**SELECTORES**

# **El DOM**

Cuando un navegador carga un archivo HTML, muestra su contenido en la pantalla aplicando los estilos CSS correspondientes. Pero no todos los navegadores utilizan todas las etiquetas y contenidos del archivo.

Lo que hacen es memorizar un modelo HTML de la página. Es decir, el navegador recuerda las etiquetas HTML, sus atributos y el orden en el que aparecen en el archivo. A esta representación se le conoce como DOM.

Este modelo proporciona la información que necesita JavaScript para comunicarse con un elemento de la página. También proporciona las herramientas necesarias para navegar, cambiar y añadir código HTML a la página.

En realidad, DOM no forma parte de JavaScript. Es un modelo que desarrolló W3C y que han adoptado la mayoría de desarrolladores de navegadores. Este modelo permite que JavaScript se comunique y modifique el HTML de una página.

Como ya conocemos, JavaScript tiene diversas formas de seleccionar los elementos que aparecen en una página para que hagamos algo con ellos:

* Document.getElementById()
* Document.getElementsByTagName()
* Document.getElementsByClassName()
* Document.querySelectorAll()

# **Selección de elementos con JQuery**

Con jQuery podemos seleccionar uno o más elementos utilizando un comando especial conocido como objeto de jQuery. La sintaxis básica es:

$('selector')

Podemos utilizar todos los selectores de CSS. Por ejemplo, si queremos seleccionar una etiqueta cuyo **ID** sea ***banner*** escribiremos:

$('#banner')

Si queremos cambiar el código HTML que se encuentra dentro de un elemento podemos escribir:

$('banner').html('<h1>Esto es JavaScript</h1>');

# **Selectores básicos**

Los selectores básicos de CSS son los identificadores, las clases y los selectores de elementos.

# **Selectores de identificación**

Podemos seleccionar cualquier elemento de la página que tenga un identificador. Utilizaremos jQuery y un selector de identificador CSS. Por ejemplo, tenemos el siguiente código HTML en una página:

<p id="mensaje">Mensaje especial</p>

Para seleccionar este elemento usando jQuery, escribimos:

var mensaje=$('**#**mensaje');

# **Selectores de elementos**

JQuery sustituye el método **elementsByTagName()** por un selector. Basta con entregarle el nombre de la etiqueta:

En HTML: var listaLinks = document.getElementsByTagName('a');

En jQuery: var listaLinks = $('**a**');

JQuery posee más selectores aparte de los que vamos a ver. La lista completa se puede consultar en <http://api.jquery.com/category/selectors/>.

# **Selectores de clases**

Otra forma interesante de seleccionar elementos es utilizar el nombre de sus clases. Imaginemos que queremos diseñar una barra de navegación que incorpore menús desplegables. Cuando el usuario coloque el cursor del ratón sobre los botones del menú, aparecerá un desplegable con las opciones.

Usaremos JavaScript para controlar los menús, pero necesitamos una forma de programar el funcionamiento de los botones.

Una técnica es añadir una clase (como ***navBoton***) a todos los vínculos que tengan la barra de navegación principal, y utilizar JavaScript para localizarlos mediante el nombre de la clase. Para que esta barra de navegación funcione, necesitamos seleccionar únicamente los vínculos que tengan una clase determinada. Con JQuery tenemos un método sencillo para seleccionar todos los elementos que compartan el mismo nombre de clase:

$('.subMenu');

A partir de aquí, podremos trabajar, por ejemplo, ocultando todos los elementos que posean la clase submenú:

Ejemplo: $('subMenu').hide();

**Nota**: sería interesante visitar páginas que contengan tutoriales relacionados con JavaScript y CSS. Por ejemplo:

<https://www.htmldog.com/guides/css/>.

<https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Selectors>

# **Selectores avanzados**

Existen también selectores más avanzados. Con ellos podemos desarrollar muchas acciones.

