario.

Posteriormente viene el head, el cual en este caso contiene solo el titulo del documento, pero aquí se suelen poner tanto información de indexación como solicitudes de archivos necesarios como hojas de estilos y funcionalidades en javascript.

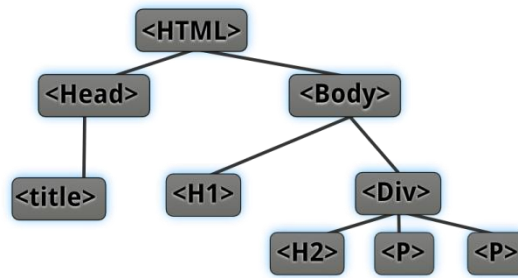
```
<head>
  <title>Titulo</title>
</head>
```

Sí desea conocer un poco más sobre etiquetas validas puedes ver el siguiente ejemplo:
[Etiquetas_head.html](#).

Concepto DOM

El **Modelo en Objetos para la Representación de Documentos** ó **DOM** (*Document Object Model*), es un estándar de objetos para representar documentos **HTML** y **XML**. A través de él, los programas pueden acceder y modificar el contenido, estructura y estilo.

La siguiente imagen es una **representación** del **DOM** del documento visto anteriormente:



La imagen anterior muestra el documento **Html** en forma de un árbol. Esta representación es conocida como un árbol **DOM**.

El responsable del DOM es el World Wide Web Consortium ([W3C](http://www.w3.org/)).

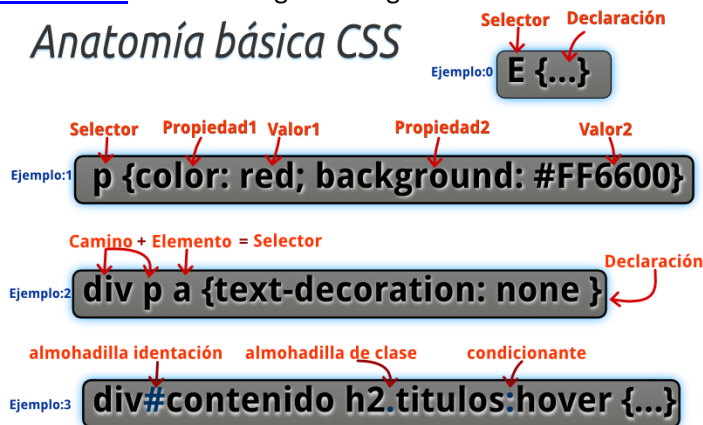
Breve referencia CSS

Cascading Style Sheets (hojas de estilo en cascada). CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

El responsable del CSS es también el World Wide Web Consortium ([W3C](http://www.w3.org/)).

Anatomía CSS

Con **CSS** podemos hacer declaraciones de estilo sobre los elementos **HTML**, para esto el [W3C](http://www.w3.org/) ha [definido una serie de selectores](#) los cuales siguen el siguiente formato:



jQuery hace uso de estos selectores para interactuar con el **DOM**, por esta razón es importante que el lector comprenda **CSS** lamentablemente el alcance de este manual no contempla la explicación detallada de **CSS**.

Para nuestros fines basta con que conozcamos bien el uso de selectores **CSS**:

Lista de Selectores CSS

Selector	Significado	Versión CSS
*	Cualquier elemento	2
E	Elementos con etiqueta E	1
E[foo]	Elementos E con foo como atributo	2
E[foo="bar"]	Elementos E con atributo foo con valor bar	2
E[foo~="bar"]	an E element whose "foo" attribute value is a list of whitespace-separated values, one of which is exactly equal to "bar"	2
E[foo^="bar"]	Elementos E con atributo foo cuyo valor inicia en bar	3

E[foo\$="bar"]	Elementos E con atributo foo cuyo valor termina en bar	3
E[foo*="bar"]	Elementos E con atributo foo cuyo valor contiene como subcadena a bar	3
E[foo ="en"]	an E element whose "foo" attribute has a hyphen-separated list of values beginning (from the left) with "en"	2
E:root	Elementos E , donde E es un elemento raíz del documento	3
E:nth-child(n)	Elementos enésimo E	3
E:nth-last-child(n)	Elementos enésimo E contando del último elemento al primero	3
E:nth-of-type(n)	an E element, the n-th sibling of its type	3
E:nth-last-of-type(n)	an E element, the n-th sibling of its type, counting from the last one	3
E:first-child	an E element, first child of its parent	2
E:last-child	an E element, last child of its parent	3
E:first-of-type	an E element, first sibling of its type	3
E:last-of-type	an E element, last sibling of its type	3
E:only-child	an E element, only child of its parent	3
E:only-of-type	an E element, only sibling of its type	3
E:empty	an E element that has no children (including text nodes)	3
E:link E:visited	an E element being the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited)	1
E:active E:hover E:focus	an E element during certain user actions	1 y 2
E:target	an E element being the target of the referring URI	3
E:lang(fr)	an element of type E in language "fr" (the document language specifies how language is determined)	2
E:enabled E:disabled	a user interface element E which is enabled or disabled	3
E:checked	a user interface element E which is checked (for instance a radio-button or checkbox)	3
E::first-line	the first formatted line of an E element	1
E::first-letter	E::first-letter	1
E::before	generated content before an E element	2
E::after	generated content after an E element	2
E.warning	an E element whose class is "warning" (the document language specifies how class is determined).	1
E#myid	ID selectors	1
E:not(s)	an E element that does not match simple selector s	3
E F	an F element descendant of an E element	1
E > F	an F element child of an E element	2
E + F	an F element immediately preceded by an E element	2
E ~ F	an F element preceded by an E element	3
Fuente:	http://www.w3.org/TR/css3-selectors/#selectors	

Software a usar

Para desarrollar sitios de Internet con **jQuery** básicamente necesitamos 4 cosas:

