Manual de jQuery

Autores del manual

Este manual ha sido realizado por los siguientes colaboradores de DesarrolloWeb.com:

Miguel Angel Alvarez
Director de DesarrolloWeb.com
http://www.desarrolloweb.com
(26 capítulos)

Introducción a jQuery

Bienvenidos al manual sobre jQuery que vamos a publicar en DesarrolloWeb.com, con el que pretendemos clarificar a los usuarios el método de trabajo y programación de aplicaciones del lado del cliente, compatibles con todos los navegadores más comunes.

Qué es jQuery

Para simplificar, podríamos decir que jQuery es un framework Javascript, pero quizás muchos de los lectores se preguntarán qué es un framework. Pues es un producto que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales. Por decirlo de otra manera, framework son unas librerías de código que contienen procesos o rutinas ya listos para usar. Los programadores utilizan los frameworks para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio framework ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar.

> **Nota:**si no sabes lo que es Javascript seguramente no te interesará este artículo, pero puedes aprenderlo también en DesarrolloWeb.com: Qué es Javascript

Por ejemplo, en el caso que nos ocupa, jQuery es un framework para el lenguaje Javascript, luego será un producto que nos simplificará la vida para programar en este lenguaje. Como probablemente sabremos, cuando un desarrollador tiene que utilizar Javascript, generalmente tiene que preocuparse por hacer scripts compatibles con varios navegadores y para ello tiene que incorporar mucho código que lo único que hace es detectar el browser del usuario, para hacer una u otra cosa dependiendo de si es Internet Explorer, Firefox, Opera, etc. jQuery es donde más nos puede ayudar, puesto que implementa una serie de clases (de programación orientada a objetos) que nos permiten programar sin preocuparnos del navegador con el que nos está visitando el usuario, ya que funcionan de exacta forma en todas las plataformas más habituales.

Así pues, este framework Javascript, nos ofrece una infraestructura con la que tendremos mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Por ejemplo, con jQuery obtendremos ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de Ajax, etc. Cuando programemos Javascript con jQuery tendremos a nuestra disposición una interfaz para programación que nos permitirá hacer cosas con el navegador que estemos seguros que funcionarán para todos nuestros visitantes. Simplemente debemos conocer las librerías del framework y programar utilizando las clases, sus propiedades y métodos para la consecución de nuestros objetivos.

Además, todas estas ventajas que sin duda son muy de agradecer, con jQuery las obtenemos de manera gratuita, ya que el framework tiene licencia para uso en cualquier tipo de plataforma, personal o comercial. Para ello simplemente tendremos que incluir en nuestras páginas un script Javascript que contiene el código de jQuery, que podemos descargar de la propia página web del producto y comenzar a utilizar el framework.

El archivo del framework ocupa unos 56 KB, lo que es bastante razonable y no retrasará mucho la carga de nuestra página (si nuestro servidor envía los datos comprimidos, lo que es

bastante normal, el peso de jQuery será de unos 19 KB). Además, nuestro servidor lo enviará al cliente la primera vez que visite una página del sitio. En siguientes páginas el cliente ya tendrá el archivo del framework, por lo que no necesitará transferirlo y lo tomará de la caché. Con lo que la carga de la página sólo se verá afectada por el peso de este framework una vez por usuario. Las ventajas a la hora de desarrollo de las aplicaciones, así como las puertas que nos abre jQuery compensan extraordinariamente el peso del paquete.

Ventajas de jQuery con respecto a otras alternativas

Es importante comentar que jQuery no es el único framework que existe en el mercado. Existen varias soluciones similares que también funcionan muy bien, que básicamente nos sirven para hacer lo mismo. Como es normal, cada uno de los frameworks tiene sus ventajas e inconvenientes, pero jQuery es un producto con una aceptación por parte de los programadores muy buena y un grado de penetración en el mercado muy amplio, lo que hace suponer que es una de las mejores opciones. Además, es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework. Otra cosa muy interesante es la dilatada comunidad de creadores de plugins o componentes, lo que hace fácil encontrar soluciones ya creadas en jQuery para implementar asuntos como interfaces de usuario, galerías, votaciones, efectos diversos, etc.

Uno de los competidores de jQuery, del que hemos publicado ya en DesarrolloWeb.com un amplio manual para programadores, es Mootools, que también posee ventajas similares. Os dejo el enlace al Manual de Mootools, que también puede ser interesante, porque seguramente lo tengamos explicado con mayor detalle que jQuery.

jQuery, es para mi?

Si estás interesado en enriquecer tu página web con componentes de la llamada Web 2.0, como efectos dinámicos, Ajax, interacción, interfaces de usuario avanzadas, etc., jQuery es una herramienta imprescindible para desarrollar todas estas cosas sin tener que complicarte con los niveles más bajos del desarrollo, ya que muchas funcionalidades ya están implementadas, o bien las librerías del framework te permitirán realizar la programación mucho más rápida y libre de errores.

Ahora bien, todas estas mejoras de la web 2.0, que en un principio puede ser muy atractivas, también tienen un coste en tiempo de desarrollo de los proyectos. Sin un framework como jQuery, el tiempo de creación y depuración de todos esos componentes dinámicos sería mucho mayor, pero aun así nadie dice que todo sea instalar el sistema y empezar correr. Sin embargo, lo más complicado de jQuery es aprender a usarlo, igual que pasa con cualquier otro framework Javascript. Requerirá del desarrollador habilidades avanzadas de programación, así como el conocimiento, al menos básico, de la programación orientada a objetos. Una vez aprendido las ventajas de utilizarlo compensarán más que de sobra el esfuerzo. Esperamos que con este Manual de jQuery, que vamos a publicar en DesarrolloWeb.com puedas aprender lo necesario para desarrollar tus propios componentes dinámicos en Javascript con los que enriquecer tus aplicaciones.

Podemos conocer jQuery accediendo a la página de inicio del framework Javascript: http://jquery.com/

*Artículo por **Miguel Angel Alvarez***

Demo muy simple de uso de jQuery

Con objetivo de que los lectores puedan hacerse una rápida idea de las posibilidades de jQuery, escribiendo unas brevísimas líneas de código Javascript, vamos a publicar un par de ejemplos bien simples que nos ilustren, pero sin complicarnos demasiado. Nos servirán para la introducción a jQuery que estamos publicando en el Manual de jQuery.

La idea de este artículo no es explicar las funcionalidades que vamos a demostrar, sino ver el poco código que hemos tenido que escribir para realizar unos scripts con dinamismos sencillos. Quizás los scripts en si no digan mucho a un lector poco experimentado, pero los que ya han tenido contacto con los pormenores que hay que seguir para hacer estos efectos, de manera que sean compatibles con todos los navegadores, sabrán que jQuery nos ha simplificado mucho nuestra tarea.

Así pues, no te preocupes demasiado con los detalles de estos códigos, que los explicaremos en DesarrolloWeb.com más adelante con detalle.

Demo 1 de jQuery

Para empezar vamos a ver este ejemplo, donde tenemos dos botones y un texto. Al pulsar un botón, cambiaremos el texto y al pulsar el otro pondremos otro texto distinto.

Podemos ver el ejemplo en marcha en una página aparte.

En este ejemplo tenemos una capa que tiene este código

```
<div id="capa" style="padding: 10px; background-color: #ff8800">Haz clic en un botón</div>
```

Luego tenemos dos botones con estos códigos:

```
<input type="button" value="Botón A" onclick="$('#capa').html('Has hecho clic en el botón <b>A</b>')">
<input type="button" value="Botón B" onclick="$('#capa').html('Recibido un clic en el botón <b>B</b>')">
```

Como se puede ver, en los botones hay definidos un par de eventos onclick (uno en cada uno) que ejecutan una instrucción Javascript cuando se hace clic sobre ellos. La instrucción está en Javascript, pero hace uso de algunas herramientas disponibles en jQuery para trabajo con los elementos de la página. En este caso, por explicarlo rápidamente, se hace una selección del elemento DIV de la capa y luego se ejecuta un método sobre él para cambiar el contenido HTML del elemento.

Demo 2 de jQuery

En este otro ejemplo vamos a ver algo un poquito más complejo. Bueno, realmente no tiene mucha mayor complejidad, pero estamos utilizando jQuery de una manera un poco distinta, que requiere algo más de código por nuestra parte. Ahora vamos a tener dos capas en nuestra página. Una capa estará siempre visible y otra capa aparecerá inicialmente oculta y la vamos a mostrar u ocultar dependiendo de si el usuario coloca el ratón encima de la capa que está siempre visible.

Para hacerse una idea exacta de nuestro ejemplo puedes verlo en una ventana aparte.

En este segundo ejemplo tenemos este código HTML, con las dos capas.

```
<div id="capa" style="padding: 10px; background-color: #ff8800;">Pon el ratón encima de esta capa</div>
<br>
<div id="mensaje" style="display: none;">Has puesto el ratón encima!! <br>(Ahora quítalo de la capa)</div>
```

Ahora vamos a tener un código Javascript con jQuery que defina los eventos del usuario, para cuando sitúa el ratón dentro o fuera de la primera capa.

```
$("#capa").mouseenter(function(evento){
  $("#mensaje").css("display", "block");
});
$("#capa").mouseleave(function(evento){
  $("#mensaje").css("display", "none");
});
```

De esta sencilla manera, hemos creado dos eventos en el DIV con id="capa". Además, hemos definido el código de los eventos por medio de funciones, que se encargarán de mostrar o ocultar la segunda capa con id="mensaje".

```
$("#mensaje").css("display", "block");
```

Esto nos selecciona la capa con id "mensaje" y con el método css() indicamos que queremos cambiar el atributo "display" al valor "block" de ese elemento.

```
$("#mensaje").css("display", "none");
```

Esta otra línea muy similar, simplemente cambia el "display" a "none" para ocultar el elemento.

Esperamos que estos dos ejemplos te hayan servido para ver rápidamente algunas cosas que se pueden hacer con jQuery con muy poco esfuerzo y que funcionan en todos los navegadores.

*Artículo por **Miguel Angel Alvarez***

Pasos para utilizar jQuery en tu página web

En este artículo te vamos a explicar cómo comenzar a utilizar jQuery en tus páginas web, paso a paso, para que puedas hacer tu primer script jQuery y así comprender los fundamentos de uso de este framework Javascript. En este punto estaría bien que sepas lo que es jQuery, lo que ha sido explicado ya en el Manual de jQuery que venimos publicando en DesarrolloWeb.com.

La idea es que puedas ir haciendo tú mismo los pasos que vamos a relatar, para que compruebes tú mismo lo sencillo que es comenzar a utilizar jQuery. Este artículo sigue el esquema con el que los propios creadores de jQuery enseñan a utilizar su producto, así que está directamente inspirado en la documentación de jQuery.

Vídeo: Todo el proceso de creación de un primer ejemplo con jQuery relatado en este artículo lo hemos grabado en el videotutotial comenzar a programar con jQuery.

1.- Descarga la última versión del framework

Accede a la página de jQuery para descargar la última versión del framework. En el momento de escribir este artículo la release actual es la 1.3.2, y con la que hemos realizado estos ejemplos. Sin embargo, puede que haya publicado una nueva versión que mejore la actual.

Dan dos posibilidades para descargar, una que le llaman PRODUCTION, que es la adecuada

para páginas web en producción, puesto que está minimizada y ocupa menos espacio, con lo que la carga de nuestro sitio será más rápida. La otra posibilidad es descargar la versión que llaman DEVELPOMENT, que está con el código sin comprimir, con lo que ocupa más espacio, pero se podrá leer la implementación de las funciones del framework, que puede ser interesante en etapa de desarrollo, porque podremos bucear en el código de jQuery por si tenemos que entender algún asunto del trabajo con el framework.

Verás que la descarga es un archivo js que contiene el código completo del framework. Coloca el archivo en una carpeta en tu ordenador para hacer las pruebas.

