

jQuery

Desarrollo de Aplicaciones Multiplataforma

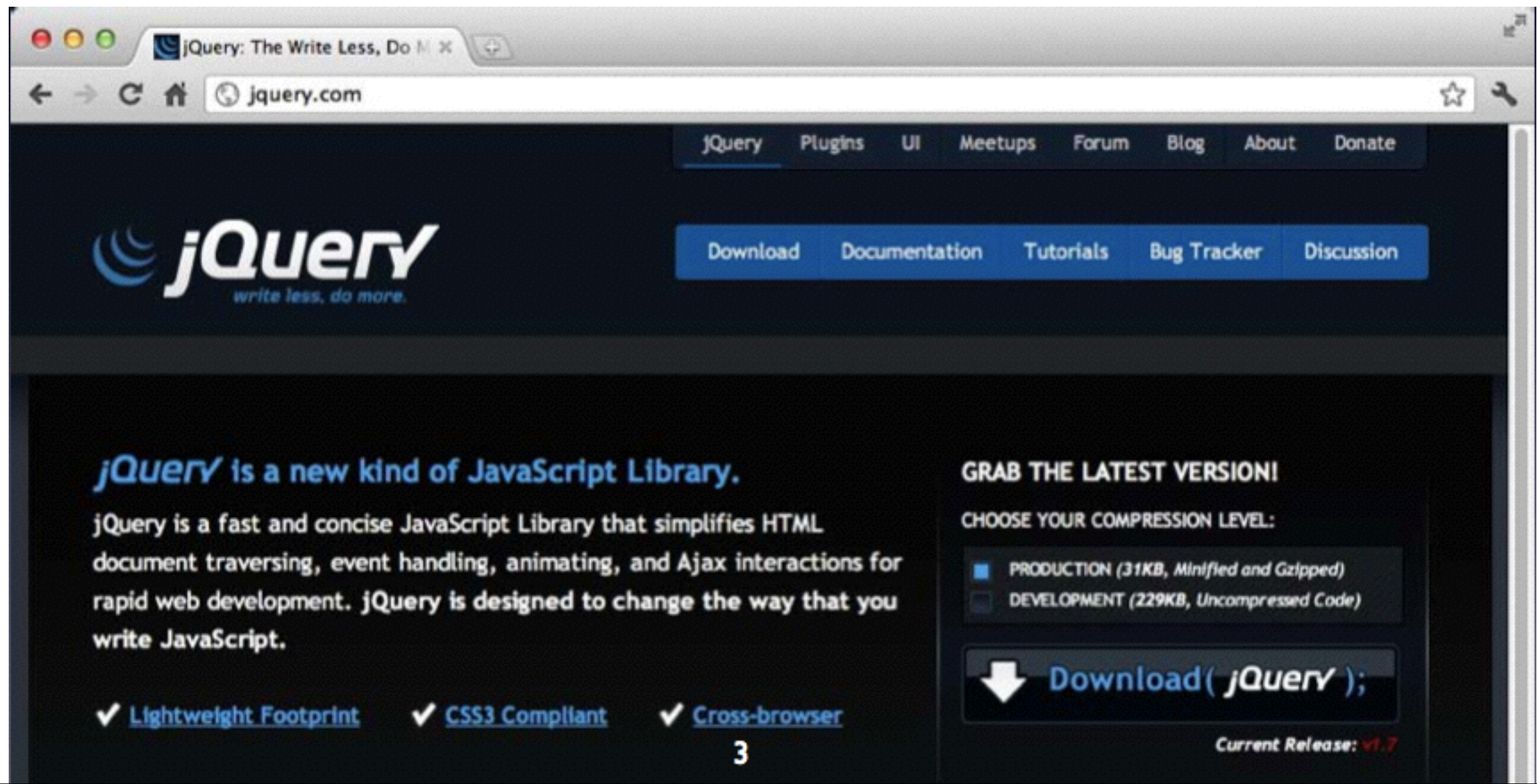


Introducción

- jQuery es un framework en JavaScript que facilita las tareas de desarrollo.
- Una de las mayores ventajas de estos frameworks radica en el hecho de que funcionan en todos los principales navegadores sin ningún esfuerzo extra.