1. Editor de texto
2. Navegador web.

3. La librería jQuery.
4. Servidor web (para la sección ajax, json).

1. Editor de texto:

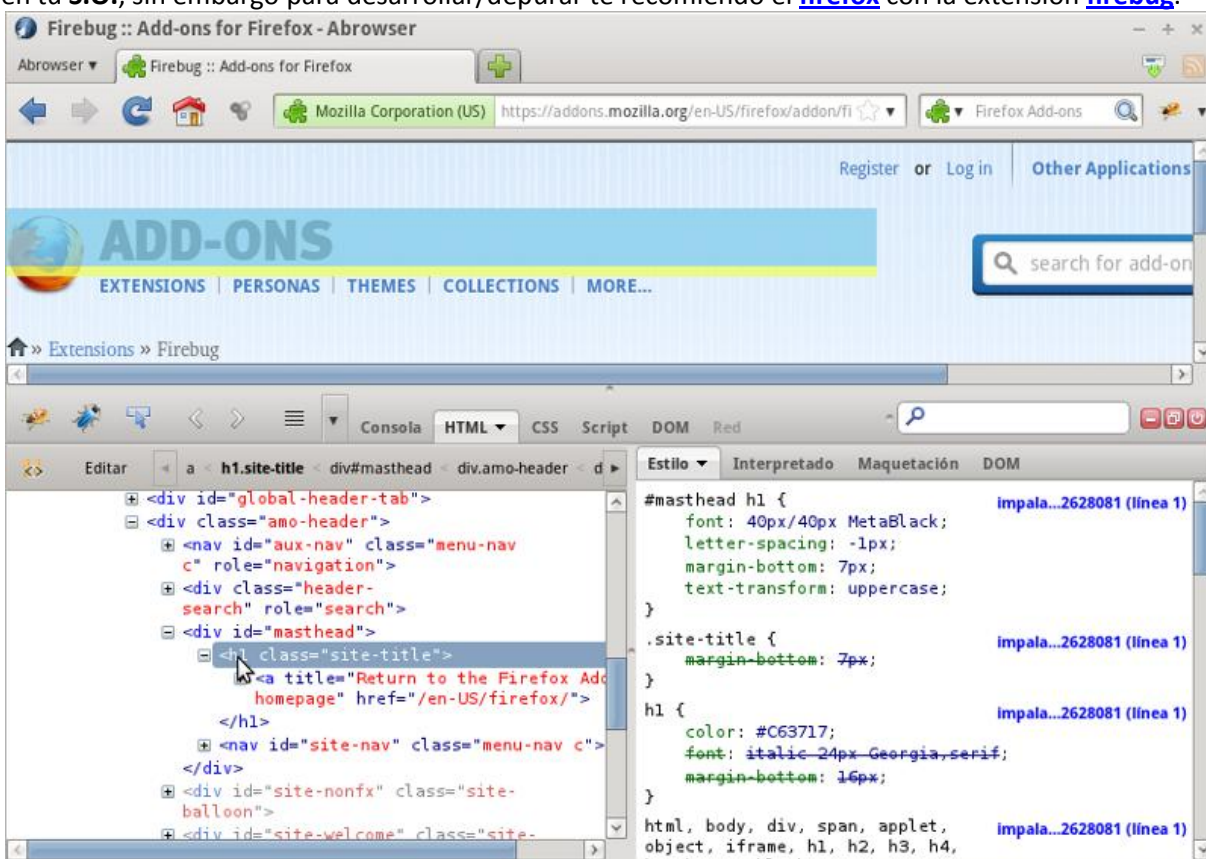
El editor de texto es la herramienta básica para editar archivos: **HTML**, **JavaScript** y **CSS**. Hay muchos editores muy buenos y usted puede usar el editor de su agrado, entre los que recomendamos son los siguientes (*ordenados según mis preferencias*):

- ✓ [Geany](#)
- ✓ [Komodo EDIT](#)
- ✓ [Vim](#)
- ✓ [Emacs](#)
- ✓ [Kate](#)
- ✓ [Notepad++](#)
- ✓ [TextMate](#)
- ✓ [sublimeText](#)

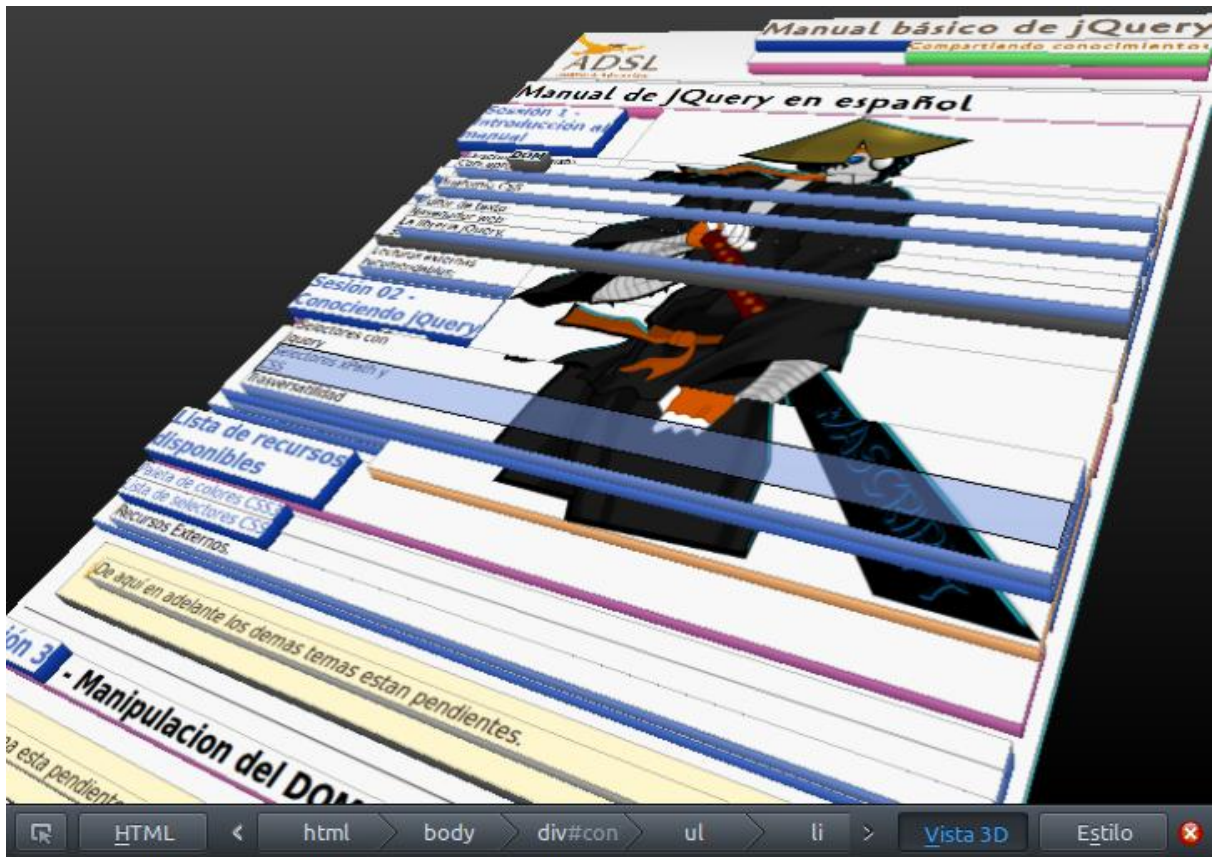
Te recomiendo el [geany](#) el cual está disponible para múltiples Sistema operativos, además de agregarle el [geany-web-ninja](#) el cual es una colección de archivos de configuración orientadas al desarrollo web.

2. Navegador web:

Por cuestiones de pruebas te recomiendo que tengas todos los navegadores que estén disponibles en tu **S.O.**, sin embargo para desarrollar/depurar te recomiendo el [firefox](#) con la extensión [firebug](#).



Otra de las ventajas que tiene es que a partir de la [versión 11](#) de **Mozilla Firefox**, trae integrado el **3D inspector** que nos permite analizar la profundidad de las capas **CSS** desde una forma gráfica, como podemos ver a continuación:



Otra razón para usar firefox es que es un proyecto de código libre y abierto, distribuyéndose bajo [triple licencia](#):

- ✓ Licencia Pública de Mozilla (MPL)
- ✓ Licencia pública general de GNU (GPL)
- ✓ Licencia pública general reducida de GNU (LGPL).

3. La librería jQuery.

jQuery: biblioteca de *JavaScript*, fue creada por [John Resig](#), permite simplificar la manera de interactuar con los documentos **HTML**, manipular el árbol **DOM**, manejo de eventos, desarrollar animaciones y agregar interacción con la técnica **AJAX** en páginas web.