**Selectores descendientes**

Podemos acceder a una etiqueta que se encuentra dentro de otra. Supongamos que queremos crear una lista desordenada de vínculos y que añadimos el identificador ***barNav*** para la etiqueta **<ul>:**

<ul id="barNav"></ul>

Si queremos obtener solo los enlaces que se encuentran dentro de dicha lista:

$('#barNav a');

**Selectores afiliados**

Se usan para obtener un hijo directo de una etiqueta. Por ejemplo, para seleccionar las etiquetas ***p*** afiliadas a la etiqueta ***body*** escribiremos:

$('body **>** p');

**Selectores de elementos adyacentes**

Permiten seleccionar una etiqueta que se encuentra inmediatamente después de otra. En este ejemplo tenemos un panel no visible (***div***) que aparece al hacer click sobre una pestaña que se encuentra en un elemento ***h2***. El elemento **div** está inmediatamente detrás del elemento **h2**. Para asegurarnos de que se mostrará el elemento ***div*** tendremos que seleccionarlo:

$('h2 **+** div');

Los selectores pueden ser de cualquier tipo: identificadores, clases o elementos.

**Selectores de atributos**

Permite seleccionar elementos basándonos en uno de sus atributos. Con un selector de atributos podemos localizar las etiquetas que tengan definido el atributo ***alt***. También podríamos localizar todos las atributos de los vínculos que apunten fuera de nuestro sitio web, y añadirles código para que abran dicho vínculo en una nueva ventana.

Por ejemplo, para localizar las etiquetas ***img*** que tengan definido su atributo ***alt***, escribiremos:

$('img**[**alt**]**');

Existen muchos selectores de atributos:

* [atributo]: selecciona los elementos a los que el código HTML haya asignado un atributo determinado. Por ejemplo, $('a[href] ') localiza todas las etiquetas ***a*** a las que se haya definido su atributo ***href***. Al realizar una selección a partir de atributos podremos excluir vínculos tomando como referencia su nombre:

<a name="nombre"></a>

* [atributo**=**"valor"]: selecciona los elementos que tengan un atributo con un valor determinado. Por ejemplo, localizar todos los cuadros de texto de un formulario:

$('input[type="text"] ');

* [atributo**^=**"valor"]: selecciona los elementos que tengan un atributo que comience por un valor determinado. Por ejemplo, localizar todos los vínculos que apunten a un destino que se encuentre fuera de nuestro sitio:

$('a[href^="http:// "] ');

* [atributo**$=**"valor"]: selecciona los elementos que tengan un atributo que termine por un valor determinado. Viene bien para trabajar con las extensiones de los archivos. Por ejemplo, localizar todos los vínculos que apunten a archivos .PDF:

$('a[href$=".pdf"] ');

* [atributo**\*=**"valor"]: selecciona los elementos que tengan un atributo que contenga un valor determinado. Por ejemplo, localizar todos los vínculos que apunten al dominio de microsoft:

$('a[href\*="microsoft.com"] ');

**Nota**: existen selectores muy interesantes para trabajar con formularios. Los veremos en otro apartado.

# **Filtros de jQuery**

JQuery también puede filtrar las selecciones basándose en ciertas características. Por ejemplo, el ***filtro*** ***:even*** permite seleccionar todos los elementos de una colección. Además, podemos localizar elementos que contengan determinadas etiquetas, texto, elementos que se oculten al usuario e incluso aquellos que no cumplan el valor del selector.

Para utilizar un filtro debemos escribir el ***nombre del selector, dos puntos y el nombre del filtro***. Por ejemplo, si queremos localizar todos los filtros de una tabla:

$('tr:even');

Este filtro selecciona todas las etiquetas **<tr>**. Para ajustar esta selección y centrarnos en todas las filas de una tabla utilizaremos el nombre de su clase:

$('.filtro tr:even');

**Funcionamiento de los filtros**:

***:even y :odd***: seleccionan todos los elementos de un grupo. Cada elemento de una selección jQuery tiene un número de índice (base 0 como en los arrays). Los filtros ***:even*** devuelven las posiciones 0,2,4,6,… y los filtros ***:odd*** devuelven 1,3,5,7,…

***:first y :last***: seleccionan el primer y el último elemento respectivamente. Por ejemplo, si queremos seleccionar el primer párrafo de una página web:

$('p:first');

***:not***: podemos utilizarlo para localizar los elementos que no coincidan con un tipo de selector determinado. Por ejemplo, queremos seleccionar todas las etiquetas ***a,*** a excepción de las que tengan la clase ***botonNav***.

$('a:not([href^=http://]) ');

***:hash():*** localiza los elementos que contengan otro selector. Por ejemplo, queremos localizar las etiquetas ***li*** pero solo si contienen la etiqueta ***a***.

