



Librería de JavaScript para acceder a los objetos del DOM de un modo simplificado.

<http://jquery.com/>

Para más info ir a:

<https://www.youtube.com/playlist?list=PL9ADAF257242D75FA>

Ó:

<http://programando.la/>


```

<!DOCTYPE html>

<html lang="es">

  <head>

    <meta charset="utf-8"/>

    <title>Pág. Con JQuery</title>

    <!--Scripts Aca -->

    <script                                     type="text/javascript"
    src="http://code.jquery.com/jquery-1.11.0.min.js">

    </script>

    <script type="text/javascript">
      $(document).on("ready",function(){
        $("#prueba").text('<p>Hola Mundo!</p>');
        $(".prueba").html('<strong>Estoy          usando
        JQuery</strong>');
      });
    </script>

  </head>

  <body>

    <div id="prueba" ></div>

    <div class="prueba" ></div>

  </body>

</html>

```

- La función principal de la librería se llama \$.
- Se accede a los elementos del dom mediante id (#) ó clase (.) CSS.
- On('ready'... → es cuando la pág. ya se ha cargado.

.text() → muestra el texto con todo y <p>

.html() → cambia el contenido HTML, no muestra el

SELECTORES:

```
<!DOCTYPE html>
```

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

```
  <head>
```

```
    <meta charset="utf-8"/>
```

```
    <title>Ejercicio Selectores</title>
```

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

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

```
      $(document).on("ready",function(){
```

```
        $("div.cajas article p").text("Selecciona el Párrafo Uno");
```

```
        $("#p1, #p2, span").text("Selecciona los 3 elementos");
```

```
      });
```

```
    </script>
```

```
  </head>
```

```
  <body>
```

```
    <div class="cajas" >
```

```
      <article>
```

```
        <p id="p1">Párrafo Uno</p>
```

```
      </article>
```

```
    </div>
```

```
    <div>
```

```
      <article>
```

```
        <p id="p2">Párrafo Dos</p>
```

```
      </article>
```

```
    </div>
```



↑
↑

 </body>
</html>

Ej: Verifica si lo que se seleccionó existe o no, y ese resultado se ve en **consola del navegador**.

```
var seleccion; → es global
```

```
$(document).on("ready",function(){  
    seleccion =$(".ejem1");  
    if seleccion.length  
        console.log("Existen: "+selección.length);  
    else  
        console.log("No existe el elemento seleccionado");
```

// .not() → Selecciona los que no tenga

```
seleccion.not("cl1").text("Este elemento no tiene la clase cl1");
```

// .has() → Selecciona los que tengan

```
seleccion.has("p").text("Este elemento tiene un <p> en su contenido");
```

// .first() → Selecciona el primer elemento

```
$(".li").first().html("El primer elemento");
```

// .eq() → Selecciona el elemento indicado, recordar que se enumeran desde el 0 como un array

```
$(".li").eq(2).text("Soy el elemento #3");  
});
```

```
<div class="ejem1">  
    <p></p>  
</div>
```

```
<div class="ejem1 cl1">  
    <span></span>  
</div>
```

```
<ul>  
    <li></li>  
    <li></li>  
    <li></li>  
    <li></li>  
</ul>
```

GET Y SET:

//Obtener el contenido html

```
var contenido=$("#parrafo").html();  
alert(contenido);
```

//Este método sirve para obtener (GET) el id

```
var id=$("#prueba").attr("id");  
alert(id);
```

//Este método asigna (SET) un valor

```
$("#prueba").attr("class","ejemplo");
```

```
<div id="prueba"></div>
```

```
<p id="parrafo"><strong>Hola</strong></p>
```

LOADER:

\$(document).ready → se activa cuando carga todo el Dom, incluso aunque las imágenes no terminen de cargar.

\$(window).load → se activa cuando la página termine de cargar todas las peticiones.

Ej:

<!DOCTYPE html>...