2.- Crea una página HTML simple

Ahora, en el mismo directorio donde has colocado el archivo js, coloca un archivo html con el siguiente código.

```
<html>
  <head>
    <script src="jquery-1.3.2.min.js" type="text/javascript"></script>
    <script>

    </script>
  </head>
<body>
<a href="http://www.desarrolloweb.com/">DesarrolloWeb.com</a>
</body>
</html>
```

Como podrás ver, es una página bien simple, en la que tenemos una llamada a un script externo llamado jquery-1.3.2.min.js. Este es el código de jQuery que hemos descargado de la página del framework. Si has descargado una versión distinta, quizás el archivo se llame de otra manera, así que es posible que tengas que editar ese nombre de archivo para colocar el que tengas en el directorio.

Con ese script ya hemos incluido todas las funciones de jQuery dentro de nuestra página. Sólo tienes que prestar atención a que tanto el archivo .html de esta página, como el archivo .js del framework estén en el mismo directorio. Y, como decía, que el archivo que incluimos con la etiqueta SCRIPT sea el .js que nosotros hemos descargado.

Además, como se puede ver, hemos dejado dentro del HEAD una etiqueta SCRIPT de apertura y cierre que está vacía. Todo el código que vamos a explicar a continuación tendrás que colocarlo en dentro de esa etiqueta.

3.- Ejecutar código cuando la página ha sido cargada

El paso que vamos a explicar ahora es importante que se entienda, aunque sin lugar a dudas a lo largo del manual publicado en DesarrolloWeb.com, lo veremos hasta la saciedad. Se trata de detectar el momento en que la página está lista para recibir comandos Javascript que hacen uso del DOM.

Cuando hacemos ciertas acciones complejas con Javascript tenemos que estar seguros que la página haya terminado de cargar y esté lista para recibir comandos Javascript que utilicen la estructura del documento con el objetivo de cambiar cosas, como crear elementos, quitarlos, cambiar sus propiedades, etc. Si no esperamos a que la página esté lista para recibir instrucciones corremos un alto riesgo de obtener errores de Javascript en la ejecución.

En el taller de programación con el DOM de Javascript hemos podido explicar que es el DOM y

la importancia de realizar acciones sólo cuando el DOM está listo. Pero si no queremos entretenernos con la lectura de este texto, sirve con saber que, cuando queremos hacer acciones con Javascript que modifiquen cualquier cosa de la página, tenemos que esperar a que la página esté lista para recibir esos comandos.

Generalmente, cuando se desea ejecutar Javascript después de la carga de la página, si no utilizamos ningún framework, lo más normal será utilizar un código como este:

```
window.onload = function () {
  alert("cargado...");
}
```

Pero esta sentencia, que carga una funcionalidad en el evento onload del objeto window, sólo se ejecutará cuando el navegador haya descargado completamente TODOS los elementos de la página, lo que incluye imágenes, iframes, banners, etc. lo que puede tardar bastante, dependiendo de los elementos que tenga esa página y su peso.

Pero en realidad no hace falta esperar todo ese tiempo de carga de los elementos de la página para poder ejecutar sentencias Javascript que alteren el DOM de la página. Sólo habría que hacerlo cuando el navegador ha recibido el código HTML completo y lo ha procesado al renderizar la página. Para ello, jQuery incluye una manera de hacer acciones justo cuando ya está lista la página, aunque haya elementos que no hayan sido cargados del todo. Esto se hace con la siguiente sentencia.

```
$(document).ready(function(){
  //código a ejecutar cuando el DOM está listo para recibir instrucciones.
});
```

Por dar una explicación a este código, digamos que con $(document) se obtiene una referencia al documento (la página web) que se está cargando. Luego, con el método ready() se define un evento, que se desata al quedar listo el documento para realizar acciones sobre el DOM de la página.

4.- Insertar un manejador de evento a la etiqueta A (enlace) que hay en la página

Ahora que ya sabemos cómo y cuando debemos ejecutar las acciones de jQuery que alteren la funcionalidad, contenidos o aspecto de la página, podemos insertar un poco de código para demostrar el método de trabajo con jQuery.

Para este primer ejemplo vamos a crear un evento click en el enlace, para mostrar un mensaje cuando se haga clic sobre el link. Para crear un evento click sobre un elemento tenemos que invocar al método click sobre ese elemento y pasarle como parámetro una función con el código que queremos que se ejecute cuando se hace clic.

```
$("a").click(function(evento){
  //aquí el código que se debe ejecutar al hacer clic
});
```

Como vemos en el código anterior, con $("a") obtenemos una referencia al enlace. Bueno, en realidad con ello estamos estamos seleccionando todas las etiquetas A (enlaces) del documento, pero como sólo hay un enlace, en realidad nos vale. Luego, el método click() del sobre $("a") estamos definiendo un evento, que se ejecutará cuando se haga clic sobre el enlace. Como se puede ver, al método click se le pasa una función donde se debe poner el código Javascript que queremos que se ejecute cuando se haga clic sobre el enlace.

Ahora veamos la definición del evento clic completa, colocada dentro del evento ready del document, para que se asigne cuando la página está lista.

```
$(document).ready(function(){
  $("a").click(function(evento){
    alert("Has pulsado el enlace...nAhora serás enviado a DesarrolloWeb.com");
  });
});
```

Por líneas, esto es una recapitulación de lo que hemos hecho:

```
$(document).ready(function(){
```

Esta línea sirve para hacer cosas cuando la página está lista para recibir instrucciones jQuery que modifiquen el DOM.

```
$("a").click(function(evento){
```

Con esta línea estamos seleccionando todas las etiquetas A del documento y definiendo un evento click sobre esos elementos.

```
alert("Has pulsado el enlace...nAhora serás enviado a DesarrolloWeb.com");
```

Con esta línea simplemente mostramos un mensaje de alerta informando al usuario que se ha hecho clic sobre el enlace.

5.- Guarda el archivo html y ábrelo en un navegador

Una vez que tenemos hecho nuestra primera página con jQuery, la puedes guardar y ejecutarla en un navegador. Prueba hacer clic en el enlace y debería salirte la ventana de alerta.

Puedes ver este script en funcionamiento en una página aparte.

Ya está hecho y funcionando tu primer script con jQuery!

Por si acaso quedaron dudas, dejamos aquí el código completo que deberías tener:

```
<html>
<head>
  <title>Primer script con jQuery</title>
<script src="jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
  $("a").click(function(evento){
    alert("Has pulsado el enlace...nAhora serás enviado a DesarrolloWeb.com");
  });
});
</script>
</head>

<body>

<a href="http://www.desarrolloweb.com">Ir a DesarrolloWeb.com</a>

</body>
</html>
```

6.- Extra: Bloquear el comportamiento normal de un enlace

Ahora veamos una pequeña modificación para alterar el comportamiento por defecto de los enlaces. Como sabemos, cuando se pulsa un enlace nos lleva a una URL. Luego al hacer click, primero se ejecuta lo que hayamos colocado en el evento click del enlace y luego el enlace lleva al navegador a una nueva URL.

Este comportamiento se puede bloquear también desde jQuery, añadiendo una llamada al

método preventDefault() sobre el evento. Si te fijas, la función definida para marcar el comportamiento del evento click sobre el enlace recibía un parámetro. Ese parámetro es un manejador de evento. Y tiene sus propios métodos y propiedades, como este preventDefault() que comentábamos. Su uso es el siguiente:

```
$(document).ready(function(){
  $("a").click(function(evento){
    alert("Has pulsado el enlace, pero vamos a cancelar el evento...nPor tanto, no vamos a llevarte a
DesarrolloWeb.com");
    evento.preventDefault();
  });
});
```

Como hemos podido ver invocando a evento.preventDefault() lo que conseguimos en este caso es que el link no lleve a ningún sitio, simplemente se ejecutará el código Javascript contenido para el evento click.

*Artículo por **Miguel Angel Alvarez***

Vídeo: comenzar a programar con jQuery

Presentamos el primero de los vídeos que vamos a publicar en DesarrolloWeb.com para explicar el funcionamiento de jQuery, un conocido framework Javascript del que venimos creando un manual en texto: Manual de jQuery.

En este primer vídeo se muestra desde el principio la creación de un primer ejemplo con jQuery. Comenzaremos mostrando algo tan sencillo como descargar el framework e incluirlo en una página HTML. En el videotutorial realizaremos luego un script sencillo para probar que todo está funcionando correctamente y de paso ver como con jQuery podemos crear funcionalidades dinámicas en páginas web con realmente poco esfuerzo.

En el vídeo se verán rápidamente muchos conceptos que no se llegan a explicar con detalle, puesto que para hacer un primer ejemplo se necesitan utilizar varios componentes que son complicados de explicar con detalle sin conocimientos previos. El objetivo no es sino mostrar cómo cualquier persona puede hacer rápidamente un script con jQuery en pocos minutos. No obstante, en este vídeo seguimos el guión que se explicó en el artículo Primeros pasos con jQuery, donde encontraremos diversas guías adicionales, comentadas con más calma. Para poder publicar el vídeo en Youtube, nos hemos limitado a una duración de 10 minutos, por lo que algunas veces merecerá la pena consultar el mencionado artículo.

El ejemplo que realizaremos a lo largo de este vídeo se puede ver en una página aparte.

Así pues, esperamos que este videotutorial sirva de utilidad a las personas que quieren empezar a programar con jQuery y no sepan por donde empezar.

Claro que estos ejercicios son un poco especiales, dado que sirven para ilustrar el modo de trabajo con jQuery, pero sin explicar todos los detalles relacionados con el uso del framework. Por que de momento lo que queremos es mostrar una introducción al sistema y mostrar por encima algunas de sus posibilidades. En el futuro publicaremos artículos que irán poco a poco explicando todos los detalles de trabajo con jQuery.

En el caso que nos ocupa, queremos demostrar el uso de jQuery para alterar elementos de una página web, añadiendo y quitando clases CSS. Esto es bien simple, porque en jQuery los elementos tienen dos clases llamadas addClass() y removeClass(), que sirven justamente para que el elemento que recibe el método se le aplique una clase CSS o se le elimine. Así que lo que vamos a aprender, con respecto al artículo anterior -Pasos para utilizar jQuery-, es utilizar esos nuevos métodos de los elementos.

Para complicarlo sólo un poco más, vamos a añadir y quitar clases del elemento con respuesta a acciones del usuario, para aprender también nuevos eventos de usuario.

En nuestro ejemplo vamos a tener dos elementos. Primero una capa DIV con un texto. Después tendremos un enlace que estará fuera de la capa DIV. Al situar el usuario el ratón sobre un enlace añadiremos una clase CSS a la capa DIV y al retirar el ratón del enlace eliminaremos la class CSS que habíamos añadido antes. Si se desea, para aclarar el caso de nuestro ejemplo, podemos ver el ejercicio en marcha en una página aparte.

Nota: Se supone que ya hemos leído el artículo anterior, en el que explicamos paso por paso hacer tu primera página con jQuery, pues necesitaremos conocer algunas cosas que ya se han comentado allí.

1.- Crear la página con una capa, un enlace y la definición de una clase CSS

El primer paso sería construir una página con todos los elementos que queremos que formen parte de este primer ejemplo: la capa DIV, el enlace y la definición de la class CSS.

Además, ahora también vamos a incluir el script de jQuery, que lo necesitaremos para acceder a las funciones del framework Javascript.

```
<html>
<head>
  <title>Añadir y quitar clases CSS a elementos</title>
  <script src="jquery-1.3.2.min.js" type="text/javascript"></script>
<style type="text/css">
.clasecss{
  background-color: #ff8800;
  font-weight: bold;
}
</style>
</head>

<body>
<div id="capa">
Esta capa es independiente y voy a añadir y eliminar clases css sobre ella
</div>
<br>
<br>
<a href="http://www.desarrolloweb.com">Añadir y quitar clase en la capa de arriba</a>
</body>
</html>
```

Perfecto, ahora ya tenemos la infraestructura necesaria para nuestro ejemplo, con todos los

integrantes del mismo. Sólo nos faltaría hacer el siguiente paso, que es añadir los comportamientos dinámicos con jQuery.