$('li:hash(a) ');

Esta forma difiere del selector de descendencia, ya que no selecciona la etiqueta ***a***, sino la etiqueta ***li***. Solo las ***li*** que contengan las etiquetas ***a***.

***:contains()***: localiza los elementos que contienen un texto determinado:

$('a:contains(Hola) ');

***:hidden***: localiza los elementos que están ocultos, incluidos aquellos cuya propiedad ***display*** CSS sea ***none***, a los que se hayan ocultado utilizando la función ***hide()*** de jQuery, a los que tengan asignados ***valores 0*** a las propiedades ***width*** y ***height***, y a los ***campos ocultos*** de un formulario. Por ejemplo, si tenemos varias etiquetas ***div*** y queremos localizarlas y hacerlas visibles usando jQuery:

$('div:hidden').show();

Este código no tiene efecto sobre las etiquetas ***div*** visibles actualmente en la página.

***:visible***: es lo opuesto a ***hidden***.

# **Funcionamiento de los selectores de jQuery**

Cuando utilizamos jQuery para seleccionar uno o varios elementos, no terminaremos con una lista de nodos DOM, como los que se obtienen con las funciones ***getElementById()*** o ***getElementsByTagName().*** En realidad, obtendremos una selección de elementos especiales de jQuery. Estos métodos no pueden trabajar con los métodos DOM tradicionales. Por lo que ninguno de los métodos de JavaScript funciona con los objetos de jQuery.

Sin embargo, esto no es un problema ya que los métodos de un ***nodo DOM*** tienen equivalencias en jQuery. Es decir, podremos hacer lo mismo, pero más rápido.

# **Bucles automáticos**

Las funciones de jQuery tienen una propiedad para recorrer una colección de elementos. Por ejemplo, para seleccionar todas las imágenes que se encuentren dentro de una etiqueta ***div*** con un **id** denominado **deslizar** y luego ocultarlas, escribiremos:

$('#deslizar img').hide();

La función ***hide()*** genera un bucle que recorre la lista ocultando las imágenes de una en una.

# **Funciones variables**

En ocasiones querremos realizar diferentes acciones con los elementos seleccionados. Imaginemos que queremos definir el ancho y el alto de una etiqueta ***div*** cuyo ***ID*** se llama ***popUp*** utilizando JavaScript.

$('#popup').width(300).height(300);

JQuery utiliza un principio denominado encadenamiento, que permite añadir una función tras otra. Las funciones se encadenan mediante el carácter ***punto***. Por ejemplo, supongamos que además de definir el alto y el ancho de la etiqueta ***div***, quisiéramos añadir un texto y hacer que apareciese paulatinamente en la ventana.

$('#popup').width(300).height(300).text('Hola').fadeIn(2000);

Esta línea de código se suele leer mejor del siguiente modo:

$('#popup').width(300)

.height(300)

.text('Hola')

.fadeIn(2000);

# **Añadir contenido a una página**

JQuery tiene muchas funciones para manipular los elementos y el contenido de una página, sustituir el código HTML, agregar nuevo código a la página tomando como referencia un elemento seleccionado, e incluso eliminar todas las etiquetas y contenido de la página.

Para ver los ejemplos de este apartado utilizaremos este código:

<div id="contenedor">

<div id="errores">

<h2>Errores:</h2>

</div>

</div>

Vamos a ver las cinco funciones de jQuery que permiten manipular el contenido de una página web:

.***html()*** puede leer el código HTML que se encuentre dentro de un elemento y sustituirlo por otro. Para recuperar el código HTML que se encuentre dentro de una selección, añadiremos ***.html()*** detrás de ella. Por ejemplo:

alert($('#errores').html());

Este código crea un cuadro de diálogo con el texto **<h2>Errores:</h2>.**

Cuando utilizamos la función de esta forma, podremos crear una copia del código HTML que se encuentre dentro de un elemento concreto y pegarlo en otro elemento de la página.