<head>

<style type="text/css">

#pre-load-web{width:100%;

position:fixed;

background:#92def8;

left:50%;

top:50%;

z-index:100000;

}

/*aquí centramos la imagen si coloco margin left -30 es porque la imagen mide 60 */

#pre-load-web #imagen-load{top:50%;

left:50%;

margin-left:-30px;

position:fixed;

}

</style>

<script>

var ...; //variables globales

function ...(){} //otras funciones JavaScript o JQuery

\$(document).on("ready",function(){

//eliminamos el scroll de la pagina

\$("body").css({"overflow-y":"hidden"});

//guardamos en una variable el alto del que tiene tu browser que no es lo mismo que del DOM



```

    var alto=$(window).height();

    //agregamos en el body un div que ocupe toda la pantalla y se
    muestra encima de todo

    $("body").append('<div id="pre-load-web"><div id="imagen-load">
    </div></div>');

    //le damos el alto

    $("#pre-load-web").css({"height":"alto"});

    //esta será la capa que está dentro de la capa que muestra un gif

    $("#imagen-load").css({"margin-top":(alto/2)-30});

    //llamado a funciones que se deben ir cargando

    });

    $(window).on("load",function(){

        $("#pre-load-web").fadeOut(1000,function(){

            //eliminamos la capa de precarga

            $(this).remove();

            //permitimos scroll

            $("body").css({"overflow-y":"auto"});

        });

        $("canvas").css({"visibility":"visible"});

    })

</script>

</head>

<body>

    <canvas id="canvas" width="1200" height="675" style="background-
    color:#000000;visibility:hidden;"></canvas>

</body></html>

```


MÉTODO CSS:

`$("body,html").css('background','#FFF');`

- ➔ Cambia una sola propiedad css, en este caso cambia la propiedad background asignandole el color #FFF.
 - ➔ **"body,html"**: sirve para darle estilo al body, se pone así para que con algunas propiedades css, Firefox las reconozca.
-

- ➔ Objeto JSON.

```
var cssH1={  
    'text-align':'center',  
    'color':'red',  
    'font-family':'Helvetica'  
}
```

`$("h1").css(cssH1);`

- ➔ Modifica varias propiedades.
- ➔ También se puede usar directamente:

```
$("h1").css({  
    _ _ _ _ _ _ _ _ _ _  
    _ _ _ _ _ _ _ _ _ _  
    _ _ _ _ _ _ _ _ _ _  
});
```

`var bgItem2=$("#item2").css('background-color');`

- ➔ Obtiene el valor de la propiedad background-color.
-

CLASES CSS:

`.addClass("class");` ➔ añade una clase de css al elemento.

`.removeClass("class");` ➔ quita la clase al elemento.

.toggleClass("clase"); → si el elemento tiene la clase, se la quita, si no la tiene, se la agrega.

.hasClass("clase"); → pregunta si el elemento tiene o no la clase, si la tiene devuelve true.

```
$(document).on("ready",function(){  
    $(document).on("click",function(){  
        //$("#test").addClass("test");  
        //$("#test").removeClass("test");  
        $("#test").toggleClass("test");  
        if($("#test").hasClass("test")){  
            $("#test").text("Hola ahora es rojo");  
        }  
        else{  
            $("#test").text("Ahora es normal");  
        }  
    });  
});
```

```
<style type="text/css">  
    .test{  
        background-color:red;  
    }  
</style>
```

```
<div id="test" class="test">Haz clic aquí</div>
```

Nota: Ver el ejercicio y video número 13.

DIMENSIONES:

.width() → saca el ancho del elemento (puede ser una imagen), sin unidades (px,em,%,etc.).

.height() → saca el alto del elemento.

.height(100) → redimensiona el elemento 100px, el ancho se auto-redimensiona, tambien se le puede indicar un string para otras unidades (em, %).

.css('height') → este si indica el alto del elemento con unidades.

.position() → saca las coordenadas (x,y) (left,top) del elemento, sin contar padding, border, o margin de los elementos que lo contengan o sean superiores.

```
var pos=$("#parrafo").position();
```

```
pos.left; //Posición left
```

```
pos.top; //Posición top
```

.offset() → indica las coordenadas (left,top), relativas al documento, es decir, si se tienen en cuenta el padding, border, margin de los elementos que lo contengan o sean superiores.

MÉTODO ATTR:

.attr("nombreAtributo"); → obtiene el valor del atributo indicado.

```
<div id="prueba" class="pruebaClase"></div>
```

```
var valor=$("#prueba").attr("class");
```

```
//valor=pruebaClase
```

.attr("nombreAtributo","valor"); → agrega el atributo con el valor indicado.

```
//le agrega un atributo inventado data-lightbox="roadtrip"
```

```
$('#prueba').attr("data-lightbox","roadtrip");
```

.removeAttr("atributo"); → elimina el atributo del element.

