

Jquery: Javascript simple, aplicaciones complejas.

Curso E-Ghost Julio 2008



ironotec
Internet y Sistemas sobre GNU/Linux

ironotec
GNU/Linux

- ¿Qué es Javascript?
 - Javascript != Java
 - Creado por Bredan Eich para Navigator 2.0 (1995)
 - Lenguaje ligero interpretado en navegadores web (aunque no exclusivamente)
 - Javascript es una implementación de ECMAScript (lenguaje de scripting basado en prototipos)
 - El script se descarga directamente al navegador, y es ejecutado siempre en el lado del cliente: tecnología del cliente, dependiente del software del cliente (de su navegador).

- ¿Cómo debería ser Javascript?
 - Multinavegador
 - Debe respetar las “implementaciones” de todos los navegadores y sus versiones más utilizadas.
 - No Intrusivo
 - Separar Javascript / código XHTML / CSS.
 - Versiones accesibles.
 - Ligero
 - Facilitar tiempos de carga
 - jsmin / mod-gzip / etc...
 - Ejecución rápida
 - Es un lenguaje de muy alto nivel... su ejecución a veces de demasiado lenta, innavegable casi...

- DOM
 - Document Object Model
 - Representación estándar de un documento XML (o HTML/XHTML), que representa como objetos los elementos que lo componen.
 - Independiente del lenguaje y/o de la plataforma.
 - Permite acceder y modificar dinámicamente tanto la estructura y como el estilo de documentos.
 - Javascript posee un conjunto de métodos que permite acceder y modificar los objetos y atributos de un documento XHTML.
 - Existe un estándar del W3C, implementado a medias por la mayoría de los navegadores.

- ¿Qué es jQuery?
 - Librería abstracción Javascript
 - Creada por John Resig (Sept. 2005)
 - Se encarga de “gustar” a todos los navegadores
 - IE6.0 / Firefox 2.0 / Safari 2.0 / Opera 9.0
 - Acceso DOM ágil y optimizado
 - Permite animaciones simples
 - Manejador Eventos
 - Facilitar AJAX
 - Ahorro espectacular de muchas líneas de código

- ¿Por qué jQuery? (y no otra librería de abstracción)
 - API sencilla y potente
 - Muy ligera (16Kb gzipeada)
 - Potente sistema de plugins (<http://plugins.jquery.com>)
 - Buen soporte de la comunidad
 - Gran continuidad en su desarrollo
 - Si Google, IBM, BBC, Wordpress, Drupal, Digg, Meneame o Irontec ;) utilizan jQuery... por algo será.
 - Funciona junto con otras librerías de abstracción

- Como incluir jQuery en un documento xHTML
 - Simplemente añade en la cabecera de tu documento:

- Puedes descargar la última versión:

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

- Puedes vincular directamente con la última versión:

```
<script type="text/javascript"  
src="http://code.jquery.com/jquery-latest.min.js"></script>
```

OJO: No siempre existe compatibilidad hacia atrás con jQuery. Para un sitio en producción, mejor descargarse la librería para utilizarla.

- `$()`
 - Esta función es el núcleo de jQuery.
 - En realidad es un alias para la función `jQuery()`.
 - Encapsula 1 ó más objetos DOM y permite interactuar con ellos de varias maneras.
 - Acepta casi cualquier parámetro que parseará e introducirá en un objeto jQuery (selectores).
 - Automáticamente interactúa sobre los N objetos DOM que sean resueltos mediante el selector escogido.

```
$('p') // devuelve todos los nodo p en el DOM  
$('#id2') // devuelve el nodo con el atributo 'id' = 'id2'  
$('.c3') // devuelve todos los nodos con la clase 'c3'
```

- Constructor `$()`

- `$('.clase3')`

Un selector jQuery tradicional

- `$('.clase3', $('#id2'))`

delimita la búsqueda de selector dentro de selector_contexto.

- `$(document.getElementById(id2).childNodes())`

Convierte en objeto/s jQuery un objeto DOM tradicional

- `$('<h2>hola mundo</h2>')`

Convierte un string HTML en un nuevo elemento DOM (no lo incluye en el body automáticamente).

- `$('#.clase3:eq(0)')`

Clona otro objeto jQuery

- Selectores CSS(1/3)

- jQuery soporta casi todas las implementaciones CSS (desde la implementación 1, hasta la 3).
- Muy útil y bien usado, optimiza mucho el parseo.