Si entregamos una cadena como argumento a la función ***.html()***, sustituiremos los contenidos de esta selección:

$('#errores').html('<p>Tenemos 5 errores en el formulario</p>');

Esta línea de código sustituye todo el código HTML que se encuentre dentro y que esté identificado como un error. Modificamos el código de ejemplo por:

<div id="contenedor">

<div id="errores">

**<p>Tenemos 5 errores en el formulario</p>**

</div>

</div>

***Nota***: si utilizamos las funciones ***html()*** o ***text()*** para recuperar el código HTML, de una selección jQuery que contenga varios elementos, solo conseguiremos el código o el texto del primer elemento de la selección. Por ejemplo, si tuviésemos una página con **6 div** y ejecutásemos este código:

var contenidosDiv = $('div').html();

En la variable ***contenidosDiv*** solo se almacenarían los elementos del primer ***div***. Sin embargo, al utilizar las funciones ***html()*** o ***text()*** para insertar código HTML o texto en una selección jQuery, se incluirán todos los elementos seleccionados. Por ejemplo, este código:

$('div').html('<p>Hola</p>');

Sustituirá el código HTML de todos los ***div*** de la página por un único párrafo que tendrá la palabra Hola.

**.text()** funciona como ***html()*** pero no acepta argumentos. Se suele usar para sustituir el texto contenido en una etiqueta. Partiendo el ejemplo base, supongamos que después de ejecutar un programa para comprobar si hay errores en un formulario, queremos sustituir el texto **Errores:** por **No se han encontrado errores**. Utilizaríamos este código:

$('#errores h2').text('No se han encontrado errores');

No se modifica la etiqueta. Solo se modifica el contenido. JQuery codificará cualquier etiqueta que le pasemos a través de la función ***text()***, por lo que ***<p>*** se convertiría en ***&lt;p&gt;***. Por tanto, se puede usar para mostrar en la página un extracto de código.

**.append()** añade código HTML como un elemento afiliado del seleccionado. Por ejemplo, podemos seleccionar una etiqueta ***div*** pero, en vez de reemplazar sus contenidos, podemos añadir código antes de la etiqueta ***</div>*** de cierre. Esta función es interesante para añadir un elemento al final de una lista de ***<ul>*** o numerada ***<ol>.*** Veamos este código:

$('#errores').append('<p>Tenemos cinco errores en este formulario</p>');

Después de ejecutar esta función, el código será:

<div id="contenedor">

<div id="errores">

<h2>Errores:</h2>

<p>Tenemos 5 errores en el formulario</p>

</div>

</div>

**.prepend()** es como ***.append()***. La diferencia es que coloca el código HTML después de la etiqueta de apertura de la sección. Veamos este código:

$('#errores').prepend('<p>Tenemos cinco errores en este formulario</p>');

Después de ejecutar esta función, el código será:

<div id="contenedor">

<div id="errores">

<p>Tenemos 5 errores en el formulario</p>

<h2>Errores:</h2>

</div>

</div>

Para añadir código HTML fuera de la selección, ya sea antes de la etiqueta de apertura o después de la de cierre, usaremos las funciones .***before()*** o .***after()***. Por ejemplo, al trabajar con formularios se suele comprobar que no haya campos vacíos cuando el usuario envíe el formulario. Supongamos que nuestro código HTML contiene esta línea de código dentro de un formulario:

<input type="text" name="usuario" id="usuario" />

E imaginemos que aún quedase algún campo vacío cuando el usuario vaya a enviar el formulario. Podríamos escribir un programa que comprobase el estado del campo y que escribiese un mensaje de error. Para añadir un mensaje detrás del campo, podríamos utilizar la función **.after()**.

$('#usuario').after('<span class="error">Nombre de usuario obligatorio</span>');

Por lo que el código de error quedaría así:

<input type="text" name="usuario" id="usuario" />

<span class=“error”>Nombre de usuario obligatorio</span>

La función ***.before()***, hace lo mismo ubicando el contenido antes.

***Nota***: estas funciones son las más habituales, pero existen otras funciones. Podemos consultarlas en <http://api.jquery.com/category/manipulation/>.

# **Sustituir y eliminar selecciones**

Existen situaciones en las que necesitamos sustituir o eliminar un elemento seleccionado. Por ejemplo, supongamos que hemos utilizado JavaScript para crear un cuadro de diálogo, pero sin utilizar el método ***alert().***

Queremos crear algo más profesional, con un **div** **flotante** que colocaremos en la parte superior de la página. Cuando el usuario haga click sobre el botón ***Cerrar***, tendremos que eliminar el cuadro de diálogo de la página. Usaremos la función ***remove()*** de jQuery. Supongamos que el id del ***div*** se llama ***popup***. Lo eliminamos:

$('#popup').remove();

Esta función puede eliminar más de un elemento. Imaginemos que deseamos eliminar todas las etiquetas ***span*** que tuviesen la clase ***error***.