Evolución de jQuery

jQuery es una librería en constante cambio, es recomendable usar la [librería estable más reciente](#), ya que en cada nueva versión se integran características nuevas al tiempo que se mejoran las anteriores, la imagen de la derecha es una comparativa en el rendimiento de **jQuery 1.6.4** vs **jQuery 1.7** en distintos navegadores:

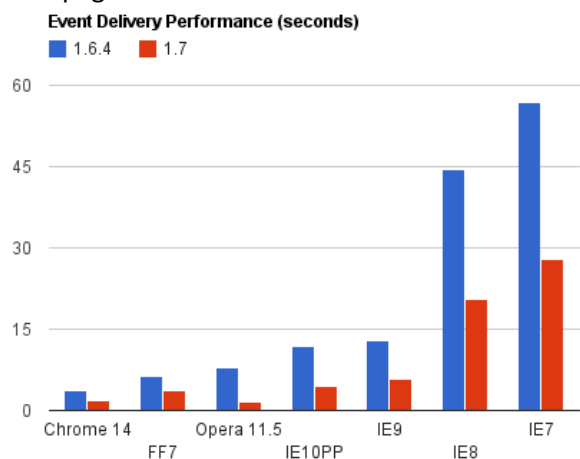
Fuente:

<http://blog.jquery.com/2011/11/03/jquery-1-7-released/>

Cambios entre 1.7 y 1.8

Quando **jQuery** migro a la versión **1.8** se hicieron algunos cambios importantes, por ejemplo la versión **1.8** es más pequeña que **1.7**, las funciones `css()` y `animate()` libres de prefijos (-webkit, -moz, -ms, -o), se repararon 160 errores, mayor flexibilidad en animaciones y se mejoraron las consultas.

Fuente: <http://blog.jquery.com/2012/08/09/jquery-1-8-released/>



Cambios en la futura versión 2.0

En estos momentos la librería **jQuery** está evolucionando a la versión 2.0,

Fuente: <http://blog.jquery.com/2013/04/09/jquery-2-0-beta-3-released/>

Agregando jQuery

Una vez que tenemos la librería **jQuery**, lo siguiente es incluir la librería en nuestros documentos **HTML**, esto lo hacemos, agregando el script dentro de la cabecera del documento.

```
<head>
  <!-- agregando libreria jQuery -->
  <script type='text/javascript' src='js/jquery1.9.js'></script>
</head>
```

Por otra parte también puedes agregar la librería directamente del servidor de **google API**:

<https://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js>

4. Servidor web.

Un servidor web es un equipo que esta en la escucha del puerto http(80), para nuestros fines nos es suficiente con instalar **apache web server**, sin embargo cuando se habla de servidor web casi siempre se incluye **apache**, **php** y **mysql**, si deseas tener todos estos paquetes los cuales te serán útiles a futuro te recomiendo instalar un **xampp** el cual está disponible para:

- ✓ GNU/Linux
- ✓ Mac®
- ✓ Windows®

Selectores con jQuery

La propiedad más significativa de **jQuery** es su flexibilidad para hacer consultas a través del **DOM** (de ahí el nombre).

jQuery básicamente nos permite hacer 3 tipos de consultas:

- ✓ [Consultas **CSS**]
- ✓ [Consultas XPath](#)
- ✓ [Consultas **Trasversales**]

Este manual explica las consultas **CSS** y algunos métodos **trasversales**, dejando un poco de lado las consultas **xPath** ya que casi no se usan, pero es bueno que el lector sepa que existen y que también son una forma de acceder a los elementos.

Transversalidad

Se ha explicado cómo funciona [la anatomía CSS](#), falta explicar la **transversalidad**, la cual [es un conjunto de métodos definidos por jQuery](#) para refinamiento de selectores, para explicar esto ocuparemos 2 funciones.

- ✓ **find**: Nos permite realizar una consulta sobre un objeto jQuery.
- ✓ **filter**: Nos permite realizar un filtrado, descartando elementos, sobre un objeto jQuery.

Imaginemos que tenemos una *tabla* que tiene como atributo **id** el valor **idTabla**, en el cual queremos acceder a todos sus elementos **th**. Una forma de hacerlo sería:

```
$("#idTabla th").css('background', '#ddd');
```

Otra alternativa es usar el método **find** para buscar en un objeto de **jQuery** el cual me representa una sub-rama del DOM en **idTabla**.

```
$("#idTabla").find('th').css('background', '#ddd');
```

Si existe una alternativa mejor a otra ¿cual crees que sea la mejor?, ¿por qué?.

Enlaces externos recomendables:

- ✓ [Libro CSS Avanzado](#)
- ✓ [Secuencias de caracteres de escape CSS](#)
- ✓ [Como maneja CSS los errores.](#)
- ✓ [Sitio oficial jQuery](#)
- ✓ [Foro de jQuery](#)
- ✓ [Documentación jQuery](#)
- ✓ [IRC de freeNode canal jQuery](#)

Características principales de jQuery.**Técnicamente ¿que es jQuery?.**

jQuery es un objeto define un conjunto funciones(ó métodos) en él.

constructor jQuery

El constructor es una función sobrecargada que:

- ✓ Si recibe un `string` esta la toma como una query(consulta) y devuelve un conjunto de elementos del **DOM** correspondientes a la consulta.
- ✓ En caso de no recibir nada devuelve un conjunto vacío.

En ambos casos el sub conjunto de funciones en jQuery está disponibles, por ejemplo todas las siguientes instrucciones son validas:

```
//construccion con consulta valida o invalida
jQuery('h2').css('color', 'red');
jQuery('esto no existe').css('color', 'red');

//construccion sin consulta
jQuery.css('color', 'red');
```

\$ es una alias de jQuery.

Es más común ver que en los ejemplos hacen referencia a la llamada de la función `$` esta es un alias (un sobrenombre) de **jQuery**. Los ejemplos anteriores usando el alias seria:

```
//construccion con consulta valida o invalida
$('h2').css('color', 'red');
$('esto no existe').css('color', 'red');

//construccion sin consulta
$.css('color', 'red');
```

Ejemplo - jQuery Alias**Código HTML.**

```
<a class="button alias" href="#">Acción con $</a>
<a class="button jquery" href="#">Acción con jQuery</a>
```

Código jQuery

```
$h2 = $('h2');
//Ejecución del evento hover sobre el alias $
$('a.alias').hover(function(event) {
    $h2.css('color', 'red');
}, function(event) {
    $h2.css('color', 'black');
});

//Ejecución del evento hover sobre jQuery
jQuery('a.jquery').hover(function(event) {
    $h2.css('color', 'red');
}, function(event) {
    $h2.css('color', 'black');
});
```

Esto no siempre es válido, cuando se utilizan otras librerías que usan la función `$`, para esto se emplea el `noConflict` de jQuery.

Trabaja por grupos (Iteración implícita).

Los métodos de **jQuery** que realizan consultas trabajan con iteración implícita, por ejemplo, la consulta.

```
$('#h2')
```

Nos devuelve el conjunto de los títulos nivel 2 (`h2`), ahora si por ejemplo quisiéramos cambiarle el color a un gris oscuro (`#333`), esto lo podríamos hacer de la siguiente manera.

```
$.each($('#h2'), function() {
    $(this).css("color", "#333");
});
```

En este caso estamos ocupando el método `each`, en el cual para cada título en la selección se ejecuta una función la cual le cambia el color.

De momento no se preocupe en comprender el `each`. **Lo importante** es que comprenda que estamos iterando explícitamente para realizar una determinada acción sobre cada elemento en el conjunto seleccionado.