Ejemplo de uso: indicarle al usuario que está saliendo de la pág. al hacer clic en un enlace:

```
<a href="http://programando.la/">Enlace a ProgramandoLA</a>
```

```

$('a').on("click",function(evento){

    evento.preventDefault();

    var link=$(this).attr('href');

    alert("Usted esta saliendo de esta pág. web");

    location.href=link;

});

```

CREAR, COPIAR Y REMOVER ELEMENTOS:

Nota: ejercicio 16, parecido al ejercicio DOM de JavaScript, pero este es con JQuery.

.append() → agrega un hijo al final.

```

<h2>Saludos</h2>

<div class="caja">
    <div class="interior">Hola</div>
    <div class="interior">Adios</div>
</div>

```

//agregamos un Nuevo párrafo:

```

$($(".interior").append("<p>Test</p>");

```

//así queda la pág.:

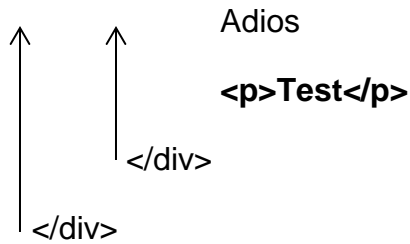
```

<h2>Saludos</h2>

<div class="caja">
    <div class="interior">
        Hola
        <p>Test</p>
    </div>
    <div class="interior">

```





También sirve para mover un elemento y colocarlo como hijo de otro:

```
$(".caja").append($(".h2"));
```

```
<div class="caja">
  <div class="interior">Hola</div>
  <div class="interior">Adios</div>
  <h2>Saludos</h2>
</div>
```

La sintaxis de `.append()` es:

```
$("#destino").append($("#origen"));
```

.appendTo() → agrega un hijo al final, igual que `append`, pero se cambia el orden de origen, destino.

```
$("#origen").appendTo($("#destino"));
```

.prepend(),prependTo() → agregan hijos al principio.

```
$(".caja").prepend($(".h2"));
```

```
<div class="caja">
  <h2>Saludos</h2>
  ...
</div>
```

```
$("#origen").prependTo($("#destino"));
```

.insertBefore(),before() → agregan antes del elemento indicado.

```
$("#destino").before($("#origen"));
```

```
$(".caja").before("<p>Párrafo Nuevo</p>");
```

<p>Párrafo Nuevo</p>

```
<div class="caja">
```

...

```
</div>
```

.insertAfter(),after() → agregan después del elemento indicado.

.clone() → hace una copia ("clon") de un elemento.

```
var clon=$(".caja").clone();
```

```
$(".caja").after(clon);
```

```
<div class="caja">
```

...

```
</div>
```

```
<div class="caja">
```

...

```
</div>
```

Nota: toca tener cuidado de clonar un elemento con id, ya que el id debe ser único y se pueden generar problemas en la pág.

.remove() → elimina un elemento.

```
$(".caja").remove();
```

→ Elimina el div con clase="caja" y sus hijos.

RECORRER DOCUMENTO:

.next() → obtiene el siguiente elemento.

```
<ul>
```

```
<li id="primero">Primero</li>
```

```
<li>Segundo</li>
```



```

      ↑
      <li>Tercero</li>
      <li>Cuarto</li>
    </ul>

```

```
console.log($('#primero').next().text());
```

```
//imprime: "Segundo"
```

.parent() → obtenemos al padre del elemento.

```

    <article id="padre">
      Algo
      <div id="hijo">
        hijo del article
      </div>
    </article>

```

```
console.log($('#hijo').parent().text());
```

```
//imprime: "Algo hijo del article"
```

.children() → obtenemos el hijo o hijos (devuelve un array) del elemento.

```
console.log($('#padre').children().text());
```

```
//imprime: "hijo del article"
```

.siblings() → obtenemos él o los hermanos (devuelve un array) del elemento.

```
console.log($('#primero').siblings().text());
```

```
//imprime: "SegundoTerceroCuarto"
```

.each() → nos sirve como el ciclo for each.

```

$("ul li").each(function(indice,elemento){
    console.log("El elemento no. "+indice+" contiene este texto: "
    +$(elemento).text());
});

```

MÉTODOS UTILITARIOS:

Son métodos útiles que se encuentran en el núcleo de JQuery.

```
var arreglo=["Primero","Segundo","Tercero"];
```

//Recorre el array:

```
$.each(arreglo,function(índice,valor){  
    console.log("En la posición: "+índice+" hay este contenido: "  
    +valor);  
});
```

//Busca si se encuentra en el arreglo, devuelve la posición (índice) si lo encuentra, sino devuelve -1.

```
var encontro=$.inArray("Primero",arreglo);  
  
if(encontro != -1){  
    console.log(arreglo[encontro]);  
}
```

//Elimina espacios al principio y al final de un string:

```
var mensaje="          al principio hay un espacio grande y al final  
tambien          ";
```

```
console.log($.trim(mensaje));
```

Existen muchos otros métodos utilitarios, para más info, ver el api y buscar:

```
jQuery.nombreMetodoBusca()
```

MÉTODO DATA:

Nos permite adjuntar datos de cualquier tipo de elementos DOM, de forma que estén a salvo de las referencias circulares y de perdidas de memoria.