- Tags

- `$('p')`

Devuelve un array con todos los objetos tipo 'p' del documento.

- ID

- `$('#contenido')`

Devuelve un array con un único elemento (ó 0), conteniendo el tag con identificador 'contenido'

- Clase

- `$('.azul')`

devuelve un array con todos los elemento con clase 'azul'

- Selectores CSS (2/3)

- Descendientes

- `$(p ul)`

Devuelve un array con todos los objetos 'ul' que estén dentro de objetos 'p'

- Hijos

- `$(#contenido > p)`

Devuelve un array con todos 'p' que sean hijos (y no nietos) del objeto con identificativo 'contenido'

- Negadas

- `$(.azul:not(#contenido))`

Devuelve un array con todos los elementos con clase 'azul', pero que no tenga un identificador 'contenido' en su atributo 'id'.

- Selectores CSS (3/3)

- Selector Múltiple

- `$(p,div,span)`

- Devuelve un array con todos los objetos 'p', con todos los objetos 'div' y con todos los objetos 'span'.

- Universal

- `$(*)`

- Devuelve un array con los objetos de un documento.

- `$(div *)`

- Devuelve un array con todos los objetos que esten contenidos en un objeto 'div'.

- Selectores XPATH

- Utiliza XML Path Language para identificar nodos dentro del XML.

- Contiene

- `$(p[ul])`

- Devuelve todos los objetos 'p' que contengan objetos 'ul'

- Por atributo

- `$(a[@href$=.pdf])`

- Devuelve todos los 'a' que contengan un atributo 'href' que termine en '.pdf'

- `$(a[@href^=mailto])`

- Devuelve todos los 'a' con un href que comience con 'mailto'

- `$(a[@href=/]) $(a[@href*=imgs]) $(a[@href!=mailto])`

- Selectores propios (1/2)

- Par (even) / Impar (odd)

- `$('.tr:even')` `$('.tr:odd')`

Devuelve todos los objetos 'tr' pares / impares

- Primero (first) / Último (last)

- `$('.tr:first')` `$('.tr:last')`

Devuelve el primer / último objeto 'tr'

- Mayor que (gt) / Menor que (lt)

- `$('.tr:gt(2)')` `$('.tr:lt(2)')`

Devuelve todos los elementos 'tr' mayores / menores que 2

- Selectores propios (2/2)
 - Elemento exacto
 - \$('tr:eq(3)') (de 0 a n)
 - \$('tr:nth(3)') (de 1 a n)

Devuelve un array con el cuarto y el tercer objeto 'tr'

- Visible (visible) / Oculto (hidden)
 - \$('li:visible') \$('li:hidden')

Devuelve todos los li visibles /ocultos

- Conteniendo texto
 - \$('tr:contains(tralara)')

Devuelve todos los objetos 'tr' que contengan el texto 'tralara'.

Selectores [formulario]

- Selectores de formulario
 - Son selectores propios, pero especiales para elementos de formularios
 - :input
 - :text
 - :password
 - :radio
 - :checkbox
 - :submit
 - :image
 - :reset
 - :button
 - :file
 - :hidden

- Funciones 'prototype' que pueden ser llamadas, por el valor devuelto desde \$().

```
$('.contenido').addClass( bonito );
```

- Casi todos los métodos son ENCADENABLES

```
$('.contenidos').addClass( bonito ).bind( click ,alertar);
```

- Todos los métodos se devuelven a si mismos.
 - Excepto los que devuelven un string (val() / text())
 - Excepto los métodos trasversales DOM

- Métodos aplicables a selectores que modifican sus valores css.

```
$.css('propiedad')  
$( '#contenido' ).css('color') // Devuelve el valor de la  
propiedad
```

```
$.css('propiedad','valor')  
$( '#contenido' ).css('color','#ff0000') // Setea el valor en  
la propiedad css.
```

```
$.css(array('propiedad':'valor'))  
$( '#contenido' ).css({'color':'#ff0000','background-  
color':'#00ff00'}) // Setea los valores el array
```

Métodos [Atributos DOM]

- Métodos aplicables a selectores que modifican los atributos de objetos DOM devueltos por el selector.

```
$.addClass( clase ) / $.removeClass( clase )  
// añade / elimina una clase al objeto jQuery
```

```
$.attr( n_attr ) / $.attr( n_attr , 'valor')  
// devuelve / asigna el nombre de un atributo
```

```
$.html() / $.html('strHTML')  
//devuelve /asigna contenido html de un selector
```

```
$.val() / $.val('valor')  
//devuelve/asigna el valor de un selector(input, select, etc)
```

```
$.text() / $.text('texto')  
// devuelve / setea texto (sin tags) a partir de un selector.
```

Métodos [Manipulación DOM]

- Métodos aplicables a selectores que modifican la estructura DOM a partir de un selector.