Existe una forma de hacer esto más "fácil" y es ocupando el comportamiento de iteración implícita de **jQuery** (o acción sobre el conjunto), en este caso la función `css` puede trabajar sobre toda una selección:

```
$('#h2').css('color', '#333');
```

Ejemplo - Comportamiento trabajo por grupo

Código HTML.

```
<a class="button each" href="#">Acción con </a>
<a class="button grupo" href="#">Acción con jQuery</a>
```

Código jQuery

```
$h2 = $('#h2');
// Cambiando el color de todos los titulos desde un ciclo each
$('#a.each').click( function(event){
    $.each($h2, function() {
        $(this).css("color", "red");
    });
    event.preventDefault();
});
// Cambiando el color de todos los titulos desde el comportamiento en grupo
$('#a.grupo').click( function(event){
    $h2.css("color", "blue");
    event.preventDefault();
});
```

En general las funciones en **jQuery** son de la siguiente forma:

```
//ejemplo 1
$('#selector').funcionNombre(param1,param2,...);
//ejemplo 2
$('#selector').find('selector 2').funcionNombre(param1,param2,...);
```

Y trabajan sobre toda la selección (*conjunto*), es decir si la selección es todo un grupo la función se ejecutara sobre todos los elementos.

A estas funciones las llamaremos `jQuery.fn.funcionNombre` o bien su forma reducida `$.fn.funcionNombre` ya que pertenecen al espacio de nombre `fn` de **jQuery** que se hacen a un

prototipo de **jQuery** del es decir de una extensión de **jQuery**, `jQuery.fn` es también un alias de [jQuery.prototype](#).

Diseñado para realizar consultas a través del DOM.

Quizás la principal característica de **jQuery** es que está diseñado para realizar consultas a través del **DOM**, como ya se menciono antes, **jQuery** nos permite realizar consultas **CSS**, **xPath** y **trasversales**.

```
$( 'h2 + p' ).css( 'color', '#946900' );
$( 'h2 + p' ).css( 'font-style', 'italic' );
```

Con esto indicamos que todo párrafo (`p`) inmediatamente seguido (+) de un titulo nivel 2 (`h2`), va a tomar el color de la letra a café y el estilo de la fuente será del tipo itálica.

No abuse de esta característica, la tarea de realizar una consulta a través del **DOM** puede llegar a ser algo bastante complejo (*dependiendo de la consulta*).

Una mejor opción es almacenar la consulta realizada en una variable:

```
$parrafosBajoTitulo = $( 'h2 + p' );
$parrafosBajoTitulo.css( 'color', '#946900' );
$parrafosBajoTitulo.css( 'font-style', 'italic' );
```

Es buena práctica agregar el prefijo `$` a las variables que almacenan el resultado de una consulta.

Es un sistema modular.

- ✓ [jQuery plugins](#)
- ✓ [jQuery UI](#)
- ✓ [jQuery mobile](#)

Chaining.

Atienda al siguiente código:

```
$( 'h2' ).css( 'color', 'red' );
$( 'h2' ).text( 'Cambiando el contenido a todos los elementos h2' );
```

Ahora compárelo con el siguiente código:

```
$( 'h2' )
    .css( 'color', 'red' )
    .text( 'Cambiando el contenido a todos los elementos h2' );
```

¿Cuál cree que sea mejor?, argumente su respuesta.

Enlaces externos.

- ✓ [Atributos personalizados](#)

Manejo de eventos

Introducción a los eventos.

jQuery define una lista de eventos y funciones para la administración de los mismos, la sintaxis por defecto para un manejador de evento es de la siguiente forma `$.fn.nombreEvento`.

```
$( 'Selector' ).nombreEvento( function( event ) {
    //funcion que administra el evento.
    //this es el elemento que disparo el evento.
} )
```

Aquí es importante resaltar que `this` contendrá la instancia del elemento que disparo el evento, por ejemplos si a los enlaces (`a`) le agregamos el evento `click`, `this` contendrá la instancia del enlace específico sobre del cual hayamos hecho click, analice el siguiente código.

jQuery define varios eventos para agregar un manejador de dicho evento basta con agregar una función en la llamada de dicho evento esta función puede recibir un argumento `event` el cual es útil para obtener información de dicho evento o para cambiar el comportamiento del mismo.

Evento click

El evento `click` es disparado cuando con el mouse le damos **click** sobre un elemento seleccionado, la forma que tiene este evento es la siguiente:

```
$("#a.button").click(
  function(event) {
    console.log('Manejador para el evento click del enlace(a) con clase(.) button');
    event.preventDefault();
  });
```

En el ejemplo anterior vemos como al evento `click` le agregamos una función como manejador de dicho evento, esta recibe el objeto `event` el cual ocupamos para ejecutar el método `preventDefault` este método detiene el comportamiento por defecto de dicho evento, en este caso tenemos un `link(a)` que al darle click lo común sería que siguiera en enlace definido en el atributo `href`.

Ejemplo - Click

Código HTML

```
<a class='testClick' href='http://adsl.org.mx'>enlace 1</a>
<a class='testClick' href='http://google.com'>enlace 2</a>
<a class='testClick' href='../sesion03/index.html'>enlace 3</a>
```

Código JS

```
$('.a.testClick').click( function(event) {
  $('.log').html( '<strong>Resultado:</strong> ' + $(this).text() );
  event.preventDefault();
});
```

Comportamiento en cola.

jQuery tiene un comportamiento en cola (el primero en entrar es el 1ro en ejecutarse), para explicar esto observe el siguiente código:

```
$("#a")
  .click(function() {
    console.log('Ejecución del manejador1 para el evento click del enlace');
  })
  .click(function() {
    console.log('Ejecución del manejador2 para el evento click del enlace');
  });
```

¿Qué cree que suceda cuando le demos click a un enlace(a) en la página?.

La respuesta la podemos encontrar al reflexionar qué sucede cuando mandamos a llamar múltiples veces el evento `document.onReady`.

Lo que hace **jQuery** es ocupar una cola para administrar los eventos, de esta forma se pueden agregar múltiples funciones sobre el mismo evento/elemento, por lo cual el resultado en la consola sería:

1. Ejecución del manejador1 para el evento click del enlace
2. Ejecución del manejador2 para el evento click del enlace

Evento hover

El evento `hand over` o `hover` es disparado cuando pasamos el cursor por encima de algún elemento, la sintaxis básica es la siguiente.

```
$("#a").hover(function() {
  console.log('Ejecución del manejador para el evento hover del enlace');
});
```

Cabe mencionar que este evento puede soportar 2 manejadores, el primero es ejecutado cuando el cursor pase por encima del elemento y el segundo es ejecutado cuando el cursor se quita de dicho elemento, p.e.:

```
$("#a").hover(
  function() {
    $(this).css('color', 'red');
  },
  function() {
    $(this).css('color', 'blue');
  }
);
```

En este caso estamos agregando 2 manejadores el resultado será que cuando pasemos el mouse por encima de un enlace el color de la fuente se cambiara a rojo y cuando quitemos el cursor el color será azul, veamos otro ejemplo:

Ejemplo - Hover

código HTML:

```
<div id='box'></div>
```

CSS

```
#box{
  width: 100px;
  height: 100px;
  background: orange;
  display: block;
  border-radius: 5px;
}
```

código JS:

```
$("#box").hover(
  function () {
    $(this).animate({ 'width': "300px", "height" : "300px"});
  },
  function () {
    $(this).animate({ 'width': "100px", "height" : "100px"});
  }
);
```

Eventos del teclado

jQuery define un conjunto de [eventos para el teclado](#) los eventos que definen son los siguientes.

focusout

Vincula a los inputs sobre el evento out Focus.

Mayor información: <http://api.jquery.com/focusout/>

keypress

El evento es disparado cuando una tecla es presionada.