$('span.error').remove();

También podemos sustituir una sección por otro contenido. Por ejemplo, supongamos que tenemos una página con fotos de los productos que vende una empresa. Cuando un usuario haga click sobre una de las fotos, la página la añadirá al carro de la compra. Cuando se haga click en la imagen, podríamos sustituir la etiqueta ***img*** por algún texto como por ejemplo **Añadido al carro.**

Vamos a suponer que tenemos una etiqueta ***img*** con un ***id*** con un valor ***prod111***. Y que queremos sustituirla por un texto:

$('#prod111').replace(<p>Añadido al carro</p>);

Este código elimina la etiqueta ***<img>*** de la página y lo sustituye por la etiqueta ***<p>***.

***Nota***: jQuery también tiene una función denominada ***clone()*** que hace una copia del elemento seleccionado.

# **Definir y leer atributos de las etiquetas**

Con jQuery también podemos modificar los atributos de las etiquetas como, por ejemplo, añadir una clase a una etiqueta o cambiar una propiedad CSS. Podemos también utilizarlo para obtener el valor de un atributo como, por ejemplo, el lugar al que apunta una URL.

# **Clases**

Javascript tiene varias funciones para modificar los atributos de clase de una etiqueta.

***addClass()*** añade una clase determinada a un elemento. Podemos colocar esta función después de una selección jQuery y entregarle una cadena que represente el nombre de la clase que deseamos agregar. Por ejemplo, para añadir la clase ***linkExterno*** a todos los vínculos que apunten fuera del sitio web:

$('a[href^=http://] ').addClass('linkExterno');

Este código tomaría un código como:

<a href="http://www.microsoft.com/"/>

Y lo transformaría en:

<a href="http://www.microsoft.com/" class="linkExterno"/>

**Nota**: cuando utilicemos las funciones ***addClass()*** y ***removeClass()***, tendremos que proporcionar el nombre de la clase sin el punto.

Esta función de jQuery se puede usar también cuando una etiqueta ya tenga aplicada una clase. En este caso, la función ***addClass()*** no elimina las clases que ya tuviesen la etiqueta, sino que añadirá una nueva. Una tarea muy habitual es añadir varias clases a una misma etiqueta.

***removeClass()*** es la opuesta a ***addClass()***. También es posible que necesitemos crear una clase que no existiese, o eliminarla si existiese. Esto permite activar o desactivar elementos. Por ejemplo, al hacer click sobre un botón de opción lo activamos, y si volvemos a hacer click lo desactivamos. Supongamos que tenemos un botón que al hacer click sobre él, cambia la clase de la etiqueta **body**. Podremos cambiar el estilo de la página utilizando selectores descendientes para aplicar un conjunto de estilos. Al volver a hacer click sobre el botón, eliminaremos la clase de la etiqueta **body**, por lo que la página recuperará su aspecto original.

**Ejemplo**: supongamos que el usuario hace click en el botón para cambiar el estilo de la página cuyo **ID** es **cambioEstilo**, y que activaremos y desactivaremos la clase llamada **altEstilo** cada vez que hagamos click sobre el botón:

$('#cambioEstilo').click.function() {

$('body').toogleClass('altEstilo');

});

# **Leer y cambiar las propiedades CSS**

La función ***css()*** de jQuery permite cambiar las propiedades CSS de un elemento. Podemos utilizar esta función de tres formas:

**Para localizar el valor actual de una propiedad CSS**. Para ello le daremos su nombre a la función ***CSS().*** Por ejemplo, supongamos que queremos averiguar el color de fondo de una etiqueta **div** cuyo ID es ***main***:

var bgColor = $('#main').css('background-color');

**Nota**: jQuery no devuelve siempre los valores que esperamos. En el caso de los colores, jQuery siempre devolverá un valor rgb, o si tuviese una transparencia en el color, devolverá un color rgba. Además, como jQuery convierte las unidades en píxeles, podríamos utilizar CSS para definir el tamaño de la fuente de la etiqueta **body** y asignarle, por ejemplo, un tamaño del 150%. Cuando se compruebe el valor de la propiedad que regule el tamaño de la fuente, jQuery devolverá un valor en píxeles.