Uso

- Podemos descargar jQuery desde jquery.com



Uso

- Como buena práctica podemos renombrar el fichero descargado como **jquery.js**, de esta forma no es dependiente de la versión y las actualizaciones no nos implican cambiar etiquetas *<script>*.

```
<script src="jquery.js"></script>
```

Uso

Podemos
descargar una
versión de
desarrollado que
nos permite ver
cómo está
implementado
jQuery.

```
/*!
 * jQuery JavaScript Library v1.7
 * http://jquery.com/
 *
 * Copyright 2011, John Resig
 * Dual licensed under the MIT or GPL Version 2 licenses.
 * http://jquery.org/license
 *
 * Includes Sizzle.js
 * http://sizzlejs.com/
 * Copyright 2011, The Dojo Foundation
 * Released under the MIT, BSD, and GPL Licenses.
 *
 * Date: Thu Nov 3 16:18:21 2011 -0400
 */
(function( window, undefined ) {

// Use the correct document accordingly with window argument (sandbox)
var document = window.document,
    navigator = window.navigator,
    location = window.location;
var jQuery = (function() {

// Define a local copy of jQuery
var jQuery = function( selector, context ) {
    // The jQuery object is actually just the init constructor 'enhanced'
    return new jQuery.fn.init( selector, context, rootjQuery );
},

// Map over jQuery in case of overwrite
_jQuery = window.jQuery,

// Map over the $ in case of overwrite
_$ = window.$,

// A central reference to the root jQuery(document)
rootjQuery,

// A simple way to check for HTML strings or ID strings
// Prioritize #id over <tag> to avoid XSS via location.hash (#9521)
quickExpr = /^(?:[^\#<]*(<[\w\W]+>)[^\>]*$|#[^\w\-\s]*$)/,

// Check if a string has a non-whitespace character in it
rnotwhite = /\S/,

// Used for trimming whitespace
trimLeft = /^\s+/,
trimRight = /\s+$/;
```


Uso

Además, jQuery proporciona una versión de producción o minimizada. Espacios y saltos de línea han sido eliminados, identificadores de variables y funciones se reducen al mínimo.

```
/*! jQuery v1.7 jquery.com | jquery.org/license */
(function(a,b){function cA(a){return f.isWindow(a)?a:a.nodeType===9?a.defaultView||a.parentWindow:!1}function cx(a){if(!cm[a])
{var b=c.body,d=f("<"+a+">").appendTo(b),e=d.css("display");d.remove();if(e==="none"||e===""){cn||
(cn=c.createElement("iframe"),cn.frameBorder=cn.width=cn.height=0),b.appendChild(cn);if(!co||!
cn.createElement)co=(cn.contentWindow||cn.contentDocument).document,co.write((c.compatMode==="CSS1Compat"?"<!doctype html>":"")
+"<html><body>"),co.close();d=co.createElement(a),co.body.appendChild(d),e=f.css(d,"display"),b.removeChild(cn)}cm[a]=e}return
cm[a]}function cw(a,b){var c={};f.each(cs.concat.apply([],cs.slice(0,b)),function(){c[this]=a});return c}function cv(){ct=b}
function cu(){setTimeout(cv,0);return ct=f.now()}function cl(){try{return new a.ActiveXObject("Microsoft.XMLHTTP")}catch(b){}}
function ck(){try{return new a.XMLHttpRequest}catch(b){}}function ce(a,c){a.dataFilter&&(c=a.dataFilter(c,a.dataType));var
d=a.dataTypes,e={},g,h,i=d.length,j,k=d[0],l,m,n,o,p;for(g=1;g<i;g++){if(g===1)for(h in a.converters)typeof
h=="string"&&(e[h.toLowerCase()]=a.converters[h]);l=k,k=d[g];if(k==="*")k=l;else if(l!=="*"&&l!==k){m=l+" "+k,n=e[m]||e["*
"+k];if(!n){p=b;for(o in e){j=o.split(" ");if(j[0]===l||j[0]===k){p=e[j[1]+" "+k];if(p){o=e[o],o===!0?n=p:p===!
0&&(n=o);break}}}}!n&&!p&&f.error("No conversion from "+m.replace(" "," to "),n,!0&&(c=n?n(c):p(o(c))))}return c}function
cd(a,c,d){var e=a.contents,f=a.dataTypes,g=a.responseFields,h,i,j,k;for(i in g)i in
d&&(c[g[i]]=d[i]);while(f[0]==="*")f.shift(),h===b&&(h=a.mimeType||c.getResponseHeader("content-type"));if(h)for(i in
e)if(e[i]&&e[i].test(h)){f.unshift(i);break}if(f[0]in d)j=f[0];else{for(i in d){if(!f[0]||a.converters[i+" "+f[0]]){j=i;break}
k||(k=i)}j=j||k}if(j){j!==f[0]&&f.unshift(j);return d[j]}function cc(a,b,c,d){if(f.isArray(b))f.each(b,function(b,e){c||
bG.test(a)?d(a,e):cc(a+"["+typeof e=="object"||f.isArray(e)?b:"")+"]",e,c,d)});else if(!c&&b!=null&&typeof b=="object")for(var
```