Mayor información: <http://api.jquery.com/keypress/>

keyup

El evento es disparado cuando una tecla deja de estar presionada.

Mayor información: <http://api.jquery.com/keyup/>

keydown

El evento es disparado cuando se encuentra una tecla presionada.

Mayor información: <http://api.jquery.com/keydown/>

Ejemplo de eventos de teclado:

Como podemos ver las funciones `keyup`, `keydown` y `keypress`, veamos un simple ejemplo:

```
$('#target').keyup(function(event) {
```

```

var msg = 'El evento keyup() fue llamado para la tecla ' + event.which;
console.log(msg);
}).keydown(function(event) {
  if (event.which == 13) {
    event.preventDefault();
  }
});

```

Ejemplo - Evento Teclado

código HTML:

```
<div id="ecenario"><div id="pollito"></div></div>
```

código CSS:

```

#ecenario {
  width: 100%;
  min-width: 500px;
  height: 180px;
  background: #6FD9FF;
  background-repeat: repeat-x;
  background-image: -moz-linear-gradient(top, #6FD9FF, #289FCB);
  background-image: -ms-linear-gradient(top, #6FD9FF, #289FCB);
  background-image: -webkit-gradient(linear, left top, left bottom, from(#6FD9FF),
to(#289FCB));
  background-image: -webkit-linear-gradient(top, #6FD9FF, #289FCB);
  background-image: -o-linear-gradient(top, #6FD9FF, #289FCB);
  background-image: linear-gradient(top, #6FD9FF, #289FCB);
  filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#6FD9FF',
endColorstr='#289FCB',GradientType=0 );
  border-bottom: 25px solid #208A00;
}
#pollito {
  position: absolute;
  margin-left: 400px;
  margin-top: 70px;
  background: URL('pollito.png') no-repeat;
  width: 161px;
  height: 134px;
}

```

Código jQuery:

```

$pollito = $("#pollito");
$ecenario = $("#ecenario");
pos_x = 400;

$('body').keydown(function(event) {
  switch(event.which) {
    case 39:
      $pollito.css("background-position", "0 0");
      if(pos_x < ($ecenario.width() - $pollito.width()) ) {
        pos_x += 10;
        $pollito.css("margin-left", pos_x + "px");
      }
      break;
    case 37:
      $pollito.css("background-position", "-161px 0");
      if(pos_x > 0) {
        pos_x -= 10;
        $pollito.css("margin-left", pos_x + "px");
      }
      break;
    default:
      console.log('Tecla: '+ event.which);
  }
});

```

Borrando eventos función off.

Podemos desvincular los eventos con la función `off` por ejemplo para eliminar el evento `click` en `H1` podríamos hacer.

```
$("#h1").off('click');
```

Con esto eliminaríamos todos los eventos `click` del `h1`, si quisiéramos eliminar un único evento `click`, podríamos hacer uso de los espacios de nombres.

Desde la versión 1.7 de **jQuery**,

Enlaces recomendados.

- ✓ [Dos formas de propagación de un evento](#)

Efectos y modificaciones sobre el DOM.

insertbefore

```
$('#<p>Test</p>').insertBefore('#selector');
```

before

```
$('#selector').before('<p>Test</p>');
```

insertafter

```
$('#<p>Test</p>').insertAfter('#selector');
```

after

```
$('#selector').after('<p>Test</p>');
```

append

```
$('#selector').append('<p>Test</p>');
```

appendto

```
$('#<p>Test</p>').appendTo('#selector');
```

fadeIn

```
$('#selector').fadeIn();
```

También puede llevar un argumento como `slow`, `fast`.

```
$('#selector').fadeIn('slow');
```

ó agregar un valor en milisegundos.

```
$('#selector').fadeIn('slow');
```

Los valores de `slow`, `fast`, están predeterminados en la librería usted puede ver los valores, predeterminados de los efectos.

```
console.log($.fx.speeds);
```

Animate

Forma simple:

```
$("#selector").animate('width', "500px");
```

Forma relativa:

```
$("#selector").animate({"left": "+=50px"}, "slow");
```

Multiples parametros.

```
$("#selector").animate({
  width: "70%",
  opacity: 0.4,
  marginLeft: "0.6in",
  fontSize: "3em",
  borderWidth: "10px"
}, 1500 );
```

animate sobre un color.

Desgraciadamente efectos como los colores no funcionan del todo bien, anteriormente había un plugin jquery-plugin-color que nos ayudaba a realizar integrar comportamiento pero ya no le han dado mantenimiento, sin embargo una forma de realizar determinada característica es por medio de [jQuery-Ui](#), del cual para este caso requerimos el núcleo **jQuery-Ui** (core), el núcleo de los efectos (Effects Core) , y finalmente la opción Effect "Highlight", una vez esto obtenemos un archivo con el nombre jquery-ui-1.8.21.custom.min.js el cual contiene de forma compacta lo que requerimos, ahora lo siguiente es agregarla a la cabecera de nuestro archivo y hacer un tests:

Código HTML.

```
<div class="box"></div>
```

Código CSS.

```
div.box{
  width: 100%;
  height: 200px;
  background:#666;
  border:1px solid #999;
}
```

Código jQuery.

```
colors = ['#FFB30C', '#58EC00', '#0087EC', 'EEEEEE', '#FF5A00' ];
var i = 0;
$box = $('div.box');
animate_loop = function() {
  $($box).animate({backgroundColor:colors[(i++)%colors.length]
  }, 900, function(){
    animate_loop();
  });
}
animate_loop();
```

show

forma simple.

```
$('.selector').show();
```

forma con velocidad.

```
$('.selector').show('fast');
```

hide

forma simple.

```
$('.selector').hide();
```

forma con velocidad.

```
$('.selector').hide('fast');
```

Lecturas recomendadas.

- ✓ [Categoría efectos jQuery](http://api.jquery.com/category/effects/) - <http://api.jquery.com/category/effects/>
- ✓ [Manipulación jQuery](http://api.jquery.com/category/manipulation/) - <http://api.jquery.com/category/manipulation/>
- ✓ [Efectos jQuery-Ui](http://jqueryui.com/download) - <http://jqueryui.com/download>

Ajax.

Ajax, acrónimo de Asynchronous JavaScript And XML (*JavaScript asíncrono y XML*), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar

cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

En concepto parte del hecho que desde javascript podemos realizar solicitudes(`httpRequests`) y que por medio que *javascript* no espera el resultado de dicha solicitud para continuar su flujo.

Ajax y JQuery.

Personalmente siento que el concepto es algo confuso con **jQuery**, ya que propiamente no se debería llamar ajax si no HttpRequest ó simplemente Request ([como funciona en mootools](#)).

En adelante cuando se mencione ajax en la jerga de **jQuery** nos referimos a una funcion de jQuery que realiza un **http request**, espero que quede más claro con algunos ejemplos:

Ejemplo básico

```
$.ajax({
  url: 'ajax/test.html',
  success: function(data) {
    $('.result').html(data);
    alert('Load was performed.');
```

Quitando la opción asincrona.