Estos datos se eliminan cuando se eliminan los objetos DOM (a través de métodos de JQuery) o cuando el usuario abandona la pág.

Este método se usa para aplicaciones grandes, pero aún no es soportado por algunos navegadores.

//Para agregar el atributo (SET)

```
$.data(elemento,'nombreAtributo',valor);
```

Ej:

```
$( "ul li" ).each(function(indice,elemento){  
    $.data(elemento,'posicion',indice);  
});
```

//Para ver el atributo (Get)

```
$.data(elemento,'nombreAtributo');
```

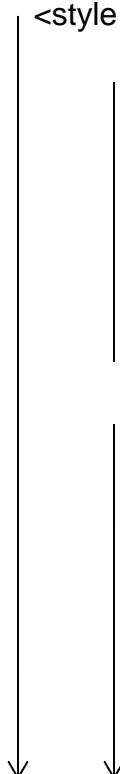
Ej:

```
$( "ul li" ).each(function(indice,elemento){  
    console.log($.data(elemento,'posicion'));  
});
```

Nota: los atributos agregados a través de \$.data, no se pueden ver con el inspector de elementos del navegador.

VINCULAR Y DESVINCULAR EVENTOS:

```
<style type="text/css">  
    body,html{  
        background:#FFF;  
        margin:0;  
        padding:0;  
    }  
  
    #caja1{  
        background:#00ff00;  
        display:block;  
        height:100px;  
        margin:20px auto;
```




```
↑
    if(contador >= 3){
        $('#caja1').off("mouseover",f1);
    }
};
```

```
var f3=function(){
    console.log("Otra funcion");
};
```

```
var f4=function(){
    console.log("Una vez");
};
```

```
$(document).on("ready",function(){

    //vinculamos varios eventos con una sola selección

    $('#caja1').on({
        "mouseover":f1,
        "mouseout":f2
    });

    //vinculamos otra vez el evento mouseover, pero para que imprima por
    consola

    $('#caja1').on("mouseover",f3);

    //para vincular un evento que se ejecuta solo una vez

    $('#caja1').one("click",f4);

});
```

```
<div id="caja1">
    Texto
</div>
```

EVENTOS PERSONALIZADOS:

//En este caso el evento personalizado se llama cambio.

```
$('#milInput').on("cambio",function(evento,id){
    console.log(id);
    if(this.checked){
        $('#milInput').prop("checked",null);
        $('.cambiar').text("Encendido");
    }
    else{
        $('#milInput').prop("checked",true);
        $('.cambiar').text("Apagado");
    }
});
```

//disparamos el evento cuando se hace clic con trigger

```
$('.cambiar').on("click",function(){
    var id=$(this).attr("id");
    $('#milInput').trigger("cambio",[id]);
});
```

```
<input id="milInput" type="checkbox" checked="checked">
<button id="btn1" class="cambiar">Apagado</button>
<button id="btn2" class="cambiar">Apagado</button>
```

.prop() → obtiene/cambia el valor de una propiedad.

.trigger() → dispara la ejecución de un evento.

EFFECTOS:

.show() → muestra el elemento.

.hide() → esconde el elemento.

.fadeIn() → muestra el elemento, aumentando poco a poco su opacidad.

.fadeOut() → oculta el elemento desvaneciéndolo.

.fadeToggle() → si está oculto, muestra el elemento, sino lo oculta.

Nota: hay más efectos, ver el API.

Como parámetro de los efectos podemos enviar la velocidad de duración del efecto: “slow”, “fast”, o indicar en milisegundos (100, 2000 → 2seg, etc.).

También podemos crear una velocidad personalizada:

```
$fx.speeds.lentooo=3000;
```

```
$fx.speeds.muyRapido=100;
```

```
.fadeToggle("lentoo",function(){
```

```
    //Funcion que se ejecuta después de la animación. También llamada  
    "callback"
```

```
});
```

EFFECTOS PERSONALIZADOS CON ANIMATE:

.animate() → Nos permite modificar propiedades CSS de un elemento para animarlo.