Uso

- Otra opción consiste en enlazar en caliente la librería directamente al repositorio sin descargarla.
- Siempre es preferible no depender de servidores de terceros.

```
<script src="http://code.jquery.com/jquery-1.7.min.js"></script>
```

Plugins

- Existen numerosos plugins (ficheros extra que puedes utilizar junto con jQuery) para añadir funcionalidad. Ej: validación de correo, características de UI, etc.



```
<script src="jquery-1.4.2.min.js"></script>
```

```
<script src="jquery.easing.1.3.js"></script>
```


jQuery UI

- También existe una librería de funciones UI (Interfaz de usuario) disponible en jquery.com.
- Estas funciones pueden ser bastante largas, y jQuery te permite escoger aquellas que necesitas y personalizar así tu fichero.

jQuery UI



[Demos](#)[Download](#)[API Documentation](#)[Themes](#)[Development](#)[Support](#)[Blog](#)[About](#)

Download Builder

Quick downloads: [Stable \(Themes\)](#) (1.10.4: for jQuery1.6+) | [Legacy \(Themes\)](#) (1.9.2: for jQuery1.6+)
[All jQuery UI Downloads](#)

Version

☒ **1.10.4** (Stable, for jQuery1.6+)

☐ **1.9.2** (Legacy, for jQuery1.6+)

Components

☒ Toggle All

UI Core <input checked="" type="checkbox"/> Toggle All A required dependency, contains basic functions and initializers.	<input checked="" type="checkbox"/> Core <input checked="" type="checkbox"/> Widget <input checked="" type="checkbox"/> Mouse <input checked="" type="checkbox"/> Position	The core of jQuery UI, required for all interactions and widgets. Provides a factory for creating stateful widgets with a common API. Abstracts mouse-based interactions to assist in creating certain widgets. Positions elements relative to other elements.
---	---	---

Objeto jQuery

- jQuery nos proporciona un objeto global:

jQuery

- Este objeto tiene un alias que es el que normalmente se usa:

\$

Objeto jQuery

jQuery(...)

puede escribirse también como

\$(...)

Selectores

- A través del objeto jQuery podemos buscar elementos en nuestro documento a través de selectores CSS.

`$("div.normal")`

`$("#uno")`

`$("tr:even")`

- Esto nos retorna colecciones de objetos jQuery para aquellos elementos que cumplen el selector. Estos objetos no son compatibles con objetos del DOM.