2.- Recordar: añadir siempre los scripts jQuery cuando el documento está "ready"

Ahora vamos a empezar a meter el código Javascript, pero quiero comenzar por recordar a los lectores que cualquier funcionalidad que queramos agregar a la página por medio de jQuery, debe ser incluida cuando el documento está listo para recibir acciones que modifiquen el DOM de la página.

Para esto tenemos que incluir siempre el código:

```
$(document).ready(function(){
  //aquí meteremos las instrucciones que modifiquen el DOM
});
```

3.- Añadir los eventos mouseover y mouseout para añadir y quitar una clase CSS

En este paso vamos a crear un par de eventos que asociaremos a los enlaces. Este par de eventos serán lanzados cuando el usuario coloque el puntero del ratón sobre el enlace (mouseover) y cuando el usuario retire el ratón del enlace (mouseout).

Por ejemplo, para definir un evento mouseover se tiene que llamar al método mouseover() sobre el elemento que queremos asociar el evento. Además, al método mouseover() se le envía por parámetro una función con el código que se quiere ejecutar como respuesta a ese evento.

En el caso de añadir una clase tenemos que utilizar el método addClass(), que se tiene que invocar sobre el elemento al que queremos añadirle la clase. A addClass() tenemos que pasarle una cadena con el nombre de la clase CSS que queremos añadir.

Para seleccionar el elemento que queremos añadir la clase hacemos $("#idElemento"), es decir, utilizamos la función dólar pasándole el identificador del elemento que queremos seleccionar, precedida del carácter "#". Por ejemplo, con $("#capa") estamos seleccionando un elemento de la página cuyo id="capa".

```
$("a").mouseover(function(event){
  $("#capa").addClass("clasecss");
});
```

De manera análoga, pero con el método mouseout(), definimos el evento para cuando el usuario saca el puntero del ratón del enlace.

```
$("a").mouseout(function(event){
  $("#capa").removeClass("clasecss");
});
```

4.- Código completo del ejemplo jQuery

Ahora veamos el código completo de este ejercicio.

```
<html>
<head>
```

```
  <title>Añadir y quitar clases CSS a elementos</title>
  <script src="jquery-1.3.2.min.js" type="text/javascript"></script>
<style type="text/css">
.clasecss{
  background-color: #ff8800;
  font-weight: bold;
}
</style>
<script>
$(document).ready(function(){
  $("a").mouseover(function(event){
    $("#capa").addClass("clasecss");
  });
  $("a").mouseout(function(event){
    $("#capa").removeClass("clasecss");
  });
});
</script>
</head>

<body>

<div id="capa">
Esta capa es independiente y voy a añadir y eliminar clases css sobre ella
</div>

<br>
<br>

<a href="http://www.desarrolloweb.com">Añadir y quitar clase en la capa de arriba</a>
</body>
</html>
```

Podemos ver el ejemplo desarrollado en el artículo en una página aparte.

*Artículo por **Miguel Angel Alvarez***

Mostrar y ocultar elementos de la página con jQuery

Para aprender rápidamente cómo hacer cosas típicas con jQuery, vamos a explicar cómo mostrar u ocultar elementos de la página, cambiando propiedades de las hojas de estilo en cascada. Para que este artículo tenga un poco más de practicidad, vamos a realizar un ejemplo de formulario donde algunos de los elementos están escondidos. En ese formulario, al marcar una opción de un campo checkbox, se mostrarán esos elementos escondidos y al desmarcar esa opción, se ocultarán.

El método que se dispone en jQuery para alterar las hojas de estilo se llama css() y lo podemos invocar sobre elementos de la página, a los que queremos cambiar sus propiedades CSS. En principio, para mostrar y ocultar elementos, nos viene bien alterar el atributo "display", poniendo el valor "none" para que no aparezca y "block" para que sí lo haga.

Lo que deberíamos saber para poder entender este ejemplo se puede aprender en el Manual de jQuery que venimos publicando en DesarrolloWeb.com. Aunque como este es un ejemplo básico, será suficiente con estudiar el artículo Pasos fundamentales para usar jQuery.

Ocultar y mostrar una capa con jQuery

Vamos a ver brevemente cómo usar el mencionado método css(). Primero, tendríamos que tener un elemento en la página, que es el que vamos a mostrar u ocultar.

```
<div id="mielemento">
Contenido del elemento...
</div>
```

Para ocultar este elemento, habría que invocar el método css() de la siguiente manera:

```
$("#mielemento").css("display", "none");
```

Como se puede ver, se accede al elemento con la función dólar $() y el selector "#mielemento". Luego al método css() le pasamos el valor "display" y "none", porque queremos alterar a propiedad display y asignarle el valor "none", para ocultar el elemento.

Para mostrarlo haríamos algo parecido, pero colocando el valor "block" en el atributo CSS "display".

```
$("#mielemento").css("display", "block");
```

Nota: el método css() admite otros parámetros. Si sólo recibe un parámetro, de tipo string, devuelve el valor CSS asignado a ese parámetro. También podría recibir un sólo parámetro, en este caso de con una notación de objeto con pares llave/valor, y entonces asignaría todos esos estilos CSS, especificados por los pares llave/valor en el objeto, al elemento de la página donde se haya invocado el método.

Mostrar u ocultar elementos como respuesta a un evento

Ahora que ya sabemos cómo realizar un cambio en el atributo display, vamos a ver cómo poner esta instrucción en marcha cuando el usuario realice acciones en la página. Recordar que al principio del artículo comentaba que íbamos a crear un formulario con una casilla de selección (campo checkbox) y que al activar ese campo se mostrarían otros contenidos en el formulario. Al desactivarlo, se ocultarían esos contenidos del formulario.

Para entender bien lo que deseamos hacer, podemos ver el ejercicio en marcha en una página aparte.

Lo primero que podemos presentar es el formulario con el que vamos a trabajar.

```
<form>
Nombre: <input type="text" name="nombre">
<br>
<input type="checkbox" name="mayor_edad" value="1" id="mayoria_edad"> Soy mayor de edad
<br>
<div id="formulariomayores" style="display: none;">
Dato para mayores de edad: <input type="text" name="mayores_edad">
</div>
</form>
```

Como se podrá ver, es un formulario como otro cualquiera. Sólo que tiene una capa con id="formulariomayores", que contiene los elementos del formulario que queremos mostrar u ocultar al marcar o desmarcar el checkbox. Esa capa estará inicialmente oculta, y para ello hemos colocado el atributo style="display: none;". Podemos fijarnos también en el checkbox con id="mayoria_edad", que es el que va servir para marcar si se desea o no ver la parte final del formulario.

Ahora hay que hacer cosas cuando se haga clic en el checkbox con id="mayoria_edad". Para

ello en deberíamos crear un código Javascript y jQuery como este:

```
$(document).ready(function(){
  $("#mayoria_edad").click(function(){
    //lo que se desee hacer al recibir un clic el checkbox
  });
});
```

Como ya hemos comentado, estas líneas sirven para especificar eventos. $(document).ready() se hace para lanzar instrucciones cuando el navegador está preparado para recibirlas y $("#mayoria_edad").click() sirve para realizar acciones cuando se ha hecho clic sobre el elemento con id "mayoria_edad", que es el checkbox del formulario. (Lee el artículo Pasos para trabajar con jQuery para más información sobre este punto).

Ahora tenemos que ver las instrucciones que debemos ejecutar como respuesta a ese evento.

```
if ($("#mayoria_edad").attr("checked")){
  $("#formulariomayores").css("display", "block");
}else{
  $("#formulariomayores").css("display", "none");
}
```

Básicamente lo que hacemos es comprobar el estado del atributo "checked" del elemento "#mayoria_edad". Esto se hace con el método attr() indicando como parámetro el valor del atributo HTML que queremos comprobar. Entonces, si estaba "checked", se tiene que mostrar el elemento y si no estaba marcado el checkbox, habría que ocultarlo.

Espero que se haya entendido bien. Ahora dejo aquí el código completo de este ejemplo:

```
<html>
<head>
  <title>Mostrar Ocultar</title>
<script src="jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
  $("#mayoria_edad").click(function(evento){
    if ($("#mayoria_edad").attr("checked")){
      $("#formulariomayores").css("display", "block");
    }else{
      $("#formulariomayores").css("display", "none");
    }
  });
});
</script>
</head>

<body>

<form>
Nombre: <input type="text" name="nombre">
<br>
<input type="checkbox" name="mayor_edad" value="1" id="mayoria_edad"> Soy mayor de edad
<br>
<div id="formulariomayores" style="display: none;">
Dato para mayores de edad: <input type="text" name="mayores_edad">
</div>
</form>

</body>
</html>
```

Finalmente, podemos verlo en marcha en una página aparte.

*Artículo por **Miguel Angel Alvarez***

Efectos rápidos con jQuery

Una de las ventajas más destacadas de jQuery es la realización de efectos especiales para páginas web, que se desarrollan rápido y con poco código fuente. Estos efectos sirven para aplicar dinamismo a una página web y una respuesta atractiva frente la interacción con el usuario, lo que hace que las páginas programadas con jQuery ofrezcan una imagen puntera.

Los efectos con jQuery, al menos un buen puñado de ellos, se pueden realizar sin muchas complicaciones, ya que existen unas funciones que simplifican la tarea de los desarrolladores (Ver la librería Effects). En muchos casos conseguir un efecto nos llevará una línea de código en nuestro programa, como esta:

```
$("#capaefectos").hide("slow");
```

Con esto conseguimos que el elemento con id="capaefectos" desaparezca de la página. Pero además, el efecto no es un simple fundido del elemento en la página (hacerse transparente), sino que también va acompañado de una reducción de tamaño progresiva hasta desaparecer.

Combinando los efectos con la interacción de usuario, por medio de eventos, podemos conseguir que los efectos respondan a las acciones del visitante, lo que multiplica las posibilidades, manteniendo la sencillez, elegancia y facilidad de manutención del código Javascript. Lo vamos a ver en un ejemplo a continuación.

Ejemplo de efectos e interacción en jQuery

En el siguiente ejemplo vamos a mostrar un uso sencillo de las funciones de efectos de jQuery. Vamos a implementar un simple efecto de ocultar y mostrar un elemento de la página web.

Podemos ver el ejemplo en marcha en una página aparte.

Como hemos podido ver, vamos a tener una capa y un par de enlaces. Con jQuery haremos que al pulsar los enlaces se oculte y se muestre la capa, con las funciones de la librería Effects.

Para comenzar, este es el código HTML del ejemplo, que comprende tanto la capa como los enlaces.

```
<div id="capaefectos" style="background-color: #cc7700; color:fff; padding:10px;">
Esto es una capa que nos servirá para hacer efectos!
</div>

<p>
<a href="#" id="ocultar">Ocultar la capa</a> |
<a href="#" id="mostrar">Mostrar la capa</a>
</p>
```

Ahora viene la parte interesante, que es en la que asociamos eventos a estos dos enlaces y codificamos las llamadas a las funciones de Effects, que harán que se muestre y oculte la capa.

El código Javascript, que hace uso de jQuery sería el siguiente:

```
$(document).ready(function(){
  $("#ocultar").click(function(event){
   event.preventDefault();
   $("#capaefectos").hide("slow");
  });

  $("#mostrar").click(function(event){
```

```
    event.preventDefault();
    $("#capaefectos").show(3000);
   });
});
```

Como se puede ver, primero tenemos que definir el evento ready del objeto $(document), para hacer cosas cuando el documento está preparado para recibir acciones.

Luego se define el evento click sobre cada uno de los dos enlaces. Para ello invoco el método click sobre el enlace, que hemos seleccionado con jQuery a través del identificador de la etiqueta A.

```
$("#ocultar").click(function(event){
```

Con esto estoy definiendo el evento clic sobre el elemento con id="ocultar".

> **Nota:** leer el artículo anterior Pasos para utilizar jQuery en tu página web, en el que hablábamos sobre eventos y otras generalidades de este framework Javascript. Podremos encontrar explicaciones más detalladas sobre cómo definir eventos Javascript con jQuery.