`$.append(contenido)`

`//Añade contenido al final del selector`

`$.appendTo(destino)`

`//Añade el selector al final del 'destino'`

`$.prepend(contenido)`

`//Añade contenido al principio del selector`

`$.prependto(destino)`

`//Añade el selector al principio del 'destino'`

`$.before(contenido)`

`// Añade 'contenido' antes del selector`

`$.after(contenido)`

`//Añade 'contenido' después del selector`

Métodos [Manipulación DOM]

```
$.wrap(elemento/html)
```

```
//'rodea' el elemento seleccionado con el elemento existente  
o el html especificado
```

```
$.empty()
```

```
//Vacía (pero no elimina) el selector  
// Es como $.html('');
```

```
$.remove()
```

```
//Elimina completamente del DOM el selector específico
```

- Métodos aplicables a selectores que devuelven 'familiares' DOM a partir de un selector.

`$.find(selector)`

//Devuelve un objeto jquery nuevo que contiene los elementos descendientes del selector aplicado, que además, se correspondan con el selector pasado como parámetro.

`$.children([selector])`

//Devuelve un objeto jquery nuevo que contiene todos los 'hijos' del selector. Si hubiera selector, se filtrarán los hijos se filtrarán.

`$.parent([selector])`

// Devuelve un objeto jQuery nuevo, que contiene (o no) el padre del selector, filtrado por el argumento opcional selector.

`$.parents([selector])`

//Devuelve un objeto jQuery nuevo que contiene los antecesores al selector, pudiendo ser filtrados por el argumento opcional.

Métodos [Trasversales]

`$.siblings([selector])`

//Devuelve un objeto jquery nuevo que contiene los elementos hermanos del selector aplicado, que además, se correspondan con el selector pasado como parámetro.

`$.prev([selector])`

//Devuelve un objeto jquery nuevo que contiene el elemento previo a todos los elementos del selector aplicado, que además, se correspondan con el selector pasado como parámetro.

`$.next([selector])`

//Devuelve un objeto jquery nuevo que contiene el elemento siguiente a todos los elementos del selector aplicado, que además, se correspondan con el selector pasado como parámetro.

`$.is(selector)`

//Devuelve true y false, para determinar si el selector que es invocado, coincide con el selector parámetro.

`$.each(function(i) {})`

//Recorre todos los elementos de un selector, pudiendo interactuar con ellos en la función callback, donde `$(this)` se corresponderá a cada elemento dentro de la interacción.

`$.lenght`

//Devuelve el tamaño de un selector. [0..n]
//Si es 0, el selector no está encontrando coincidencias

- Introducción
 - EVENTO: método que se activa tras la interacción del usuario o del navegador.
 - Existen diferentes implementaciones del sistema de eventos entre todos los navegadores del mercado.
 - Pero con jQuery nos da igual ;)
 - TIPOS DE EVENTOS
 - TECLADO
 - keydown / keypress / keyup
 - RATÓN
 - click / dblclick / mouseover / mousemove / mouseout
 - INTERFAZ
 - load / unload / resize / scroll / focus / blur
 - FORMULARIO
 - submit / reset / change / select

- `$(document).ready();`
 - Evento que se lanza cuando termina de cargarse el DOM, sin contar la carga de imágenes u otros elementos multimedia.
(Y no con los típicos `window.onload` o `<body onload= >`)
 - Suele considerarse como la badera de salida para empezar a ejecutar un script.

```
$(document).ready(function() {  
    alert( Hola Mundo );  
});
```

- Es importante esperar a este evento, para estar seguros de tener todos los objetos del documento perfectamente cargados (y no depender de la velocidad de descarga).