Como 1er parámetro se le pasan las propiedades CSS que se quieren modificar en formato JSON.

Como 2do parámetro se le indica la velocidad de la animación.

Como 3er parámetro, se le puede indicar la función “callback”, que se ejecuta después de hacer la animación.

Nota: con .animate(), no se pueden animar propiedades de color, como: backgroundColor, color, borderColor, etc. Para poderlas animar, se necesita añadir un plugin ó utilizar JQuery UI.

Plugin para animar colores, con el método animate de JQuery:

<http://www.bitstorm.org/jquery/color-animation/>

```
<style type="text/css">
    body,html{
        background:#FFF;
        margin:0;
        padding:0;
    }
    div{
        display:block;
        margin:0 auto;
        height:100px;
        width:400px;
    }
    #caja1{
        background:#00ff00;
        margin-bottom:100px;
    }
    #caja2{
        background:#871f78;
    }
</style>
```

//incluyendo el plugin:

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

```
$('#caja1').animate({
    marginBottom:0,
},
5000,
function(){
    $("#caja1").animate({
        marginTop:100,
        opacity:0.5,
        backgroundColor:"#0000FF"
    });
});
);
```

EASING:

Especifica la velocidad del elemento, en diferentes puntos de la animación.

El valor por defecto es: “swing”, puede tener dos valores posibles:

“**swing**” → se mueve lento al principio y al final, pero rápido en la mitad.

“**linear**” → se mueve a velocidad constante.

Nota: si se quieren otros valores para el easing, se puede utilizar un plugin con más valores.

Plugin easing para animaciones JQuery:

<http://gsgd.co.uk/sandbox/jquery/easing/>

<http://easings.net/es#>

Ej: usando el plugin:

```
.fadeToggle(2000,"easeInBounce");
```

Con el método .animate() es posible indicar un easing para cada propiedad CSS que se cambia.

```
.animate({
    height:["+=100","easeInOutCirc"],
    opacity:["0.5","easeInOutExpo"],
    backgroundColor:[get_random_color(),"easeInElastic"]
});
```

.delay() → retrasa la siguiente animación o efecto.

```
$("#caja1").slideUp(300).delay(800).fadeIn(400);
```

```
$("#caja2").slideUp(300).fadeIn(400);
```

//en este caso la “caja1” se demora más en aparecer.

.stop() → detiene la animación o efecto.

MÉTODO \$.AJAX():

```
<script type="text/javascript" src="spin-min.js"></script>
```

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

//el plugin spinner, crea un icono de cargando, sin usar gifs, sino con JQuery

```
$(document).on("ready",function(){
```

```
    //sacamos del formulario la url a la que se le va a hacer la petición:
```

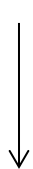
```
    var peticion=$("#principal form").attr("action");
```

```
    //sacamos del formulario el tipo de método (get o post):
```

```
    var metodo=$("#principal form").attr("method");
```

```
    $("#principal form").on("submit",function(evento){
```

```
        //evitamos el envío normal del formulario
```



evento.**preventDefault()**;

\$.ajax{

beforeSend: function(){

 //cargamos la imagen de cargando

 \$("#estatus").spin('miCustom');

 //armamos la info para enviar al server

 //una por una: var
 nom=document.formulario1.nombre.value; ...

 //y luego creamos un objeto Json y lo enviamos en data

 //{nombre: nombre,email:email,mensaje:mensaje}

 //o todas de una vez con serialize

 },

url:peticion,

type:metodo,

data: \$("#principal form").**serialize()**,

success:function(resp){

 if(resp == "Correcto"){

 \$("#estatus").html("");

 }else{

 \$("#estatus").html("");

 console.log(resp);

 }

 },

error:function(jqXHR,estado,error){

```

    });
  });
});
$( "#estatus" ).html( "<img src='x.png'>" );
console.log(estado);
console.log(error);
},
complete:function(jqXHR,estado){
    console.log(estado);
},
timeout:10000
});
});
});

```

```

<div id="principal">
    <h1>Formulario de Contacto</h1>
    <form name="formulario1" method="post" action="validar.php">
        <label for="nombre">Nombre:</label>
        <input type="text" id="nombre" name="nombre" /><br/>
        <label for="email">Email:</label>
        <input type="text" id="email" name="email"/><br/>
        <label for="mensaje">Mensaje:</label>
        <textarea
                                type="text"
                                name="mensaje"></textarea><br/>
                                id="mensaje"
        <input type="submit" value="Validar datos"/>
        <div id="estatus"></div>
    </form>
</div>