Dentro de la función a ejecutar cuando se hace clic, se coloca la llamada a la función de los efectos.

```
$("#capaefectos").hide("slow");
```

Esto hace que nuestra capa, a la que habíamos puesto el identificador (atributo id) "capaefectos", se oculte. Pasamos el parámetro "slow" porque queremos que el efecto sea lento.

Ahora veamos la función de los efectos con otra llamada:

```
$("#capaefectos").show(3000);
```

Esto hace que se muestre el elemento con id "capaefectos", y que el proceso de mostrarse dure 3000 milisegundos.

No hay más complicaciones, así que si habéis entendido esto ya sabéis hacer efectos simples pero atractivos con jQuery en vuestra página web. Ahora podréis ver el código completo de este ejemplo creado por DesarrolloWeb.com para demostrar el uso de efectos.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>
<head>
   <title>Efectos con jQuery</title>
   <script src="jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
   $("#ocultar").click(function(event){
    event.preventDefault();
    $("#capaefectos").hide("slow");
   });

   $("#mostrar").click(function(event){
    event.preventDefault();
    $("#capaefectos").show(3000);
   });
});
</script>
</head>
```

```
<body>

<div id="capaefectos" style="background-color: #cc7700; color:fff; padding:10px;">
Esto es una capa que nos servirá para hacer efectos!
<br>
<br>
Pongo este texto simplemente de prueba
</div>

<p>
<a href="#" id="ocultar">Ocultar la capa</a> |
<a href="#" id="mostrar">Mostrar la capa</a>
</p>

</body>
</html>
```

Por último, pongo el enlace de nuevo al ejemplo en marcha.

Como se ha podido comprobar, hacer efectos con jQuery es bastante sencillo. Claro que hay otros detalles importantes y otros tipos de efectos y funcionalidades de personalización de los mismos, pero esto nos ha servido para demostrar lo sencillo que es trabajar con este framework Javascript. En siguientes artículos seguiremos explorando casos de uso típicos de jQuery.

*Artículo por **Miguel Angel Alvarez***

Callback de funciones jQuery

A menudo cuando hacemos aplicaciones enriquecidas del lado del cliente con jQuery nos vemos en la necesidad de encadenar varias llamadas a funciones, para que una se ejecute detrás de otra, creando un efecto más elaborado. En este artículo veremos lo sencillo que es realizar lo que en inglés se llama "callback", es decir una función que se ejecuta después de otra.

Apilar funciones, para que se ejecuten una detrás de otra, nos servirá para hacer muchas cosas. En nuestro día a día con jQuery iremos encontrando la utilidad, pero de momento para explicar un caso en el que nos resultará imprescindible, se me ocurre que deseemos hacer una secuencia de efectos y cambios dinámicos en un elemento.

Por ejemplo imaginemos que se desea ocultar una capa con un efecto de fundido (de opaco a transparente), luego moverla a otra posición y volverla a mostrar (ya en la nueva posición) con otro efecto de fundido (en este caso de transparente a opaco). En principio podríamos pensar en hacer un código como este:

```
$("#micapa").fadeOut(2000);
$("#micapa").css({top: 300, left:200});
$("#micapa").fadeIn(2000);
```

En este caso estamos alterando las propiedades de una capa con id="micapa". Primero llamamos a fadeOut() para ocultarla con un fundido, que durará 2 segundos (véase el parámetro 2000, que son los milisegundos que durará el efecto). Luego alteramos la posición de la capa, cambiando sus atributos CSS. Para acabar la volvemos a mostrar con un fundido de otros 2000 milisegundos.

Nota: para poder entender mejor estas llamadas a efectos, por favor, consulta el artículo Efectos Rápidos con jQuery.

Si lanzamos la ejecución de estas sentencias, tal como aparece en el código, será como si se ejecutasen todas a la vez. Como los fadeOut y fadeIn tardarán 2 segundos en ejecutarse y los cambios de las propiedades CSS top y left son inmediatos, lo que ocurrirá será que primero veremos la capa moverse a la nueva posición y luego veremos los dos efectos de fundido.

Lo mejor para darse cuenta de este caso es verlo en marcha.

Cómo realizar una pila de ejecución de funciones

Ahora que ya hemos visto uno de los casos en los que necesitaríamos ejecutar funciones en una pila, una después de otra, esperando a que termine completamente la ejecución de cualquier efecto o acción antes de comenzar con la siguiente. Vamos a ver cómo hacerlo con jQuery.

Simplemente tenemos que saber que todas las funciones o métodos de jQuery pueden recibir un parámetro adicional con el nombre de la función que se tiene que ejecutar después que termine el procesamiento de la primera. Esa segunda función que se ejecuta después de la primera es la que se conoce en inglés por callback. Un ejemplo sencillo para entenderlo.

```
miFuncion ("parametros de la función", funcionCallback);
```

En ese esquema de llamada a miFuncion(), se le están pasando dos parámetros. El primero sería un supuesto parámetro que necesitase miFuncion() y el segundo, que es le que nos interesa en este caso, el nombre de la función que se tiene que ejecutar después que acabe. Con este código, primero se ejecuta miFuncion() y cuando acaba completamente, se ejecuta funcionCallback(). Pero atención que este ejemplo lo hemos simplificado para que se pueda entender fácilmente y esta sintaxis sólo valdrá si funcionCallback no recibe parámetros, porque no los podemos indicar con el nombre de la función. Veamos entonces una forma de hacer este callback que funcione siempre:

```
miFuncion ("parametros de la funcion", function(){
  funcionCallback();
});
```

Con este código, que funcionaría igual que el anterior, lo bueno es que sí podemos indicar los parámetros que se necesiten para la llamada a funcionCallback().

Ejemplo real de callback con jQuery

Ahora que hemos aprendido toda la teoría, veamos un ejemplo práctico que solucionaría el problema comentado anteriormente sobre el procesamiento de diversos efectos y cambios en las propiedades de los objetos, para que se hagan siempre en la secuencia que deseamos. Se trata simplemente de aplicar las llamadas con callback que hemos antes.

```
$("#micapa").fadeOut(1000, function(){
  $("#micapa").css({'top': 300, 'left':200});
  $("#micapa").fadeIn(1000);
});
```

Como se puede ver, en la llamada a fadeOut() estamos pasando como parámetros el valor 1000, que son los milisegundos tiene que durar el efecto fade out (fundido hacia transparente), y luego la función callback, que se ejecutará después de que fadeOut() haya terminado.

Como el método css() (se encuentra como primera instrucción de la función callback) es instantáneo, no necesita hacerse un callback para ejecutar el fadeIn(), sino que se puede escribir directamente en la siguiente línea de código. Así pues, se ve que el callback, al menos en este ejemplo, sólo es necesario hacerlo cuando se ejecutan funciones que realizarán un procesamiento prolongado.

Podemos ver este ejemplo de callback en una página aparte.

Hasta aquí, a lo largo de los capítulos de este manual de jQuery, hemos leído la introducción a este popular framework Javascript, tal como se puede ver en el apartado "How to use jQuery" publicada en la página de documentación. Desde DesarrolloWeb.com hemos enriquecido este tutorial de jQuery aportando nuevos ejemplos y explicaciones adicionales, encaminadas a que cualquier persona, con unos conocimientos básicos de Javascript, pueda entender y aprender a usar estas librerías de programación cross-browser. Ahora podemos hacer una pausa en este manual y volveremos pronto con nuevos artículos para explicar otros asuntos sobre la programación con jQuery.

*Artículo por **Miguel Angel Alvarez***

Uso de Ajax muy sencillo con jQuery

Ha llegado el momento de hacer una primera aproximación a Ajax, en la serie de artículos que estamos publicando en DesarrolloWeb.com para mostrar el uso de este framework (léase el Manual de jQuery).

Una de las ventajas de los frameworks Javascript es que nos permiten desarrollar scripts que hacen uso de Ajax de una manera muy fácil y encima, sin tener que complicarnos la vida con la compatibilidad entre distintos navegadores. Sin duda, cualquier persona que sepa un poco de Javascript, podría en poco tiempo empezar a utilizar Ajax con alguno de estos frameworks. Nosotros vamos a demostrar cómo es de sencillo en jQuery.

La primera impresión que he tenido sobre el uso de Ajax en jQuery es realmente grata, por la facilidad con la que se puede realizar un primer ejemplo. Se trata de un ejemplo extremadamente sencillo, pero sirve para abrirnos las puertas a muchas aplicaciones interesantes. Habíamos visto en otros frameworks Javascript ejemplos similares, como en el artículo Ajax con Mootools, pero tenemos que quitarnos el sombrero ante la extremada sencillez con la que se puede hacer un ejemplo similar en jQuery. Sea suficiente decir que en este ejemplo de Ajax utilizaremos tan sólo 4 líneas de código, de las cuales sólo 1 es para ejecutar la propia llamada al servidor por Ajax.

Traer un contenido con Ajax al pulsar un enlace

En este sencillo ejemplo haremos llamada a Ajax, para traer un contenido, cuando se pulse un enlace. Esto lo hemos puesto en marcha en el servidor de DesarrolloWeb.com y lo puedes ver en una página aparte.

Aquí podemos ver el enlace, al que ponemos un identificador (atributo id) para seleccionarlo desde jQuery.

```
<a href="#" id="enlaceajax">Haz clic!</a>
```

Si hemos leído hasta aquí el Manual de jQuery podremos saber cómo asignar un evento a un

enlace. No obstante, recordemos cómo asignar una función para cuando se haga clic en el enlace:

```
$(document).ready(function(){
  $("#enlaceajax").click(function(evento){
    //elimino el comportamiento por defecto del enlace
    evento.preventDefault();
    //Aquí pondría el código de la llamada a Ajax
  });
})
```

Ya se trata sólo de poner en marcha Ajax dentro de la función del evento "click" sobre el enlace. Pero antes voy a necesitar una capa donde mostrar el contenido que vamos a recibir del servidor con la llamada Ajax. Le pondremos id="destino" para poder referirnos a ella desde jQuery:

Y ahora la parte más interesante, donde podemos ver qué tan fáciles son las cosas con este framework Javascript. Una única línea de código será suficiente:

```
$("#destino").load("contenido-ajax.html");
```

Con esta simple sentencia estamos realizando una llamada a Ajax con jQuery. Es una simple invocación al método load() de un elemento de la página, en concreto el que habíamos puesto con id="destino". Al método load() le pasamos como parámetro la ruta de la página que queremos cargar dentro del elemento.

El archivo que cargamos con load() en este ejemplo es "contenido-ajax.html" y simplemente le hemos colocado un texto cualquiera. Lo hemos guardado en el mismo directorio que la página web donde está el script jQuery.

> **Nota:** para que este ejemplo funcione debe colocarse en un servidor web, puesto que la llamada por Ajax se hace por http y el archivo que se carga entonces tiene que recibirse por un servidor web, que lo mande con ese protocolo al navegador. Si pones los archivos en tu disco duro y los ejecutas tal cual, no te funcionará. Puedes utilizar cualquier espacio de hosting que tengas o bien un servidor web que puedas tener instalado en tu ordenador.

Así de simple! Podemos ver el código completo de este ejemplo:

```
<html>
<head>
  <title>Ajax Simple</title>
<script src="jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
  $("#enlaceajax").click(function(evento){
    evento.preventDefault();
    $("#destino").load("contenido-ajax.html");
  });
})
</script>
</head>
<body>

<a href="#" id="enlaceajax">Haz clic!</a>
<br>
<div id="destino"></div>

</body>
</html>
```

Podemos ver el ejemplo en marcha en una página aparte.

Cabría comentar que jQuery tiene muchos otros métodos de realizar conexiones por Ajax, que pueden servir para muchos otros casos de trabajo que podemos encontrarnos. Nosotros hemos escogido el más sencillo para dar una primera demostración de las posibilidades.

Pasar parámetros y ejecutar acciones después de la llamada a Ajax

El método load() que hemos visto en el ejemplo anterior tiene otros dos parámetros opcionales que podemos utilizar si fuera necesario:

Parámetros a pasar a la página: la página que carguemos con Ajax puede recibir parámetros por la URL, que se especifican con la típica notación de propiedades y valores de jQuery.