Esta función también permite definir una propiedad CSS de un elemento. Por ejemplo, para cambiar el tamaño de la fuente de la etiqueta **body** al 200%:

$('body').css('font-size','200%');

Por ejemplo, para asignar 100 píxeles al relleno de las etiquetas que tengan una clase denominada ***miClase***:

$('.miClase').css('padding',100);

Podemos utilizar el método abreviado de CSS. Por ejemplo, podríamos agregar un borde negro, de un píxel de ancho, a todos los párrafos que tengan la clase **resaltar**:

$('p.resaltar').css('border', '1px solid black');

Muchas veces es interesante cambiar la propiedad CSS basándonos en su valor actual. Supongamos que deseamos añadir un botón “***Aumentar el tamaño del texto***” a una página para que, cuando el usuario haga click en el botón, se duplique el tamaño de su fuente.

var fuenteBase = $('body').css('font-size');

fuenteBase = parseInt(fuenteBase);

$('body').css('font-size',fuenteBase \* 2);

**Nota**: este código modifica el tamaño de la fuente solo si el valor del tamaño del resto de las etiquetas de la página utiliza valores porcentuales, como **ems** o **porcentajes**. Si el resto de las etiquetas utilizase valores absolutos, como píxeles, el cambio del tamaño de la fuente ***body*** no les afectará.

# **Cambiar de forma simultánea varias propiedades CSS**

Si queremos cambiar dinámicamente una etiqueta, como por ejemplo **div**, podríamos modificar el color del fondo y añadir un borde de esta forma:

$('#resaltarDiv').css('background-color, '#FF0000');

$('#resaltarDiv').css('border', '2px solid #FE0037');

Otra forma de hacerlo es usar la función ***css()*** pasándole el literal de un objeto. El literal es una lista que contiene conjuntos de pares (nombres de las propiedades y sus valores). Ejemplo:

{

'background-color, '#FF0000',

'border', '2px solid #FE0037'

}

Para utilizar un literal con la función ***css()***:

$('#resaltarDiv').css({

{

'background-color, '#FF0000',

'border', '2px solid #FE0037'

});

Las comas separan el nombre de la propiedad del valor. Las llaves interiores indican el inicio del objeto y las exteriores el inicio de la función CSS.

También podemos aprovechar la capacidad de jQuery para encadenar funciones. Encadenar es aplicar varias funciones jQuery a una misma colección de elementos. Ejemplo:

$('#resaltarDiv').css('background-color, '#FF0000')

.css('border', '2px solid #FE0037');

Al encadenar funciones obtenemos un rendimiento mejor que al realizar dos veces la sección ***$('#resaltarDiv')***, ya que cada vez que realizamos una selección, obligamos al navegador a ejecutar todo el código jQuery. Por esta razón, el código siguiente no era la mejor opción:

$('#resaltarDiv').css('background-color, '#FF0000');

$('#resaltarDiv').css('border', '2px solid #FE0037');

Al encadenar, el navegador solo tiene que seleccionar el elemento una vez, y luego ejecutar dos veces la función CSS.

# **Leer, definir y eliminar atributos HTML**

JQuery cuenta con funciones nativas para realizar estas tareas. JQuery tiene funciones especialmente diseñadas para trabajar con atributos HTM: las funciones ***attr()*** y ***removeAttr()***.

La función ***attr()*** permite leer un atributo HTML de una etiqueta. Por ejemplo, para obtener el archivo al que señala una etiqueta ***img***, entregamos a la función la cadena '***src***':

var archivoImagen = $('#banner img').attr('src');

Este ejemplo nos devuelve el valor de la propiedad ***src*** de la primera etiqueta ***img*** que se encuentra dentro de otra cuyo ***ID*** sea ***banner***.

**Nota**: no se distingue entre mayúsculas y minúsculas al aportar el nombre del atributo.

Podemos aportar un segundo atributo. Si queremos cambiar a otra imagen:

$('#banner img').attr('src','imagenes/nuevaImagen.jpg');

Para eliminar un atributo, tenemos la función **removeAttr():**

$('body').removeAttr('bgColor');

# **Actuar sobre cada elemento de una sección**

Una de las ventajas de jQuery es que la mayoría de las funciones establecen bucles de forma automática con todos los elementos de una selección jQuery. Por ejemplo, para difuminar todos los elementos ***img*** de una página hasta que desaparezcan, solo necesitaremos una línea de código jQuery.