Objetos del DOM

- Si se prefiere trabajar con objetos del DOM en vez de utilizar los objetos jQuery, podemos utilizar la función ***get()***.

```
var domArray = $("div.normal").get();
```

```
var domEl = $("div.normal").get(0);
```

Selectores CSS3

jQuery acepta selectores CSS3 aun cuando en navegador no los tiene implementados.

Funciones

- La mayoría de las funciones de jQuery se aplican a todos los elementos retornados por el selector.
- El mensaje que envías al objeto se envía automáticamente a todos los elementos de la colección.
- Ej: el siguiente código oculta todo párrafo:

```
$("#p").hide();
```

Encadenado

- Se pueden especificar varias acciones a aplicar sobre los elementos.

```
$("div").show().slideUp();
```

- Las dos funciones son ejecutadas en secuencia. **slideUp()** se ejecuta una vez que **show()** ha finalizado.

Duración

- Para algunas de estas funciones podemos establecer el tiempo de duración.

```
$("#div").show().slideUp(500);
```

- El efecto **slideUp** durará medio segundo (500 milisegundos).

Pausa

- Incluso podemos definir pausas dentro de la secuencia de ejecución.

```
$("#div").show().delay(1000).slideUp();
```

- Se realiza una pausa durante 1 segundo.

`$(this)`

- Para la mayor parte de jQuery el equivalente a ***this*** es:

\$(this)

Bucle

- Se puede definir una función para que se aplica a cada uno de los objetos de la colección mediante el uso de ***each()***

```
$("#div").each(function(){ ... });
```

`$(this)`

- En el caso de los manejadores de eventos y bucles, **`$(this)`** hace referencia al elemento sobre el cuál se está aplicando la función.

```
$("#div").each(function() {  
    $(this).html("");  
});
```

`$(this)`

- **`$(this)`** hace referencia al elemento de la iteración en curso.
- Se puede incluso pasar como parámetro el índice del elemento dentro de la colección.

```
$("#div").each(function(index) {
```

```
    $(this).html("número: " + index);
```

```
});
```


`$(this)`

- Alternativamente, se puede hacer que jQuery pase a la función el elemento sobre el que se está iterando.

```
$("#div").each(function(index, element) {  
    $(element).html("número: " + index)  
});
```

Atributos

- Se puede cambiar un atributo de un elemento a través de la función ***attr()***:

```
$("#imagen").attr("src", "arbol.jpg");
```

- También se pueden leer atributos:

```
var src = $("#imagen").attr("src");
```

- Cuando se invoca la función sobre una colección, el atributo que se retorna es el del primer elemento de la colección.

Contenido

- Se puede añadir HTML a un elemento, de la misma manera que con *innerHTML* mediante la función ***html()***.

```
$(“div#principal”).html("Hola, ¿qué tal?");
```

Contenido

- A través de esta función también podemos leer el contenido HTML del elemento.

```
var html = $("div#principal").html();
```

- Si la colección tiene más de un elemento, el HTML devuelto pertenece al primer elemento de la misma.

Contenido

- También podemos rescatar el texto (sin etiquetas HTML) de los elementos de la colección.

```
var texto = $("div").text();
```

- Existe la posibilidad de añadir texto:

```
$("div#contenido").text("Bienvenido");
```


Contenido

- Si el texto incluye etiquetas HTML, éstas se escapan.

```
$("div#contenido").text("<p>Texto</p>");
```

- Esta expresión añade el siguiente texto al div cuyo id es 'contenido':

```
&lt;p&gt;Texto&lt;/p&gt;
```

Contenido

- Para rescatar o cambiar el valor de un campo en un formulario tenemos la función ***val()***.

```
var nombre = $("input#nombre").val();
```

Contenido

- jQuery puede crear y parsear elementos HTML que después pueden incorporarse en cualquier parte del documento.

```
$( "<h2>email</h2>" ).insertBefore( "div.email" );
```

```
$( "<h2>email</h2>" ).insertAfter( "div.email" );
```

Contenido

```
$( "<h2>email</h2>" ).insertBefore( "div.email" );
```

equivale a

```
$( "div.email" ).before( "<h2>Email</h2>" );
```

Se inserta el código HTML antes del div con clase 'email'.