```
{nombre: "Pepe", edad: 45}
```

Por ejemplo, con ese código estaríamos enviando a la página los datos nombre y edad, con los valores "pepe" y 45. Esos datos viajarían en la URL, por el método "POST".

> **Nota:** Desde jQuery 1.3 también se pueden enviar los parámetros a la página a cargar con Ajax por medio de una variable de tipo string, en lugar de una notación de objetos como hemos comentado. Cuando se use un string para especificar los parámetros a enviar en el request http, éstos viajan por el método GET. Cuando se utiliza una notación de objetos como la que hemos visto, los datos viajan por el método POST.

Función callback: como otros métodos de jQuery, podemos especificar opcionalmente una función a ser ejecutada cuando se termine de ejecutar el método. En este caso, cuando se termine la llamada Ajax, se pueden hacer acciones, como borrar un mensaje típico de "cargando...".

> **Nota:** En un artículo anterior ya comentamos el habitual uso de funciones callback en jQuery.

Ahora veamos un código donde hacemos uso de estos dos parámetros:

```
$(document).ready(function(){
  $("#enlaceajax").click(function(evento){
    evento.preventDefault();
    $("#destino").load("recibe-parametros.php", {nombre: "Pepe", edad: 45}, function(){
      alert("recibidos los datos por ajax");
    });
  });
})
```

En este caso estamos cargando con load() una página PHP llamada "recibe-parametros.php". Estamos pasando los parámetros "nombre" y "edad" a una página, que podremos recoger por GET. Además, hemos colocado una función callback en la que simplemente hacemos un alert(), que se ejecutará cuando la llamada a Ajax haya terminado.

Este sería el código fuente de "recibe-parametros.php":

```
Recibido el siguiente dato:
<br>
Nombre: <?php echo $_POST["nombre"];?>
<br>
Edad: <?php echo $_POST["edad"];?>
```

Podemos ver este ejemplo en una página aparte.

Con esto hemos podido comprobar lo sencillo que es realizar con jQuery una carga de contenidos que se reciben por Ajax. Como decía, existen muchas otras maneras de hacer conexiones Ajax con jQuery, como el ejemplo del artículo siguiente que nos enseña a mostrar un mensaje de carga miestrás esperamos la respuesta Ajax del servidor. Además, para complementar esta información, también podéis ver el vídeo de Ajax con jQuery.

*Artículo por **Miguel Angel Alvarez***

Ajax jQuery con mensaje de carga

Queremos ver algunas cosas típicas que podríamos desear hacer con Ajax en una página web, facilitando el proceso de desarrollo con el framework Javascript jQuery. En esta línea de artículos ya publicamos hace poco un artículo sobre el uso de Ajax en jQuery simplificado. En el mencionado artículo vimos cómo hacer una llamada Ajax con 1 sola línea de código (el Ajax en si era una línea de código, aunque se necesitan varias instrucciones más para asociar las acciones Ajax como respuesta a eventos en la página).

Una de las cosas que más habitualmente podríamos desear hacer cuando se realiza una llamada Ajax es la creación de un mensaje de carga, que podemos colocar con un simple texto "cargando..." o con la típica imagen de Ajax Loader. En este artículo veremos cómo crear ese mensaje de carga al realizar una solicitud Ajax con jQuery.

Para los interesados, se puede ver este ejemplo de Ajax en una página aparte.

Por qué un mensaje de carga al hacer Ajax

Cuando hacemos una solicitud por Ajax, los resultados de dicha llamada a veces tardan en llegar. Durante ese tiempo el usuario puede no ver ninguna reacción por parte del navegador, lo que le puede confundir y pensar que no ha hecho clic correctamente en el enlace o botón. Sería normal en ese caso que el usuario pulse repetidas veces un enlace o un botón de envío de formulario, generando repetidas e innecesarias llamadas al servidor, lo que puede derivar en diversos problemas.

Así pues, es conveniente mostrar un mensaje de carga para advertir que su solicitud fue realizada y el proceso está en marcha y esperando respuesta del servidor.

Vamos a explicar la implementación de este mensaje de carga, pero siempre teniendo en cuenta que nuestra intención en este ejemplo es mantener un código pequeño que se pueda entender fácilmente. Aunque queremos remarcar que las cosas se podrían hacer de otra manera, algo mejorada, cuando sepamos más cosas sobre el framework Javascript jQuery y pongamos en marcha algunas prácticas aconsejables de programación orientada a objetos.

Preparando el código HTML

Como un primer paso, vamos a mostrar el código HTML que tendremos en la página que hará la solicitud Ajax.

```
<a href="#" id="enlaceajax">Haz clic!</a>
```

```
<div id="cargando" style="display:none; color: green;">Cargando...</div>
<br>
<div id="destino"></div>
```

Como se puede ver, tenemos tres elementos: 1) el enlace, que activará la llamada a Ajax cuando se haga clic sobre él. 2) una capa con id="cargando" que es donde está el mensaje de carga (nosotros hemos colocado un texto, pero se podría colocar cualquier cosa, como el típico gif animado que muestra que se está procesando la solicitud (conviene fijarse también que esa capa cargando está oculta de momento, gracias al atributo de estilo CSS display:none). 3) la capa "destino", donde mostraremos la respuesta recibida tras la solicitud Ajax.

Llamada a Ajax con jQuery y el mensaje de carga

En este punto vamos a describir cómo se tendría que hacer la llamada al servidor con Ajax, pero lo cierto es que este proceso ya lo explicamos con detalle anteriormente, por lo que os refiero al artículo Uso de Ajax muy sencillo con jQuery, donde encontraréis unas explicaciones que voy a dar por sabidas en este caso. Lo que explicaré en este artículo es cómo se tiene que mostrar el mensaje de carga cuando se inicia la llamada y como eliminarlo una vez hemos recibido la respuesta por Ajax.

Otra cosa que el lector deberá conocer para poder entender este ejemplo es Mostrar y ocultar elementos de la página con jQuery.

```
$(document).ready(function(){
  $("#enlaceajax").click(function(evento){
    evento.preventDefault();
    $("#cargando").css("display", "inline");
    $("#destino").load("pagina-lenta.php", function(){
      $("#cargando").css("display", "none");
    });
  });
})
```

Voy comentando línea a línea el código anterior.

En la primera línea se está especificando un método ready() sobre el objeto document, que sirve para generar un código a ser ejecutado cuando la página está lista para recibir instrucciones Javascript que trabajen con el DOM.

En la segunda línea se accede al elemento con identificador "enlaceajax" (es decir, el enlace que activará el Ajax) para definir un evento "click".

En la siguiente línea se ejecuta el método preventDefault() sobre el evento producido al hacer clic sobre el enlace. Esto se hace para anular el comportamiento típico del enlace.

Ahora viene la parte en la que se mostrará el mensaje de carga:

```
$("#cargando").css("display", "inline");
```

Simplemente se muestra la capa con id="cargando", con un simple cambio en el atributo CSS display de la capa. Ese cambio de atributo lo hacemos con el método css() sobre el elemento que queremos alterar, tal como se vio en el artículo Mostrar y ocultar elementos de la página con jQuery.

Ahora, con la siguiente línea de código:

```
$("#destino").load("pagina-lenta.php", function(){
```

Se hace la llamada Ajax, con el método load() sobre la capa que queremos actualizar con el contenido traído por Ajax, que es la capa con id="destino". Este método recibe la URL de la página a cargar y una función callback, que se ejecutará después que el método load() se haya

terminado de procesar, esto es, después de que la solicitud Ajax se haya recibido y se haya mostrado su contenido en la capa que recibe el método.

Entonces, en esa función callback, tenemos que ocultar la capa con el mensaje de carga, puesto que cuando se ejecute esta función ya se habrá realizado todo el procesamiento Ajax. Eso lo conseguimos con el método css(), alterando la propiedad display, de manera similar a como lo habíamos realizado para mostrar la capa cargando.

```
$("#cargando").css("display", "none");
```

Esto es todo. Realmente no tiene ninguna complicación especial. Aunque, como decía, estas cosas se podrán hacer de una manera más elegante cuando aprendamos un poquito más sobre jQuery.

Por si sirve para aclarar las cosas, dejo a continuación el código completo de la página que hace la solicitud con jQuery:

```
<html>
<head>
  <title>Ajax Simple</title>
<script src="jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
  $("#enlaceajax").click(function(evento){
    evento.preventDefault();
    $("#cargando").css("display", "inline");
    $("#destino").load("pagina-lenta.php", function(){
      $("#cargando").css("display", "none");
    });
  });
})
</script>
</head>

<body>
Esto es un Ajax con un mensaje de cargando...
<br>
<br>

<a href="#" id="enlaceajax">Haz clic!</a>
<div id="cargando" style="display:none; color: green;">Cargando...</div>
<br>
<div id="destino"></div>

</body>
</html>
```

Código de la página PHP que se invoca por Ajax

El código de la página PHP que traemos por ajax "pagina-lenta.php" es el siguiente:

```
<?php
sleep(3);
echo ("He tardado 3 segundos en ejecutar esta p&aactute;gina...");
?>
```

En realidad no tiene nada en especial. Simplemente hacemos un sleep(3) para ordenarle a PHP que espere 3 segundos antes de terminar de procesar la página y enviarla al cliente. Así consigo que la solicitud http tarde un poco es ser respondida y podamos ver el mensaje de carga durante un poco más de tiempo en la página.

Finalmente, pongo de nuevo el enlace para ver este ejercicio en marcha.

Si te ha interesado este ejemplo y quieres obtener una ayuda adicional para crear tus propios scripts en Ajax, te recomendamos ver el vídeo donde explcamos a hacer Ajax con jQuery.

*Artículo por **Miguel Angel Alvarez***

Videotutorial: Ajax en jQuery

Este videotutorial está indicado para las personas que quieren dar sus primeros pasos con jQuery y desean ver cómo se hace una de las tareas más solicitadas dentro de la programación con este popular framework Javascript. Se trata de una conexión Ajax, realizada paso a paso, para que todas las personas puedan entenderla.

En el vídeo mostraremos cómo realizar la página web que hará la conexión Ajax, desde el principio, incluyendo el script del framework jQuery y luego programando el script de la llamada a Ajax. Además, por complicar un poco más el ejemplo, hemos implementado directamente un mensaje de carga, que se mostrará cuando se inicie la conexión y se ocultará una vez que haya terminado con éxito.

Este vídeo ilustra de manera entendible todo el proceso que ya habíamos explicado en anteriores artículos de DesarrolloWeb.com, concretamente dentro del Manual de jQuery que ponemos a disposición de los lectores del sitio. Los artículos que deberían leerse como complemento a este vídeo son Uso de Ajax muy sencillo con jQuery y el siguiente capítulo del manual llamado Ajax jQuery con mensaje de carga.

El proceso de realización de una solicitud Ajax en jQuery es extremadamente simple, puesto que con una única línea de código se implementa toda la llamada por Ajax e incluso el volcado de la respuesta en un elemento de la página. Sin embargo, se requieren diversas líneas de código adicionales para realizar cosas como la ejecución de código sólo cuando el navegador está listo para recibir acciones que modifiquen el DOM, la codificación de un evento para que se solicite la página por Ajax como respuesta a un clic del usuario que visita la página, etc. Además, el script, como decíamos, también realiza la tarea de mostrar un mensaje de carga mientras que la solicitud está pendiente, con lo que hay que agregar alguna línea de código adicional también por este motivo. Luego haremos un pequeño script PHP que tenga un retardo, para que la solicitud Ajax tarde un poco más en obtener respuesta y podamos llegar a ver el mensaje de carga.

Como se podrá ver en el videotutorial, esta tarea no tiene mucha dificultad, pero serán necesarios conocimientos de varios ámbitos para poder entenderla toda.

Por cierto, durante el vídeo trabajamos con un editor de texto para programadores llamado Komodo Edit, lo que puede resultar de interés para las personas que no conozcan este editor de código fuente gratuito. Komodo Edit tiene un previsualizador que nos permite ver en ejecución, durante las primeras pruebas, la página que vamos realizando. Aunque la parte en la que trabajamos con PHP la tenemos que ejecutar a través de un servidor web y luego ver en un navegador. Nosotros como servidor web utilizamos Wamp, que instala en un solo paso todo lo necesario para ejecutar páginas PHP.

El vídeo lo hemos publicado en YouTube y producido en menos de 10 minutos, por requerimientos de dicho sitio web. No obstante, creemos que ha quedado un videotutorial suficientemente claro y con las necesarias explicaciones para que todas las personas puedan entenderlo prestando suficiente atención.

Sin más, os dejamos con este vídeo que esperamos pueda servir de utilidad a todas las

personas que quieren desarrollar sus páginas con Ajax y utilizar una de las herramientas más populares para facilitarnos la vida al programar páginas con scripts del lado del cliente.

*Artículo por **Miguel Angel Alvarez***

Core de jQuery

Hasta este momento ya hemos publicado varios artículos en el Manual de jQuery de DesarrolloWeb.com, pero realmente lo que hemos visto es una mera introducción a las posibilidades del API, así como unas explicaciones básicas para empezar a trabajar con jQuery en nuestras páginas web. Sin embargo, nos gustaría ir publicando una documentación en español completa, para que cualquier desarrollador pueda aprender a fondo este framework Javascript y crear cualquier tipo de aplicación enriquecida del cliente.

Las personas con un nivel medio-alto de Javascript y de programación orientada a objetos, con lo que hemos visto hasta el momento en el manual y la referencia del API jQuery, podrán con relativa facilidad adentrarse en características avanzadas del framework, pero para los demás y los que deseen una ayuda adicional, esperamos que este artículo y los siguientes puedan abrir un camino, sencillo de seguir, para aprender a hacer todo tipo de desarrollos con Javascript y jQuery.

Comencemos pues por el inicio, tal como está en la documentación del sistema, tratando en primer caso el "Core". De todos modos, cabe decir que lo que hemos podido ver en los artículos anteriores de este Manual de jQuery, nos va a venir muy bien para poder entender todo lo que viene a continuación y sobre todo, poder poner en marcha ejemplos un poco más elaborados.

Core de jQuery

El core de jQuery es la base sobre la que se trabaja para hacer cualquier cosa con jQuery. Contiene una serie de clases y métodos útiles para hacer tareas reiterativas, que vamos a necesitar en las aplicaciones. Integra desde funciones que serán útiles en cualquier script, por sencillo que sea, hasta funciones menos recurridas pero que nos facilitarán la vida a hora de hacer código limpio, corto y reutilizable.

Utilizaremos el Core para realizar cosas útiles con objetos, clases, datos, etc, pero realmente lo que más haremos será utilizar la función jQuery, que es el pilar fundamental sobre el que se basarán las aplicaciones.

Core tiene las funciones clasificadas en diversos apartados:

$() (La funcion jQuery):
Es la función principal de jQuery, que además tiene diversas utilidades según los parámetros que se le envíen. Su utilidad principal es obtener elementos de la página.

Accesorios de objetos:
Es una gama de funciones de diversa y variada utilidad, que sirven de utilidad para hacer cosas con objetos, tales como iterar con cada uno de sus elementos, saber su tamaño, longitud, el selector o contexto con el que se obtuvo, obtener todos sus elementos DOM que contenga, etc.

Trabajo con datos:
Unas funciones útiles para trabajar con datos y asociarlos a elementos, una forma de guardar información adicional a elementos de la página. También tiene diversas funciones para trabajar con colas y administrar la salida y entrada de sus elementos.

Plugins:
Funciones que permiten extender los elementos jQuery para incorporar nuevos métodos, algo que se utiliza habitualmente a la hora de crear plugins para añadir funcionalidades a jQuery.

Interoperabilidad:
Funciones que permiten que jQuery no tenga problemas de compatibilidad con otras librerías Javascript que también suelen utilizar la función dólar $().

> **Nota:** En el momento de escribir este manual estamos en la release 1.3.2, en la que se han publicado como novedad un par de métodos, de los clasificados en accesorios de objetos. Como debemos saber, de vez en cuando actualizan las librerías para incorporar nuevos métodos.

En el próximo artículo comenzaremos a tratar a fondo el Core de jQuery, hablando de la función dólar $().

*Artículo por **Miguel Angel Alvarez***

Función jQuery o función $()

Con el objetivo de seguir tratando el Core de jQuery, del que ya empezamos a hablar en este Manual de jQuery, vamos a explicar la función jQuery, también conocida como $().

El funcionamiento del Core de jQuery se basa en esta función. Como dicen en la propia documentación del framework, "Todo en jQuery está basado en esta función o la usa de alguna

forma".

La función jQuery sirve para hacer varias cosas, según los parámetros que le pasemos. El uso más simple es seleccionar elementos o grupos de elementos de una página web, pero también sirve para crear elementos sobre la marcha o para hacer un callback de funciones.

En realidad esta función se invoca también con el símbolo dólar $(), lo que sería una manera resumida de utilizarla y que nos ahorrará bastantes caracteres en el código y a la postre, tiempo de descarga y peso de los scripts de la página.

Veamos los distintos usos de esta función, según los parámetros que reciba.

Función jQuery enviando un selector y un contexto

Este uso de la función sirve para seleccionar elementos de la página. Recibe una expresión y se encarga de seleccionar todos los elementos de la página que corresponden con ella, devolviendo un **objeto jQuery** donde se encuentran todos los elementos de la página seleccionados y extendidos con las funcionalidades del framework. Adicionalmente, podemos pasarle un contexto de manera opcional. Si no se le envía un contexto, selecciona elementos del documento completo, si indicamos un contexto conseguiremos seleccionar elementos sólo dentro de ese contexto.

La expresión que debemos enviar obligatoriamente como primer parámetro sirve de selector. En ese parámetro tenemos que utilizar una una notación especial para poder seleccionar elementos de diversas maneras y la verdad es que las posibilidades de selección con jQuery son muy grandes, como veremos en el tema de "Selectores".

Este paso de selección de elementos es básico en cualquier script jQuery, para poder actuar en cualquier lugar de la página y hacer nuestros efectos y utilidades Javascript sobre el lugar que deseemos.

Veamos un uso de esta función:

```
var elem1 = $("#elem1");
```

Con esta línea de código habremos seleccionado un elemento de la página que tiene el identificador (atributo id del elemento HTML) "elem1" y lo hemos guardado en una nueva variable llamada elem1. La variable elem1 guardará entonces lo que se llama el objeto jQuery con el cual podremos trabajar, solicitando métodos o funciones de dicho objeto, para trabajar con el elemento seleccionado.

> **Nota:** Como hemos dicho, $() es un resumen o forma corta de invocar la función jQuery. También podríamos haber escrito la línea de esta manera: var elem1 = jQuery("#elem1");

Luego, podríamos hacer cualquier cosa con ese elemento seleccionado, como lo siguiente:

```
elem1.css("background-color", "#ff9999");
```

Este método css() no forma parte del core, pero sirve para cambiar atributos CSS de un elemento, entre otras cosas. Así pues, con esa línea cambiaríamos el color de fondo del elemento seleccionado anteriormente, que habíamos guardado en la variable elem1.

Ahora veamos otro ejemplo de la selección de elementos con la función dólar.

```
var divs = $("div");
divs.css("font-size", "32pt");
```

Aquí seleccionamos todas las etiquetas DIV de la página, y les colocamos un tamaño de letra

de 32pt.

Podemos ver en una página aparte este pequeño script en uso.

El código de esta página es el siguiente:

```
<html>
<head>
  <title>función jquery</title>
  <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
  var elem1 = $("#elem1");
  //podríamos haber escrito: var elem1 = jQuery("#elem1");
  elem1.css("background-color", "#ff9999");

  var divs = $("div");
  //podríamos haber escrito: var elem1 = jQuery("#elem1");
  divs.css("font-size", "32pt");
});

</script>
</head>
<body>
<div id="elem1">Este elemento se llama elem1</div>
<div id="elem2">Este otro elemento se llama elem2</div>
</body>
</html>
```

Si queremos, podemos utilizar el segundo parámetro opcional, que es el contexto. Con él podríamos conseguir seleccionar elementos que pertenecen a una zona concreta de nuestro documento. Por defecto el contexto es la página entera, pero lo podemos restringir de esta manera:

```
var inputs = $("input",document.forms[0]);
inputs.css("color", "red");
```

Con la primera línea conseguimos seleccionar todos los elementos INPUT que pertenecen al primer formulario de la página. Se devolverá un objeto jQuery que contiene todos los input seleccionados. Con la segunda línea, invocando el método css() sobre el objeto jQuery recibido, estaríamos cambiando el color del texto de esos elementos. Esto no seleccionaría los INPUT de otros formularios, como se puede ver en esta página de ejemplo.

Ahora por completar un poco más estas notas, veamos otro ejemplo en el que seleccionamos todos los párrafos (etiqueta P), pero restringimos el contexto únicamente los que están en un DIV con id="div1".

```
var parrafos_div1 = $("p","#div1");
parrafos_div1.hide()
```

En la segunda línea lanzamos el método hide() sobre el objeto jQuery que contiene los párrafos seleccionados, con lo que conseguimos que se oculten. Podemos ver una página que con este ejemplo en marcha.

En el siguiente artículo veremos otros usos de la función jQuery() / función dólar $().

*Artículo por **Miguel Angel Alvarez***

Otros usos de la función $()

En el pasado capítulo del manual de jQuery vimos unas primeras notas sobre el Core de jQuery y comenzamos a explicar la función jQuery(), que es la más importante en este framework Javascript. Ahora veremos como esta función, que también se puede acceder por medio del símbolo dólar $(), puede tener otras aplicaciones útiles, cuando recibe parámetros distintos a los que vimos anteriormente.

Función jQuery pasando un HTML

Una posibilidad de trabajar con la función jQuery es enviarle un string con un HTML. Esto crea esos elementos en la página y les coloca los contenidos que se indique en el string. Ojo, que el HTML tiene que estar bien formado para que funcione en cualquier navegador, esto es, que se coloquen etiquetas que se puedan meter en el BODY de la página y que todas las etiquetas tengan su cierre.