Observe el siguiente código.

```
var datos = null;
$.ajax({
  url: 'ajax/test.html',
  success: function(data) {
    datos = data;
  }
});
console.log(datos);
```

Datos será `null`, ya que la respuesta es asíncrona es decir `console.log()` no espera la finalización de `$.ajax` para ejecutarse, hacer `async` igual a `false` resulta una opción útil para esta logica.

```
var datos = null;
$.ajax({
  url: 'ajax/test.html',
  //detiene la ejecución.
  async : false,
  success: function(data) {
    datos = data;
  }
});
console.log(datos);
```

Otra opción para esperar que ajax se haya ejecutado para realizar determinada acción sería encaminar la función `done` como se muestra a continuación.

```
var datos = null;
$.ajax({
  url: 'ajax/test.html',
  success: function(data) {
    datos = data;
  }
}).done(function( msg ) {
  console.log(datos);
});
```

Ejemplo Manejando el error.

Si somos detallistas nos daremos cuenta que `console.log(datos);` no garantiza que los datos no sean nulos, pues la respuesta puede llegar a fallar y en este caso tal vez deseamos rellenar a datos con valores por defecto.

```

var datos = null;
$.ajax({
  url: 'ajax/test.html',
  success: function(data) {
    datos = data;
  }
  error: function() {
    datos = {v1:'valor defecto 1', v2:'valor defecto 2'};
  }
}).done(function( msg ) {
  console.log(datos);
});

```

Otras funciones Ajax

Existen otras funciones definidas para manejar respuestas asíncronas. Para un detalle más extenso se recomienda consultar la categoría **Ajax** de la api de **jQuery**:

getScript

Es una función ajax de **jQuery** equivalente a:

```

$.ajax({
  url: url,
  dataType: "script",
  success: success
});

```

A partir de **jQuery 1.5** puedes ocupar la función `.fail()` para detectar y manejar errores.

```

$.getScript("ajax/test.js")
  .done(function(script, textStatus) {
    $( "div.log" ).text( textStatus );
  })
  .fail(function(jqxhr, settings, exception) {
    $( "div.log" ).text( "Triggered ajaxError handler." );
  });

```

Ejemplo - getScript

Código JS

```

$('a.button').one("click", function(e){
  $.getScript("cambiar.color.js")
  .done(function(script, textStatus) {
    $( "div.log" ).text( textStatus );
  })
  .fail(function(jqxhr, settings, exception) {
    $( "div.log" ).text( "Error al cargar el guion." );
  });
  //Detiene el comportamiento del evento
  e.preventDefault();
});

```

Código CSS

```

div.log {
  padding: 10px;
  color: #004983;
  background: #ADE0FF;
}

```

Código HTML

```

<div class="log"></div>
<a class='button' href="#">cargar dinamicamente un script</a>

```

getJSON

Esta función es en particular muy útil ya que varios servicios por Internet nos brindan una API y por lo regular los datos son enviados en formato json.

<http://api.jquery.com/jquery.getJSON/>

Ejemplo - getJson y la API de Flickr

Código HTML

```
<a class='button' href="#">Cargar info de Flickr</a>
<div id="imagenes"></div>
```

Código CSS:

```
#imagenes {
  color: #131A04;
  background: #CFEC99
}
#imagenes img{
  margin: 10px auto;
}
```

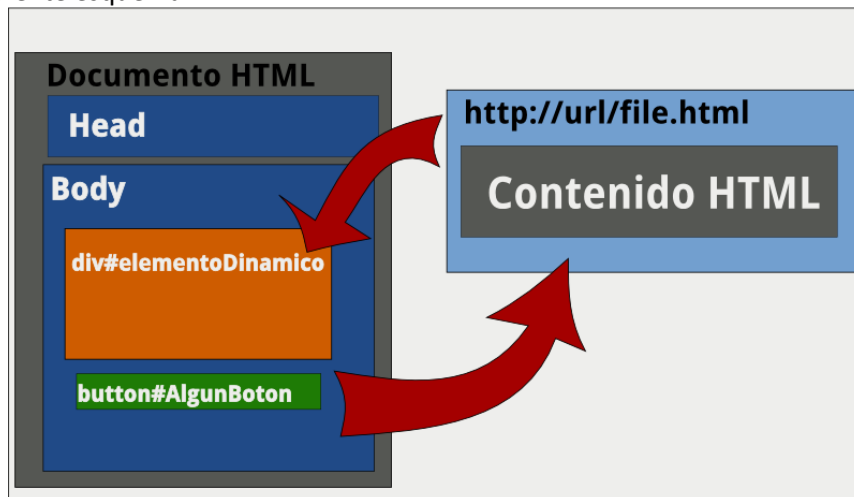
Código jQuery

```
var flickerAPI = "http://api.flickr.com/services/feeds/photos_public.gne?jsoncallback=?";

$('a.button').one('click', function(e){
  $.getJSON( flickerAPI, {
    tags: "fitorec",
    tagmode: "any",
    format: "json"
  }).done(function( data ) {
    $.each( data.items, function( i, item ) {
      $( "<img/>" ).attr( "src", item.media.m ).appendTo( "#imagenes" );
      if ( i === 3 ) {
        return false;
      }
    }); // Fin each
  }); // Fin done de getJSON
  e.preventDefault();
});
```

Ejemplo load

Entre las funciones que ofrece **jQuery** para el control asincrono esta la función [load](#) dicha función se encarga de cargar algun contenido obtenido desde alguna url, para entender mejor esta función observe el siguiente esquema:



El esquema anterior muestra un documento **Html** el cual contiene un **div** identificado como `elementoDinamico` el cual va a cargar el contenido de `http://url/file.html` en el momento que hagan click sobre el botón identificado como `algunBoton`. Bien ahora veamos como quedaría este ejemplo:

```
//relacionamos el evento click del botón #algunBoton
$('#algunBoton').click(function(event) {
  //En el div #elementoDinamico vamos a cargar el contenido de file.html
  // Notar que el parámetro que recibe puede ser una url absoluta o un ruta referenciada.
  $('#elementoDinamico').load('file.html');
  // Detenemos los demás eventos(si no hacemos esto subiría el scroll)
  event.preventDefault();
});
```

Lecturas recomendadas.

- ✓ <http://api.jquery.com/category/ajax/>
- ✓ [Usando herramientas de calidad de código - por Anton Kovalyov \(inglés\)](#)
- ✓ [Principios para escribir JavaScript consistente e idiomático](#)

Organización de código.

Organizando el código en archivos.

Como sabemos con **jQuery** podemos agregar múltiples funciones para manejar un evento, y estas funciones se van a ir ejecutando en formato de cola.

Ahora imagine que tiene 2 documentos, uno con el nombre **titulos.js** con el siguiente contenido:

```
$(document).ready(function(){
  //Cambiamos el color para el titulo nivel 2
  $('h2').css('color', 'green');

  //Agregamos un evento para h1
  $('h1').hover( function(){
    $(this).css('color', 'red');
  },function(){
    $(this).css('color', '#222');
  });
});
```

Por otra parte tenemos el archivo con el nombre **enlaces.js** con el siguiente contenido:

```
$(document).ready(function(){
  //Obliga abrir los enlaces en una pestaña nueva
  $('a').attr('target', 'blank');
});
```

Si mandamos a llamar ambos documentos de la manera siguiente:

```
<script src="js/titulos.js"></script>
<script src="js/enlaces.js"></script>
```

¿Qué crees que pase?.

Es una práctica bastante común separar la lógica en archivos distintos de esta manera tenemos más ordenado nuestro código.