Animación

- Se puede hacer que los elementos aparezcan o desaparezcan mediante efectos de animación.
 - ***show() / hide()***
 - ***fadeIn() / fadeOut()***
 - ***slideDown() / slideUp()***

Animación

- Se puede especificar en milisegundos el tiempo que queremos que estos efectos duren.

```
$("#id").fadeIn(500);
```

```
$("#div").hide().fadeIn(500).fadeOut(500);
```

Retorno

- Muchas de estas funciones permiten especificar una función a ejecutar al finalizar la ejecución.
- Estas funciones se llaman ***callbacks***.

```
$("#id2").fadeOut(500, function () {  
    $(this).html("");  
});
```

Retorno

- La expresión anterior es equivalente a:

```
function accion () {
```

```
    $(this).html("");
```

```
}
```

```
$("#id2").fadeOut(500, accion);
```


CSS

- Si queremos cambiar el estilo CSS de un elemento podemos especificar los estilos dentro de un objeto que pasamos a la función **css()**.

```
$("#div").css({
```

```
  "border": "1px solid red",
```

```
  "background": "#ddd"
```

```
});
```

Animación CSS

- Se pueden realizar animaciones especificando los estilos CSS que se desean aplicar el tiempo de duración que deseamos que tenga la transición.
- Se puede especificar también el easing, progreso, y una función de retorno, callback.

Animación CSS

```
$(this).animate(  
    {  
        "marginTop": "50px" ,  
        "marginLeft": "10px"  
    } ,  
    1000);
```

- Las propiedades CSS deben pasarse dentro de un objeto.

Easing

- Se puede especificar cómo queremos que se ejecute la animación.
- Existen opciones simples de easing en la distribución estándar de jQuery. Ej: easeIn, linear, etc.
- Podemos descargar plugins extra para añadir más efectos de easing:
 - <http://gsgd.co.uk/sandbox/jquery/easing/>

Easing

- El efecto easing lo especificamos en la llamada a la función.

```
$(".menu").animate (  
    {"marginTop": "50px"}  
    , 1000  
    , "easeOutBounce");
```

```
$(".menu").slideDown(400, "easeOutBounce");
```

Easing

- El ejemplo anterior añade un rebote al final de la animación.
- Muchas de las funciones de animación permiten especificar una opción de easing.
- Además, podemos especificar una opción de easing por defecto para todas las animaciones.

```
jQuery.easing.def = "easeOutBounce";
```

Manipulando la cola

- Consideremos el siguiente ejemplo:

```
<ul class = "menu">
```

```
<li>One</li>
```

```
<li>Two</li>
```

```
<li>Three</li>
```

```
</ul>
```

One
Two
Three

Manipulando la cola

- Queremos realizar el siguiente efecto. Cuando el ratón pase por encima de un elemento queremos que se desplace ligeramente a la derecha, y que vuelva a su posición cuando el ratón ya no esté por encima del elemento.



Manipulando la cola

```
$("#ul.menu li").hover(  
    function () {  
        $(this).animate({ "marginLeft": "30px"}, 500);  
    }  
,  
    function () {  
        $(this).animate({ "marginLeft": "0px"}, 500);  
    });
```

Manipulando la cola

- En el ejemplo anterior especificamos a la función ***hover()*** que acción realizar cuando el cursor está por encima del elemento (primera función especificada) y qué hacer cuando ya no esté sobre él (segunda función).
- Estas funciones se añaden a la cola y se ejecutan en orden. Si nos hemos posicionado sobre el elemento y salimos rápidamente veremos que la acción de entrar se ejecuta hasta el final antes de ejecutarse la acción de salida.