```
var nuevosElementos = $("<div>Elementos que creo en <b>tiempo de ejecución</b>.<h1>En ejecución...</h1></div>");
```

Esto nos creará en la variable nuevosElementos los elementos HTML que hemos especificado en el string. Luego podríamos hacer lo que queramos con ellos, como colocarlos en la página con el método appendTo(), por ejemplo de esta manera:

```
nuevosElementos.appendTo("body");
```

> **Nota:** el método appendTo() no pertenece al Core, pero nos viene bien utilizarlo y así hacer algo con los elementos que acabamos de crear.

Veamos el código completo de una página que hace uso de este ejemplo:

```
<html>
<head>
  <title>función jquery</title>
  <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
  var nuevosElementos = $("<div>Estos elementos ..</b>.<h1>Título...</h1></div>");
  nuevosElementos.appendTo("body");
});

</script>
</head>
<body>
<p>Esto es el cuerpo de la página, que tiene poca cosa...</p>
</body>
</html>
```

Ahora, dejamos el link para ver el ejemplo en una página aparte.

Función jQuery pasando elementos

Otro posible valor que se le puede enviar a la función jQuery es un elemento o una jerarquía de elementos del DOM, para extenderlos con las funcionalidades que aporta el framework para los elementos.

Por ejemplo:

```
var documento = $(document.body);
documento.css("background-color", "#ff8833");
```

Con la primera línea creamos una variable llamada documento, a la que asignamos el valor que devuelve el método $() enviando el parámetro document.body.

> **Nota:** La variable document.body corresponde con un elemento del DOM de Javascript, que crea automáticamente el navegador y hace referencia al documento de la página.

Con ello obtenemos un objeto que es el cuerpo de la página (document.body) al que le hemos agregado todas las funcionalidades del framework jQuery para los elementos.

Así pues, en la línea siguiente, invocamos el método css() sobre la variable "documento", que es el propio documento de la página extendido. Por eso el método css(), que es de jQuery(), funciona sobre ese objeto.

Algo como esto no funcionaría porque estaríamos intentando lanzar un método de jQuery directamente sobre el objeto DOM sin haberlo extendido:

```
document.body.css("background-color", "#ff8833");
```

No funcionará porque no podemos llamar a un método jQuery sobre un objeto DOM, si es que no lo hemos extendido con la función $().

> **Nota:** Esta función acepta documentos XML y objetos Window, aunque no son propiamente elementos del DOM.

Podemos ver ahora una página donde se encuentra este ejemplo en marcha.

Función jQuery pasando una función

En la función $() una última posibilidad es pasarle como parámetro una función y entonces lo que tenemos es una función callback que se invoca automáticamente cuando el DOM está listo.

> **Nota:** Ya explicamos lo que era un callback en el artículo Callback de funciones jQuery

En esa función podemos colocar cualquier código, que se ejecutará sólo cuando el DOM está listo para recibir comandos que lo modifiquen. Con ello, esta función nos vale perfectamente para hacer cualquier cosa dentro de la página que afecte al DOM.

Ejemplo:

```
$(function (){
  //Aquí puedo hacer cualquier cosa con el DOM
});
```

Este callback con la función jQuery $() sería una abreviatura de otro método que hemos visto repetidas veces a lo largo de este manual para definir acciones cuando el DOM está listo:

```
$(document).ready(function(){
  //Aquí puedo hacer cualquier cosa con el DOM
});
```

Incluso podemos hacer varios callback, para agregar distintas acciones a realizar cuando el DOM está listo, las veces que queramos, igual que cuando definíamos el evento ready() sobre el objeto document.

Podemos ver el código de una página que hace uso de la función dólar, pasando por parámetro una función.

```
<html>
<head>
  <title>función jquery</title>
  <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>

$(function (){
  var documento = $("p");
  documento.css("background-color", "#ff8833");
});

$(function (){
  var documento = $("b");
  documento.css("color", "#fff");
});

//si colocase aquí este código, no funcionaría, porque el DOM no estaría listo para realizar acciones
//var documento = $("p");
//documento.css("background-color", "#ff8833");

</script>
</head>
<body>
<p><b>Párrafo</b>!!</p>
<p>Otro <b>párrafo</b></p>
</body>
</html>
```

Se puede ver en marcha en una página aparte.

Hasta aquí hemos visto todas las posibilidades que existen para trabajar con la función jQuery. Realmente a lo largo de este manual ya la habíamos utilizado muchas veces y en lo sucesivocontinuaremos usándola, ya que cualquier cosa que deseemos hacer en una página web va a depender en algún momento de invocar a $() en alguna de sus variantes.

*Artículo por **Miguel Angel Alvarez***

Videtutorial función $ de jQuery

Estamos realizando una serie de vídeos para explicar el funcionamiento y la programación con el framework Javascript jQuery. En el anterior vídeo publicado en DesarrolloWeb.com mostramos cómo instalar jQuery y cómo hacer una primera página que utiliza el framework para hacer un pequeño efecto. Era un vídeo introductorio, que deberíamos ver antes que éste, donde se explican las cosas desde cero.

Así que en estos dos vídeos que podréis ver a continuación partimos desde la base que ya habéis puesto jQuery a funcionar en una primera página. Ahora veremos detalles sobre la programación de este framework, simples y fundamentales también para dar los primeros pasos.

Se trata de conocer el Core de jQuery, que es el núcleo donde se encuentran las funciones principales y más recurridas de estas librerías. En concreto veremos la denominada función jQuery, que también se conoce como función dólar $(). Esta función tiene varios usos, que vamos a ver a lo largo de estos dos vídeos.

Los presentes vídeos sirven para ilustrar un par de artículos publicados en el Manual de jQuery, para que las personas con menos experiencia puedan ver por la práctica cómo ponerse manos a la obra en la utilización de este framework javascript.

En realidad en el videotutorial es el mismo, pero está partido en dos partes, de 10 minutos cada una, para poder subirlo a YouTube.

Vídeo 1: función jQuery o función $()

En la primera parte de este videotutorial hemos dedicado el tiempo a explicar el uso principal de la función jQuery, que es seleccionar elementos de la página y extenderlos para que atiendan a las funcionalidades que aporta este framework. Los elementos se seleccionan indicando como parámetro a la función dólar un selector, que sirve para definir el elemento o elementos que queremos seleccionar. Existen varios tipos de selectores que veremos a lo largo del vídeo, al menos los más importantes.

El texto que complementa este vídeo está en el artículo Función jQuery o función $().

El ejemplo construido en este vídeo de jQuery se puede ver en marcha en una página aparte.

Vídeo 2: Otros usos de la función $()

En la segunda parte del vídeo mostraremos otros usos de la función dólar $(), con diversas aplicaciones según los parámetros que le pasemos. Por ejemplo podemos pasar un HTML a la función, para generar todos esos elementos que luego podríamos insertar en la página. También se puede enviar un objeto o incluso una función, como veremos en el videotutorial, junto con algunos ejemplos que ilustrarán las posibilidades de todos esos usos.

Este vídeo sirve de guía para explicar el artículo Otros usos de la función $().

Si lo deseamos, podemos ver el ejemplo realizado en este segundo vídeo del framework jQuery.

*Artículo por **Miguel Angel Alvarez***

Core/each: each del core de jQuery

Hay algo que tenemos que tener en cuenta, en la medida de lo posible, cuando creamos código Javascript: crear un código de calidad y lo más corto posible. Para ello también nos facilitan mucho las cosas los frameworks y métodos como each(), que veremos en este artículo.

El método each() pertenece al juego de funciones del CORE de jQuery, cuyas particularidades ya comenzamos a analizar en el manual de jQuery. Se trata de un método para realizar acciones con todos los elementos que concuerdan con una selección realizada con la función jQuery -también llamada función $()-. Útil porque nos da una manera cómoda de iterar con elementos de la página y hacer cosas con ellos más o menos complejas de una manera rápida y sin utilizar mucho código para definir el bucle.

Cómo funciona each

Each es un método que se utiliza sobre un conjunto de elementos que hayamos seleccionado con la función jQuery. Con each realizamos una iteración por todos los elementos del DOM que

se hayan seleccionado.

El método each recibe una función que es la que se tiene que ejecutar para cada elemento y dentro de esa función con la variable "this" tenemos una referencia a ese elemento del DOM. Adicionalmente, la función que se envía a each, puede recibir un parámetro que es el índice actual sobre el que se está iterando.

Quiero explicar las bondades de each() con un ejemplo. Por ejemplo, veamos esta línea de código:

```
$("p").css("background-color", "#eee");
```

Como ya sabemos, con $("p") seleccionamos todos los párrafos de la página. Luego con el método CSS asignamos un estilo, en este caso para cambiar el color del fondo. Esto en realidad jQuery lo hace con una iteración con todos los párrafos de la página, sin que tengamos que hacer nosotros nada más y es genial que se permita en el uso del framework. ¿Pero qué pasa si queremos cambiar el fondo de los párrafos utilizando colores alternos?

En este caso no podemos hacerlo en una sola línea de código, pero each nos vendrá como anillo al dedo.

Imaginemos que tenemos una serie de párrafos en la página y queremos cambiar el color de fondo a los mismos, de manera que tengan colores alternos, para hacer dinámicamente un típico diseño para los listados.

Entonces podríamos hacer lo siguiente:

```
$("p").each(function(i){
  if(i%2==0){
    $(this).css("background-color", "#eee");
  }else{
    $(this).css("background-color", "#ccc");
  }
});
```

Con $("p") tengo todos los párrafos. Ahora con each puedo recorrerlos uno a uno. Para cada uno ejecutaremos la función que enviamos como parámetro a each(). En esa función recibo como parámetro una variable "i" que contiene el índice actual sobre el que estoy iterando.

Con if(i%2==0) estoy viendo si el entero del índice "i" es par o impar. Siendo par coloco un color de fondo al elemento y siendo impar coloco otro color de fondo.

Como se puede ver, con la variable "this" tenemos acceso al elemento actual. Pero OJO, que este elemento no es un objeto jQuery, así que no podríamos enviarle métodos del framework jQuery hasta que no lo expandamos con la función jQuery. Así pues, tenemos que hacer $(this) para poder invocar al método css(). Por si no queda claro este punto mirar las diferencias entre estas dos líneas de código:

```
this.css("background-color", "#ccc");
$(this).css("background-color", "#ccc");
```

En la primera línea no estaríamos extendiendo la variable this con las funcionalidades de jQuery, luego no funcionaría.

En la segunda línea, que es la que habíamos utilizado en el script de ejemplo, sí estamos extendiendo la variable "this" con la función jQuery. De ese modo, se puede invocar a cualquier método de jQuery sobre los elementos.

Podemos ver un ejemplo en marcha que hace uso de ese script.

Este sería su código fuente completo:

```
<html>
```

```
<head>
  <title>each del core de jQuery</title>
  <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
  $("p").each(function(i){
    if(i%2==0){
      $(this).css("background-color", "#eee");
    }else{
      $(this).css("background-color", "#ccc");
    }
  });
});
</script>
</head>
<body>
<p>Primer párrafo</p>
<p>Otro párrafo</p>
<p>Tecer párrafo</p>
<p>Uno más</p>
<p>y acabo...</p>
</body>
</html>
```

Retornando valores en la función que enviamos a each

Ahora vamos a ver un par de posibilidades interesantes al utilizar each. Resulta que la función que enviamos como parámetro a each() puede devolver valores y dependiendo de éstos, conseguir comportamientos parecidos a los conocidos break o continue de los bucles Javascript.

Si la función devuelve "false", se consigue detener por completo el proceso de iteraciones de each(). Esto es como si hiciéramos el típico "break".

Si la función devuelve "true", se consigue pasar directamente a la próxima iteración del bucle. Es como hacer el típico "continue".

Para ver estos dos casos realizaremos otro ejemplo de uso de each.

Tenemos varios DIV, donde cada uno tiene un texto.