Ejemplo - tabla de contenidos

Observe la organización del siguiente código:

JS

```
tablaContenido = (function(div_contenedor, div_destino, tag_titulos) {
  /* variables a utilizar */
  var $content = $(div_contenedor), /* contenedor principal */
  $destino = $content.find(div_destino), /* div q contendra la tabla*/
  $tagTitulos = $content.find(tag_titulos), /*Titulos a observar*/
  modo = null, /* nos indica el tipo del archivo a generar la tabla*/
  pie_links = '', /* forma del pie de links de cada sección */
  titulo = null, /* titulo de la tabla de contenidos */
  /* funcion debug util para depurar */
  debug = function () {
    console.log($content);
    console.log($destino);
    console.log($tagTitulos);
    console.log(document.title);
    console.log(modo);
  },
  /* devuelve un string que contiene un link apartir de la url y del content */
  link = function (url,content){
    return '<a href='+url+'>'+content+'</a>';
  },
});
```

```

/* A partir del archivo(fileName) obtenemos el modo
 * index          - Pertenece a la pagina principal(index)
 * index.sesion   - Pertenece a una sesión específica.
 * index.ejemplos - Index de ejemplos.
 * algun.ejemplo  - Algun ejemplo.
 * algun.recurso  - Algun recurso.
 */
obtenerModo = function (value){
  var path = window.location+'';
  //Modo index agregamos botones de descarga
  if( pageConf.fileName == '/index.html' ){
    modo = 'index';
    repoURL = 'https://github.com/mundoSICA/tutorial_hispano_jQuery';
    pie_links += link(
      repoURL + '/tarball/master',
      '&darr; Descargar <strong>Tar.gz</strong>'
    ) + ' | ' + link(
      './todo/index.html',
      'Versión imprimible <strong>HTML</strong>'
    ) + ' | ' + link(
      './todo/manual_jquery_basico.pdf',
      '&darr; Versión imprimible <strong>PDF</strong>'
    ) + ' | ' + link(
      repoURL + '/zipball/master',
      '&darr; Descargar <strong>Zip</strong>'
    );
    //Modo index.sesion add botones de tabla de contenidos e index principal
  }else if( pageConf.fileName.match(/^\/sesion0[1-9]\/index.html/g) ){
    modo = 'index.sesion'
    pie_links += link(
      '#tabla_contenidos',
      '&uarr; Tabla de Contenidos'
    ) + ' | ' + link(
      './index.html',
      '&Theta; Indice principal'
    );
    titulo = 'Tabla de Contenidos';
    //Modo index.recurso (no hace nada)
  }else if( pageConf.fileName == '/recursos/index.html' ){
    modo = 'index.recurso';
    pie_links += '';
    titulo = '';
    //Modo index.ejemplo
  }else if( pageConf.fileName == '/ejemplos/index.html' ){
    modo = 'index.ejemplo';
    pie_links += link('./index.html', '&Theta; Indice principal');
    titulo = ' ';
    //Modo algun.ejemplo
  }else if( pageConf.fileName.match(/^\/ejemplos/g) ){
    titulo = 'Contenido del ejemplo';
    modo = 'algun.ejemplo';
    ses = pageConf.fileName.split('/').pop().split('.').shift();
    pie_links += link(
      '#tabla_contenidos',
      '&uarr; Tabla de Contenidos'
    ) + ' | ' + link(
      './sesion'+ses+'index.html',
      'Ir a la Sesion '+ses
    ) + ' | ' + link(
      './index.html',
      '&Theta; Indice principal'
    );
    //Modo version.impresa, los unicos botones son los del pie de página
  }else if( pageConf.fileName == '/todo/index.html' ){
    titulo = 'Versión impresa';
    modo = 'version.impresa';
  }
},
/* Recibe una cadena de texto y la convierte en slug */
slug_str = function( str ){
  return $.trim(str).toLowerCase().replace(/:/g, "").
  replace(/./g, "").replace(/s+/g, "-").replace(/á/g, "a").
  replace(/é/g, "e").replace(/i/g, "i").replace(/ó/g, "o").
  replace(/ñ/g, "ni").
  replace(/ú/g, "u").replace(/\/g, "-").replace(/\/g, "");
},
/* a partir del modo devuelve lo que sera los links para el pie de página */

```

```

links_fin_content = function(){
  if( modo == 'index.sesion'){
    pie_links = link(
      '../index.html',
      '&Theta; Índice principal'
    );
    if( pageConf.sesion > 1 ){
      ses_prev = '0'+(pageConf.sesion-1);
      pie_links = link(
        '../sesion'+ses_prev+'/index.html',
        '&laquo; Sesión '+ses_prev
      ) + ' | ' + pie_links;
    }
    if( pageConf.sesion < pageConf.num_sesiones ){
      ses_sig = '0'+(pageConf.sesion+1);
      pie_links += ' | ' + link(
        '../sesion'+ses_sig+'/index.html',
        'Sesión '+ses_sig + ' &raquo;'
      );
    }
  }
  else if( modo == 'version.impresa' ){
    pie_links += link(
      '../index.html',
      '&Theta; Índice principal'
    ) + ' | ' + link(
      '#tabla_contenidos',
      '&uarr; Tabla de Contenidos'
    ) + ' | ' + link(
      './manual_jquery_basico.pdf',
      '&darr; Versión imprimible <strong>PDF</strong>'
    );
  }
  $('<div class="links_paginacion">'+pie_links+'</div>' )
    .appendTo($content);
},
/* Genera la tabla de contenido segun el modo */
generaTabla = function (){
  if( modo == null ){
    return;
  }
  tContenidoHtml = '';
  if(titulo){
    tContenidoHtml = '<h2>' + titulo + '</h2><ol id="lista_contenidos">';
  }
  //Es buena practica el almacenar la longitud de un objeto antes de iterar
  for(i=1,l=$tagTitulos.length; i<l; i++){
    h2_actual = $tagTitulos[i];
    // Generamos el nuevo identificador que tendrá el h2 actual
    nuevoID = slug str($h2_actual.text());
    $(h2_actual).attr('id', nuevoID);
    $(h2_actual).before('<div class="links_paginacion">'+pie_links+'</div>');
    // Creamos un item de la lista(<li>) con un link(<a>)
    // con referencia(href) al #nuevoID
    if( titulo ) {
      tContenidoHtml += '<li>' +
        link('#' + nuevoID,$h2_actual.html()) +
        '</li>';
    }
  }
  //generamos los links que pondremos en el pie de pagina
  links_fin_content()
  if(titulo) {
    $destino.append(tContenidoHtml+ '</ol>');
    //insertamos la ancla para que suba a la tabla de contenido
    $content.find('.linkTablaContenido').click(function(event) {
      $( document ).scrollTop(0);
      event.preventDefault();
    });
  }
  if(modo=='index'){
    generaLinksIndex();
  }
},
/* Genera el indice de la página principal */
generaLinksIndex = function() {
  padreLink = '';
  href = '';

```

```

$content.find('ul a').each(function(key, link) {
    href=$(this).attr('href');
    if( href.length > 1 ){
        padreLink = href;
    }else{
        $(this).attr('href', padreLink+'#'+ slug_str($(this).text() )
    }
});
},
/* Mueve el scroll en la posicion y del elemento seleccionado */
scrollTop = function (){
    var location = window.location + '';
    if( location.indexOf('#') != -1 ){
        $el = $('#'+ location.split('#').pop() );
        if($el.length){
            $( document ).scrollTop($el.offset().top-$('#head').outerHeight(true));
        }
    }
},
/* Donde todo inicia */
init = function() {
    obtenerModo();
    generaTabla();
    scrollTop();
};
/* Mandamos a ejecutar el script */
init();
})( '#content', '#tabla_contenidos', 'h2');

```

jQuery Plugins.