- bind

- Asigna un manejador al evento del objeto devueltos por un selector.

```
$('.clase3').bind( click ,function() {  
    alert( hola mundo );  
});
```

- Dentro de la función 'callback', tendremos disponible la variable 'this', que se refiere al objeto que habría disparado el evento.

```
(':input:text').bind( focus ,function() {  
    $(this).css( border , 1px solid red );  
});
```

- unbind
 - Desasigna un manejador al evento del objeto devueltos por un selector.
 - Soporte para namespaces:

```
$('.clase3').bind( click ,func1);  
$('.clase3').bind( click.alerta ,function() {  
    alert( hola! );  
});
```

```
$('.clase3').unbind( click ,func1);  
// desbindea sólo la func1
```

```
$('.clase3').unbind( .alerta );  
// desbindea el evento alerta
```

- trigger
 - lanza un evento determinado, para el selector escogido.

```
$('#contenido').bind('click',function(a,b) {  
    alert(a+' '+b);  
});
```

//Cuando se ejecuta normalmente (al lanzarse el evento click), a y b son valores no definidos.

```
$('#contenido').trigger('click',['Hola','mundo']);  
//En el caso de ser lanzado manualmente, pueden enviarse  
parámetros.
```

- one
 - Funciona idéntico al método bind, es decir, asignando un manejador de eventos al selector.
 - Pero esta vez es automáticamente desbindeado después de la primera ejecución.

```
$('#buton1').one('click',function() {  
    alert( solo la primera vez );  
});
```

- Introducción
 - Una animación (o un efecto) simple, puede realizarse modificando simplemente propiedades css.
 - jQuery implementa `$.animate()` para que esas trasacciones se realicen en un intervalo de tiempo.
 - Se consiguen efectos espectaculares, con muy poquito código.
 - Todo ello se base en **`$.animate()`**
 - Aunque existen métodos preconstruidos que en realidad encapsulan llamadas a `animate` ya parametrizadas.

- animate

```
$.animate(propiedades [,velocidad][,efecto][,callback]);
```

- propiedades: Mapa con propiedades CSS (numéricas).
 - width / height / left / top ...
 - siempre en pixels (ems / % pueden dar resultados raros)
 - puede especificarse **show** / **hide** / **toggle**
- velocidad: Milisegundos que se utilizan para animar (total)
 - a mayor número, menor velocidad
 - puede especificarse **fast** (200) / **normal** (400) / **slow** (600)
- efecto: Por defecto **linear**. Para usar otros, es necesario tirar de plugins.
- callback: Función invocada para cada objeto animado al final del proceso de animación

- Efectos prepaquetizados
 - `$.show([velocidad][,callback]);`
 - `$.hide([velocidad][,callback]);`
 - `$.toggle([velocidad][,callback]);`
 - modifica propiedad `display`, y llama a `callback`
 - `$.slideDown([velocidad][,callback]);`
 - `$.slideUp([velocidad][,callback]);`
 - modifica el `height`, mostrando y ocultando.
 - `$.fadeIn([velocidad][,callback]);`
 - `$.fadeOut([velocidad][,callback]);`
 - `$.fadeTo([velocidad][,opacidad][,callback]);`
 - modifica la opacidad del objeto (a 100%, a 0% o a [opacidad])

- Introducción ¿Qué es AJAX?
 - Asynchronous Javascript and XML.
 - Peticiones asíncronas de datos al servidor, realizadas desde Javascript.
 - No siempre se devuelve XML; casi siempre se trata de HTML o JSON (AJAX)
 - Nos permite construir sitios web interactivos sin necesidad de recargar el documento descargado.
 - Es posible gracias al objeto XMLHttpRequest.
 - X-Requested-With = XMLHttpRequest
 - Como siempre, implementación aleatoria en cada navegador
 - Con jQuery nos da igual :)

- \$.ajax()
 - Es la única función que jQuery tiene disponible para realizar consultas.
 - El resto (las más usadas), son helpers.

```
$.ajax({  
    url: [URL],  
    type: [GET/POST],  
    success: [function callback éxito(data)],  
    error: [function callback error],  
    complete: [function callback error],  
    ifModified: [bool comprobar E-Tag],  
    data: [mapa datos GET/POST]  
    async: [bool que indica sincronía/asincronía]});
```

- Son algunos de los parámetros más importantes y utilizados

- `$.ajaxsetup()`
 - Función que setea por defecto los parámetros para `$.ajax()`
 - Mismos argumentos que `$.ajax()`;
- Si todas nuestras consultas son a un mismo fichero y todas son POST, antes incluso del `document.ready` podemos especificar:

```
$.ajaxsetup({  
    url:'procesar.php',  
    type:'post'  
});
```

- AJAX HELPERS

- \$.get(url[,data][,exito])
- \$.post(url[,data][,exito])
- \$.getJSON(url[,data][,exito])
 - método idéntico a \$.get, que transforma a los datos recibidos por el servidor en un mapa JSON (info. estructurada).
- \$.getIfModified(url[,data][,exito])
 - Métodos idénticos a \$.ajax, con los parámetros obvios pre-seteados (get / post / ifModified)
- .load(url[,data][,exito])
- .loadIfModified(url[,data][,exito])
 - Este método, aplicado en un selector jQuery, sustituye el HTML del mismo, con lo que nos devuelva la petición.