La función ***.fadeOut()*** hace que un elemento desaparezca poco a poco. Si la asociamos a una selección jQuery que contiene varios elementos, se logrará el efecto sobre todos ellos.

JQuery tiene una función para recorrer los elementos seleccionados: ***.each***.

Imaginemos que queremos publicar todos los vínculos externos de una web en una página bibliográfica. Podríamos tener una sección de la página en donde mostrásemos las web de otros sitios relacionados con el artículo. Los pasos serían:

* Recuperar todos los vínculos que apunten a algún lugar fuera de nuestra web.
* Obtener el atributo ***href*** de cada vínculo (la URL).
* Añadir esta URL a la lista de vínculos de esta bibliografía.

Podríamos utilizar: ***$('selector').each();***

# **Funciones anónimas**

Para utilizar la función ***.each()***, le entregaremos un tipo especial de argumento: una función anónima. Esta función es especial ya que contiene las acciones que queremos realizar sobre cada uno de los elementos seleccionados. Se les llama anónimas porque, a diferencia de las funciones que hemos visto hasta ahora, no les vamos a asignar ningún nombre. Esta es la estructura básica:

**Function(){**

**// código**

**}**

Al ser anónimas, no se les puede llamar. Con las funciones anónimas, las usamos como argumento que pasaremos a otra función. Así, incorporamos una función anónima como parte de la función ***each()***:

$('selector').each(function() {

// código

})

El intérprete de jQuery considerará a todo este código como una única declaración.

Una vez tenemos la estructura externa en su lugar, colocaremos algo de código dentro de ella, es decir, definiremos todo lo que haremos a los elementos de la selección. La función ***each()*** funciona como un bucle actuando sobre cada uno de los elementos seleccionados.

Imaginemos que tenemos una página con 20 imágenes y que añadimos este código JavaScript a unos de los scripts:

$('img').eachfunction() {

alert('He encontrado una imagen');

});

Aparecerán 20 alertas.

***Nota***: recordemos que hemos usado una función anónima en el caso de la función ***document.ready()***. La usábamos para asegurarnos de que se completaba la carga del código HTML de la página antes de que el navegador ejecute el código JavaScript.

# **this y $(this)**

al utilizar la función ***each()***, queremos acceder al conjunto de atributos de cada elemento. Para acceder al elemento actual del bucle, utilizamos una palabra especial: ***this***.

Por la forma de trabajar de jQuery, ***this*** se referirá al elemento del modelo DOM tradicional, y podremos acceder a sus propiedades, pero como ya hemos visto antes, la selección especial de jQuery nos permite utilizar sus funciones, de modo que para convertir todo esto en su equivalente de jQuery, escribiremos ***$(this)***.

Para entenderlo, vamos a ver un ejemplo que hicimos antes: crear una lista de vínculos externos (***bibliografia***).

<div id="bibliografia">

<h2>Páginas web referenciadas en este art&iacute;culo</h2>

<ul id="listaBiblio">

</ul>

</div>

El primer paso es conseguir una lista con todos los vínculos que apunten al exterior de nuestro sitio web:

$('a[href^="http:// "]')

Ahora añadimos la función ***each()***:

$('a[href^="http:// "]').each()

Ya añadimos otra función anónima:

$('a[href^="http:// "]').each(**function**){

});

La primera parte de la función anónima es recuperar la URL del vínculo. Como cada una tiene una URL diferente, tendremos que acceder al elemento actual en cada repetición del bucle:

$('a[href^=“http://”]').each(function){

**var linkExterno = $(this).attr('href');**

});

A continuación, utilizaremos la etiqueta ***ul*** para adjuntar una nueva lista:

$('a[href^="http:// "]').each(function){

var linkExterno = $(this).atributo('href');

**$('listaBiblio').append('<li>' + linkExterno + '</li>';**

});

**Nota**: este script es un buen ejemplo del uso de **$(this)**, pero no es la mejor forma de escribir una lista de vínculos externos. En primer lugar, si no hubiese ningún vínculo externo, la etiqueta **div** aparecería vacía. Además, si alguien visitase la página sin tener activo JavaScript en su navegador, no vería los vínculos, sino una caja vacía.

Lo mejor sería utilizar JavaScript para crear una etiqueta **div**. Lo veremos en un ejemplo.