Manipulando la cola

- Mediante el uso de la función **stop()** podemos parar la animación en curso.

```
$("ul.menu li").hover(  
    function () {  
        $(this).stop().animate({ "marginLeft": "30px"}, 500);  
    }  
,  
    function () {  
        $(this).stop().animate({ "marginLeft": "0px"}, 500);  
    }  
);
```

Eventos

- Los eventos se asignan sencillamente:

```
$("#b1").click(function(){ ... });
```

- O alternativamente:

```
function fun1 () { ... }
```

```
$("#b1").click (fun1);
```

Eventos

- En los manejadores de eventos `$(this)` hace referencia al objeto del DOM que lanza el evento.
- Algunos eventos son:
 - `.focus()`
 - `.focusin()`
 - `.focusout()`
 - `.hover()`
 - `.keydown()`
 - `.keypress()`
 - `.keyup()`
 - `.load()`

Eventos

- `.mousedown()`
- `.mouseenter()`
- `.mouseleave()`
- `.mousemove()`
- `.mouseout()`
- `.mouseover()`
- `.mouseup()`

Ready

- Este evento está vinculado al objeto jQuery y se lanza cuando el DOM está ya cargado (pero no las imágenes, ficheros, etc)

function init () { ... };

\$.ready(init)

\$(init)

\$(function(){ ... });

Toggle

- Este evento permite comportamientos diferentes para el click de un elemento.
- Recibe varias funciones que se ejecutarán en orden en función del número de click.
- La primera pulsación ejecutara la primera función, la segunda pulsación ejecutará la segunda función, etc.

`$('.box').toggle (function1, function2);`

Toggle

```
$('.box').toggle (  
    function () {  
        $('#b1').hide()  
    },  
    function () {  
        $('#b1').show()  
    }  
);
```

Toggle

- Se pueden pasar como parámetros tantas funciones como queramos y estas se ejecutarán en ciclo.

\$('.box').toggle (

function1,

function2,

function3

);

Toggle

- Se pueda usar también sin parámetros para ocultar / mostrar un elemento.

```
$(".box").click ( function(){
```

```
$(this).toggle();
```

```
})
```

Hover

- Como en ***toggle***, se especifican 2 funciones que se ejecutan cuando el cursor se posiciona dentro de un elemento, y la segunda para cuando el cursor sale fuera del elemento.

\$('.box').hover (function1, function2);

Eventos de objetos

- Ejemplo:

```
<a id = "ocultar" href = "" >Pulsa para ocultar</a>
```

```
$("#a#ocultar").click (
```

```
function (event) {
```

```
    event.preventDefault();
```

```
    $("#div.articulo").hide();
```

```
}
```

```
);
```

Eventos de objetos

- Como vemos en el ejemplo, hemos sobrecargado el evento click del enlace.
- Para prevenir que el enlace siga su proceso de redirección hemos invocado:

event.preventDefault();

- De esta forma evitamos el comportamiento por defecto del evento.

Navegación

- Nos podemos mover a través del DOM mediante funciones de jQuery.
- Estas funciones pueden recibir incluso selectores para refinar la navegación.
- Podemos añadir o eliminar elementos.

Navegación

- .parent()
- .children()
- .siblings()
- .add()
- etc

Navegación

`<p>Test</p>`

`<div>`

`<p id = "uno" >Uno</p>`

`<p id = "dos" >Dos</p>`

`</div>`

Navegación

- `$("#uno").parent().children("p").not(":first").hide();`
- jQuery buscar el padre de **<p id = “uno”>**, después recupera sus hijos que sean párrafos, se excluye el primer hijo y por último oculta los elementos.
- Podemos obtener el mismo efecto con:

`$("#uno").siblings().hide();`