- JSON: Javascript Object Notation

- Representación de información jerarquizada desde javascript.
- Es la alternativa más extendida al XML, quizás porque es mucho más simple (aunque no muy humano), y sobre todo es más rápido de parsear.
- Existen montón de servicios ofertados en JSON (en lugar de en XML): Google (kml), Yahoo(pipes), Flickr...
- Es la forma más rápida de servir a jQuery datos estructurados en una sólo petición.

```
{  
  clave : 'hola',  
  clave2 : ['mundo', 'luna', 'marte']  
}
```

- Ejemplo de uso de JSON

Partiendo de este JSON como respuesta de nuestro servidor:

```
{
  clave : 'hola',
  clave2 : ['mundo', 'luna', 'marte']
}
```

```
$.getJSON( data.php ,{id:$(this).attr( id )},function(j) {
  if (j.error) {
    alert( Error [ +j.error+ ] ;
    return;
  }
  $( #input1 ).val(j.clave);
  for(var idx in j.clave2) {
    $( #textareal ).append(j.clave2[idx]);
  }
});
```

- Cacheo de selectores:
 - Si tenemos un selector que vamos a utilizar concurrentemente dentro de un mismo ámbito, es mejor asignarlo a una variable declarada en ese ámbito (o encadenar métodos).

```
var obj = $( "#div ul:eq(2) em" );  
obj.css( "color", "#f00" );  
obj.bind( "click", func_accion );
```

- Tener siempre una versión accesible (aunque fea), y quitar esa funcionalidad por javascript.
- No confiar nunca la seguridad de un sitio en jQuery.
- No abusar. El javascript sirve para hacer bonito, hacer mejor la experiencia del usuario, quitar comprobaciones tontas al servidor (aunque siempre deben estar ahí).

- No utilizar detalles de css directamente desde javascript. Al igual que no lo utilizaríais desde XHTML (en la medida de lo posible).
- Amad a JSON sobre todas las cosas.
- Amad Firebug sobre todas las cosas.
- Odiad IE (y utilizar Opera para saber porque no funciona algo en IE, que suelen ser chorradas).

- Debug simple
 - Siempre ha sido complicado debuggear javascript (con algunos navegadores más que con otros). En general, en los últimos años, Firefox (gecko), es el rey en el desarrollo del lado del cliente:
 - **Firebug:**
 - Inspeccionar DOM / Debug JS (Watch / Breakpoints / etc...) / Profiling / Objeto **console** (console.debug)
 - **Livehttpheaders**
 - Permite observar exactamente las cabeceras HTTP tanto de petición como de respuesta.
 - **WebDeveloper**
 - Completa barra de Herramientas que se complementa perfectamente con las otras 2 herramientas.

- <http://www.jquery.com>
- <http://plugins.jquery.com>
- <http://ui.jquery.com/>
- <http://code.google.com/p/minify/>
 - Reduce el tamaño de tus .js y .css (y aumenta la velocidad de carga).
- <http://dean.edwards.name/packer/>
 - Optimiza (y ofusca) tus .js
- <http://www.getfirebug.com>
- <http://chrispederick.com/work/web-developer/>
- <http://livehttpheaders.mozdev.org/>

Jquery: Javascript simple, aplicaciones complejas.

Licencia Copyleft

Javier Infante Porro <jabi@irontec.com>

Copyright



- Este documento está protegido bajo la licencia Reconocimiento-SinObraDerivada 2.1 España de Creative Commons (<http://creativecommons.org/licenses/by-nd/2.1/es/>)

Copyright © 2008 Irontec <contacto@irontec.com>

Se permite la copia, modificación, distribución, uso comercial y realización de la obra, siempre y cuando se reconozca la autoría de la misma, a no sea ser que se obtenga permiso expreso del autor. El autor no permite distribuir obras derivadas a esta.

Esta nota no es la licencia completa de la obra, sino una traducción de la nota orientativa de la licencia original completa (jurídicamente válida).