```

METODO \$.GET():

```
$.get("hola.php",{nombre:"Ricardo"},function(data){  
    $("#receptor").html(data);  
});
```

METODO \$.POST():

```
var info=$("#principal form").serialize();
```

```
$.post(peticion,info).error(  
    function(){  
        console.log("Error");  
    }).success(function(resp,estado,jqXHR){  
        console.log(resp);  
        console.log(estado);  
        console.log(jqXHR);  
    });
```

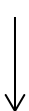
METODO \$.GETSCRIPT():

```
$.getScript("http://gsgd.co.uk/sandbox/jquery/easing/jquery.easing.1.3.js",function  
{  
    console.log("Termino de cargar el script");  
    $("#caja1").slideUp(3000,"easeOutBounce");  
});
```

METODO \$.GETJSON():

```
var info=$("#principal form").serialize();
```

```
$.getJSON("json.php",info,function(resp){  
    $.each(resp,function(clave,valor){
```



```

    console.log(clave+" -> "+valor);
  });
});

```

```

<?php
    $mnsjrespuesta=array(
        "n" => empty($_GET["nombre"])?"Sin Nombre":$_GET["nombre"],
        "c" => empty($_GET["email"])?"Sin Correo":$_GET["email"],
        "m" => empty($_GET["mensaje"])?"Sin Mensaje":$_GET["mensaje"]
    );

    echo json_encode($mnsjrespuesta);

?>

```



Librería de componentes o pluguins listos para usar del lado del cliente, nos sirve para utilizar efectos de animaciones y otras utilidades como acordeón, autocompletar, etc.

<http://jqueryui.com>

Ej: Usando la utilidad de autocompletar, de forma local (no saca datos del servidor con ajax).

```
<!doctype html>
```

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="utf-8">
```

```
    <title>jQuery UI Autocomplete - Default functionality</title>
```

↑

```
<link                                rel="stylesheet"
href="http://code.jquery.com/ui/1.10.4/themes/smoothness/jquery-
ui.css">

<script src="http://code.jquery.com/jquery-1.9.1.js"></script>

<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

<script>
    $(function() {
        var availableTags = [
            "ActionScript", "AppleScript", "Asp", "BASIC", "C",
            "C++", "Clojure", "COBOL", "ColdFusion", "Erlang",
            "Fortran", "Groovy", "Haskell", "Java", "JavaScript",
            "Lisp", "Perl", "PHP", "Python", "Ruby", "Scala",
            "Scheme"
        ];
        $( "#tags" ).autocomplete({
            source: availableTags
        });
    });
</script>

</head>

<body>
    <div>
        <label for="tags">Tags: </label>
        <input id="tags">
    </div>
</body></html>
```

Nota: para más información:

<http://www.desarrolloweb.com/articulos/primeros-paso-jquery-ui.html>

<http://www.desarrolloweb.com/manuales/manual-jqueryui.html>

RECURSOS ÚTILES:

- Pluguin para animar colores, con el método animate de JQuery:
<http://www.bitstorm.org/jquery/color-animation/>
- Pluguin easing para animaciones jquery:
<http://gsgd.co.uk/sandbox/jquery/easing/>
<http://easings.net/es#>
- Pluguin Spinner, crea un icono de cargando, sin usar gifs, sino con JQuery
<http://www.myjqueryplugins.com/jquery-plugin/spinjs>
- Taller de JQuery de desarrolloweb.com, en donde explican cómo crear pluguins e implementarlos, así como algunos pluguins ya listos para su uso.
<http://www.desarrolloweb.com/manuales/taller-jquery.html>
- Pluguins recomendados por desarrolloweb.com:
http://www.desarrolloweb.com/de_interes/interesantes-plugins-jquery-5211.html
- Muchos pluguins, animaciones y efectos de Javascript:
<http://www.ajaxshake.com/es/JS.html>
- Pluguins de JQuery:
<http://www.myjqueryplugins.com/>
- Flip Covers:
<http://www.jquery4u.com/animation/10-jquery-flip-effect-plugins/>
<http://www.jqueryrain.com/2012/12/best-jquery-page-flip-book-effect-with-examples/>
- Manual de JQuery, con ejemplos, pluguins, y se van publicando nuevos artículos con nuevos pluguins y herramientas muy útiles, también explica muy bien otros lenguajes:
<http://jquery-manual.blogspot.com>