Observe el siguiente código:

```

(function($){
    $.yourPluginName = function(el, radius, options){
        // To avoid scope issues, use 'base' instead of 'this'
        // to reference this class from internal events and functions.
        var base = this;
        // Access to jQuery and DOM versions of element
        base.$el = $(el);
        base.el = el;
        // Add a reverse reference to the DOM object
        base.$el.data("yourPluginName", base);
        base.init = function(){
            if( typeof( radius ) === "undefined" || radius === null ) radius = "20px";
            base.radius = radius;
            base.options = $.extend({}, $.yourPluginName.defaultOptions, options);
            // Put your initialization code here
        };
        // Sample Function, Uncomment to use
        // base.functionName = function(parameters){
        //
        // };
        // Run initializer
        base.init();
    };
    $.yourPluginName.defaultOptions = {
        radius: "20px"
    };
    $.fn.yourPluginName = function(radius, options){
        return this.each(function(){
            (new $.yourPluginName(this, radius, options));
            // HAVE YOUR PLUGIN DO STUFF HERE
            // END DOING STUFF
        });
    };
})(jQuery);

```

Uso.

```

$(function() {
    $("#round-me").yourPluginName("20px");
});

```

Ejemplo plugin jquery

Código JS

```
(function($){
  $.linkEjemplos = function(el, options){
    // To avoid scope issues, use 'base' instead of 'this'
    // to reference this class from internal events and functions.
    var base = this;
    // Access to jQuery and DOM versions of element
    base.$el = $(el);
    base.el = el;
    // Add a reverse reference to the DOM object
    base.$el.data("linkEjemplos", base);
    base.init = function(){
      base.options = $.extend({}, $.linkEjemplos.defaultOptions, options);
      // Put your initialization code here
    };
    // Run initializer
    base.init();
    base.$el.addClass(base.options.clase).attr('target', base.options.target);
  };
  $.linkEjemplos.defaultOptions = {
    'clase' : "button",
    'target' : "blank",
  };
  $.fn.linkEjemplos = function(options) {
    return this.each(function(){
      (new $.linkEjemplos(this, options));
    });
  };
})(jQuery);
```

Uso.

Insertamos el plugin.

```
<script type='text/javascript' src='../js/jquery.linkEjemplos.js'></script>
```

Lo usamos tal cual:

```
$('.a[href^="../ejemplos/"]').linkEjemplos();
```

Podemos ocupar los argumentos de opcionales:

```
$('.a[href^="../ejemplos/"]').linkEjemplos({'clase': 'link', 'target': '_self'});
```

Fuente: <http://css-tricks.com/snippets/jquery/jquery-plugin-template/>

Anexo I - Etiquetas en la cabecera.

Todos los siguientes ejemplos se contemplan que estarán dentro de la etiqueta head

```
<head>
  /* aqui viene el código de ejemplo */
</head>
```

Inserción de código javascript

Básicamente hay 2 formas de insertar código javascript en nuestros documentos **Html**, la primera es la inserción en línea:

```
<script type="text/javascript">
  //Código javascript
</script>
```

La segunda forma es definir el código en otro archivo y mandarlo a llamar, esto lo hacemos de la siguiente forma:

```
<!-- Insertando /js/miScript.js -->
```

Inserción de código CSS.

De forma análoga hay 2 formas de insertar código CSS en nuestras páginas web, la primera inserción en línea:

```
<style type="text/css" media="all">
  /* código css */
</style>
```

La segunda mandando a llamar archivo remoto:

```
<link rel="stylesheet" type="text/css" media="all" href="/ruta/estilo.css" />
```

Tipo de caracteres.

Es útil decir el tipo de caracteres que tendrá la página, por ejemplo para caracteres acentuados como es el caso de la lengua hispana:

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
```

Descripción de la página.

Existen una serie de etiquetas y metadatos que sirven para describir la información de la página esta información es utilizada por los navegadores para indexar. La información siguiente fue extraída del artículo [google metaetiquetas](#).

Title.

```
<title>título de la página</title>
```

Se suele utilizar en junto con la "descripción"(description). El contenido de esta etiqueta suele mostrarse como el título de los resultados de búsqueda (y, por supuesto, en el navegador del usuario). [Más información](#).

Descripción (Description)

Proporciona una descripción breve de la página. En algunos casos esta descripción se utiliza como parte del fragmento que se muestra en los resultados de búsqueda. [Más información](#).

```
<meta name="description" content="Descripción de la página" />
```

Robots.

Nos ayudan a controlar el comportamiento del rastreo y la indexación del motor de búsqueda. La metaetiqueta robots se aplica a todos los motores de búsqueda. Los valores predeterminados son "index, follow" (igual que "all") y no es necesario especificarlos. Google entiende los siguientes valores (si especificas varios valores, sepáralos con una coma):

- ✓ **noindex**: impide que la página se indexe.

- ✓ **nofollow**: impide que los buscadores sigan los enlaces de esta página.
- ✓ **nosnippet**: impide que un fragmento se muestre en los resultados de búsqueda.
- ✓ **noodp**: impide que se utilice la descripción alternativa de ODP/DMOZ.
- ✓ **noarchive**: impide que Google muestre el enlace En caché de una página.
- ✓ **unavailable_after[date]**: te permite especificar la hora y la fecha exactas en que quieres detener el rastreo y la indexación de la página.
- ✓ **noimageindex**: te permite especificar que no quieres que la página aparezca como la página de referencia de una imagen que se muestra en los resultados de la búsqueda de Google.

Ahora también se puede especificar esta información en la cabecera de las páginas mediante la directiva de cabecera HTTP "X-Robots-Tag". Esto resulta especialmente útil para limitar la indexación de archivos que no sean HTML, como archivos gráficos y otro tipo de documentos. Más información sobre el archivo [robots.txt](#).

```
<!-- quitando indexación y no explorar -->  
<meta name="robots" content="noindex, nofollow">
```

Parabras clave Keywords:

Pese a que esta *meta-etiqueta* no es utilizada por **google** para indexar búsquedas, es recomendable utilizarla ya que otros buscadores como **Yahoo** si lo hace, la sintaxis es la siguiente:

```
<meta name="Keywords" content="jquery,manual,ejemplos,metaetiquetas,keywords">
```

Como podemos ver podemos definir multiples palabras clave separadas por coma (,).

Enlaces recomendados:

- ✓ **Google metaetiquetas:**
 - <http://support.google.com/webmasters/bin/answer.py?hl=es&hlrm=en&answer=79812>
- ✓ **Robots bloqueando seguimiento:**
 - <http://searchengineland.com/meta-robots-tag-101-blocking-spiders-cached-pages-more-10665>
- ✓ **Yahoo como agregar metatags:**
 - <http://help.yahoo.com/l/us/yahoo/smallbusiness/webhosting/promote/promote-03.html>
- ✓ **Yahoo metatags por defecto:**
 - <http://help.yahoo.com/l/us/yahoo/smallbusiness/webhosting/yss/promote/promote-04.html>
- ✓ **Google Microdatos HTML5:**
 - <http://support.google.com/webmasters/bin/answer.py?hl=es&hlrm=en&answer=176035>
- ✓ **W3C Microdatos HTML5:**
 - <http://www.w3.org/TR/html5/microdata.html>
- ✓ ***Fragmentos enriquecidos_(microdatos, microformatos y RDFa)_***:
 - <http://support.google.com/webmasters/bin/answer.py?hl=es&answer=99170>
 - <http://www.data-vocabulary.org/>