```
<div>red</div>
<div>blue</div>
<div>red</div>
<div>white</div>
<div>red</div>
<div>green</div>
<div>orange</div>
<div>red</div>
<div>nada</div>
<div>red</div>
<div>blue</div>
```

Ahora queremos hacer un recorrido a esos DIV y en cada uno, mirar el texto que aparece. Entonces colocaremos como color del texto del DIV el color que aprece escrito en el DIV. Pero con dos casos especiales:

- Si el texto del DIV es "white", entonces no queremos hacer nada con ese elemento.
- Si el texto del DIV es "nada", entonces detendremos el bucle y dejaremos de colorear los siguientes elementos.

Esto lo podríamos hacer con el siguiente código:

```
$("div").each(function(i){
  elemento = $(this);
  if(elemento.html() == "white")
    return true;
  if(elemento.html() == "nada")
    return false;
  elemento.css("color", elemento.html());
});
```

Ahora podremos ver este ejemplo en funcionamiento en una página aparte.

*Artículo por **Miguel Angel Alvarez***

Método size() y propiedad length del core de jQuery

Vamos a ver en este artículo cómo obtener el número de elementos que tiene el objeto jQuery. Como pudimos aprender previamente en nuestro manual de jQuery, este framework Javascript tiene como base la llamada "función jQuery" que devuelve el objeto jQuery, en el que hay uno o varios elementos de la página, según el selector que se le haya pasado.

En un pasado artículo de DesarrolloWeb.com ya explicamos con detalle qué era la función jQuery y cómo utilizarla. Pues bien, ahora veremos rápidamente cómo saber cuántos elementos encontramos y seleccionamos por medio de esta función, lo que puede ser útil por diversos motivos al hacer código Javascript.

Método size() del Core de jQuery

Este método sirve, como decimos, para obtener el número de elementos seleccionados con la función jQuery. Simplemente devuelve el número de elementos que hay en el objeto, como un entero.

Por ejemplo, veamos este código:

```
var parrafos = $("p");
alert ("Hay " + parrafos.size() + " párrafos en la página");
```

Con la primera línea selecciono todos los párrafos de la página y los meto en el objeto jQuery de la variable "parrafos". Mediante la segunda línea muestro el número de párrafos encontrados, con una llamada al método size().

No tiene más misterio, salvo que en jQuery existe otro modo de hacer esto, pero bastante más rápido.

Podemos ver una página en marcha con este ejemplo de uso de size().

Propiedad length del objeto jQuery

La propiedad length, que existe en cualquier objeto jQuery, nos sirve para obtener el número de elementos de la página que hemos seleccionado. Lo interesante de esta propiedad es que almacena directamente este valor, por lo que es más rápido y más aconsejable que calcular los elementos seleccionados con el método size().

Tanto el método size() con la propiedad length devolverán el mismo valor, siendo las únicas diferencias la mencionada rapidez adicional de la propiedad y el acceso a este valor, que como es una propiedad, se accede a través del operador punto, pero sin colocar los paréntesis después de length. Por ejemplo, veamos este código:

```
var ElementosMitexto = $(".mitexto");
alert ("Hay " + ElementosMitexto.length + " elementos de la clase mitexto");
```

Con la primera línea estamos utilizando la función jQuery para seleccionar todos los elementos de la página que tienen la clase CSS "mitexto". Con la segunda línea se muestra en una caja de alerta el número de elementos seleccionados con ElementosMitexto.length.

Podemos ver el código completo de una página que hace uso de este método:

```
<html>
<head>
  <title>propiedad length del core jQuery</title>
  <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
  //selecciono todos los elementos de la clase "mitexto"
  var ElementosMitexto = $(".mitexto");
  //muestro el número de los párrafos encontrados
  alert ("Hay " + ElementosMitexto.length + " elementos de la clase mitexto");
});
</script>
</head>

<body>

<p>Esto es un párrafo normal</p>
<p class="mitexto">Esto es un párrado de la clase "mitexto"</p>
<div>Un div normal</div>
<div class="mitexto">Ahora un div de la clase "mitexto"</div>
</body>
</html>
```

Para acabar, dejamos el enlace a una página donde se puede ver el ejemplo de la propiedad length del objeto jQuery en funcionamiento.

*Artículo por **Miguel Angel Alvarez***

Método data() Core jQuery

Seguimos viendo componentes interesantes del "Core" de jQuery, donde están las clases y métodos más básicos de este framework Javascript. En esta entrega del manual de jQuery de DesarrolloWeb.com, explicaremos el uso del método data() y removeData(), que sirven para almacenar, consultar y eliminar cualquier tipo de dato en elementos de la página.

En algunas ocasiones resulta útil almacenar variables u objetos en determinados elementos de la página. Aunque quizás no es una acción muy corriente en los primeros pasos con jQuery, en el futuro encontraréis que resulta útil y veréis herramientas y plugins que utilizan este mecanismo para su operativa. De modo que conviene al menos saber que esto es posible y conocer de qué manera podemos utilizar los elementos de la página para guardar cosas en ellos.

Para ello vamos a comentar dos métodos distintos que forman parte del core de jQuery:

Método data()

Este método del objeto jQuery sirve tanto para guardar un dato en un elemento como para consultarlo. Según el número de parámetros que reciba, realiza una u otra acción.

- Si recibe un parámetro data(nombre): devuelve el valor que haya en el dato cuyo nombre se pasa por parámetro.
- Si recibe dos parámetros data(nombre, valor): almacena un dato, cuyo nombre recibe en el primer parámetro, con el valor que recibe en el segundo parámetro.

Como data() es un método que pertenece al objeto jQuery, podemos almacenar estos pares (dato, valor) en cualquiera de los elementos que seleccionemos con la función jQuery().

Veamos un caso de uso simple. Por ejemplo tenemos un elemento de la página como este:

```
<div id="capa">
En esta división (elemento id="capa") voy a guardar y leer datos sobre este elemento.
</div>
```

Ahora podríamos usar le método data() de la siguiente manera:

```
$("#capa").data("midato","mivalor");
```

Con esta línea hemos guardado un dato llamado "midato" con el valor "mivalor", en el elemento con identificador (atributo id) "capa".

Ahora podríamos leer ese dato en cualquier momento para acceder a su valor, de la siguiente manera:

```
alert($("#capa").data("midato"));
```

En esta línea de código extraemos el dato "midato" del elemento con identificador "capa" y lo mostramos en una caja de alerta.

Podemos ver una página en marcha que hace uso de esas dos funciones.

Método removeData()

Este método sirve para eliminar un dato de un elemento y su funcionamiento es tan simple como enviar por parámetro el dato que se quiere eliminar del elemento.

```
$("#capa").removeData("midato")
```

Con esta línea habríamos eliminado el dato llamado "midato" del elemento con identificador "capa".

Ejemplo completo de los métodos data() y removeData() del Core de jQuery

Veamos un ejemplo completo del uso de estos métodos que acabamos de aprender. Se trata de una página que tiene un elemento sobre el que vamos a guardar datos. Además tiene tres botones para guardar un dato, leerlo y borrarlo. El dato que se guardará tendrá como valor lo que se haya escrito en un campo de texto que aparece también en la página.

Podemos ver el ejemplo en marcha en una página aparte.

Tenemos, para comenzar, un elemento de la página, que es donde vamos a guardar los pares dato-valor con data().

```
<div id="division">
En esta división (elemento id="division") voy a guardar datos con la función data y luego los voy a leer.
</div>
```

Luego tendremos este formulario, que contiene el campo de texto así como los tres botones de los que hemos hablado.

```
<form name="formul">
Escribe un valor a guardar, leer o eliminar:
<input type="text" name="valor" id="valor">
<br>
<input type="button" value="guardar dato" id="guardar">
<input type="button" value="leer dato" id="leer">
<input type="button" value="eliminar dato" id="eliminar">
</form>
```

Ahora se trata de asignar los comportamientos a estos botones con Javascript, haciendo uso de jQuery.

Este sería el script para agregar el evento click al botón de guardar datos.

```
$("#guardar").click(function(evento){
  var valor = document.formul.valor.value;
  //Esta misma línea de código se puede codificar así también con jQuery
  //var valor = $("#valor").attr("value");
  $("#division").data("midato",valor);
  $("#division").html('He guardado en este elemento (id="division") un dato llamado "midato" con el valor "' + valor
+ '"');
});
```

Como se puede ver, primero se recibe el texto del campo de texto que había en el formulario. Para ello se muestran dos maneras de hacerlo:

- A través de la jerarquía de objetos del navegador, con document.formul.valor.value
- A través de su identificador, con un método de jQuery llamado attr() que sirve para recuperar el valor de un atributo HTML pasado por parámetro sobre el elemento que recibe el método. Este modo de obtener el atributo value del campo de texto está comentado, pues sólo lo quería señalar, para que se vea el modo de acceder a un elemento de formulario utilizando las funciones del framework Javascript jQuery.

Luego, se guarda el dato "midato" con el valor que se recuperó del atributo value del campo de texto. Para ello utilizamos el método data() tal como comentábamos.

Por último se muestra un mensaje en el HTML del elemento con id="division", por medio del método html() de jQuery, para informar sobre la acción que acabamos de realizar.

Ahora mostramos el código para asignar un comportamiento al evento click sobre el segundo botón:

```
$("#leer").click(function(evento){
  valor = $("#division").data("midato");
  $("#division").html('En este elemento (id="division") leo un dato llamado "midato" con el valor "' + valor + '"');
});
```

Como se puede ver, se recupera el valor del dato "midato" guardado sobre el elemento "#division" (etiqueta HTML con id="division"), y se almacena en una variable. Luego se crea un mensaje para mostrar el valor del dato.

Para acabar, tenemos el código del evento click sobre el botón de eliminar el contenido de un dato, que hace uso de removeData().

```
$("#eliminar").click(function(evento){
  $("#division").removeData("midato");
  $("#division").html('Acabo de eliminar del elemento (id="division") el dato llamado "midato"');
});
```

Como se verá, el método removeData() se invoca sobre el elemento que tiene el dato que pretendemos eliminar. Más tarde se muestra un mensaje informando sobre la acción que se ha realizado.

Para comprobar el funcionamiento de estos métodos habría que crear un dato, escribiendo el valor en el campo de texto y pulsando el botón "guardar dato". Luego podríamos leer ese dato con el botón "leer dato". Por último podríamos eliminar el dato con el botón "eliminar dato". Si, una vez eliminado pulsamos sobre el botón de "leer dato" veremos que el valor del dato aparece como "undefined", puesto que ese dato ha sido borrado (esto también ocurre si no se ha guardado ningún dato todavía, por ejemplo cuando se acaba de cargar la página).

Sería interesante ver el código fuente completo de esta página, para hacernos una idea más exacta de cómo se integrarían todos estos elementos.

```
<html>
<head>
  <title>Ejemplos de uso de la función data del core de jQuery</title>
  <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){

  $("#guardar").click(function(evento){
    var valor = document.formul.valor.value;
    //Esta misma línea de código se puede codificar así también con jQuery
    //var valor = $("#valor").attr("value");
    $("#division").data("midato",valor);
    $("#division").html('He guardado en este elemento (id="division") un dato llamado "midato" con el valor "' + valor
+ '"');
  });

  $("#leer").click(function(evento){
    valor = $("#division").data("midato");
    $("#division").html('En este elemento (id="division") leo un dato llamado "midato" con el valor "' + valor + '"');
  });

  $("#eliminar").click(function(evento){
    $("#division").removeData("midato");
    $("#division").html('Acabo de eliminar del elemento (id="division") el dato llamado "midato"');
  });
});
</script>
</head>

<body>

<div id="division">
En esta división (elemento id="division") voy a guardar datos con la función data y luego los voy a leer
</div>
<br>
<form name="formul">
Escribe un valor a guardar, leer o eliminar:
<input type="text" name="valor" id="valor">
<br>
<input type="button" value="guardar dato" id="guardar">
<input type="button" value="leer dato" id="leer">
<input type="button" value="eliminar dato" id="eliminar">
</form>

</body>
</html>
```

De nuevo, dejamos el enlace al ejemplo en marcha.

Para seguir os indicamos la lectura del siguiente artículo de este manual, donde puedes obtener explicaciones adicionales y ejemplos de uso de estos métodos data() y removeData().

*Artículo por **Miguel Angel Alvarez***

Consideraciones interesantes de data() y removeData()

Existen algunos puntos que debemos conocer sobre el funcionamiento de estos métodos que no hemos explicado todavía en el artículo anterior, en el que se comenzó a tratar acerca de data() y removeData(). Veamos a continuación una serie de consideraciones:

Admite cualquier tipo de dato: Podemos guardar lo que deseemos por medio del método data(). Los ejemplos anteriores hemos guardado simplemente cadenas de texto, pero soportaría cualquier tipo de variable, numérica, un array o incluso un objeto Javascript o jQuery.

Se guarda un dato por cada elemento del objeto jQuery selecionado: En caso que en el objeto jQuery sobre el que estemos almacenando cosas con data() haya referencias a varios elementos de la página, el dato se almacena en todos los elementos. (recordemos que, según lo explicado anteriormente en desarrolloweb.com, un objeto jQuery puede tener seleccionados varios elementos de la página, como todos los enlaces presentes, los elementos de una determinada clase CSS, etc. dependiendo del selector escogido al hacer uso de la función dólar)

Los objetos se almacenan por referencia: En el caso que estemos almacenando un objeto Javascript con data() sobre uno o varios elementos, no se copia el objeto, sino que se asigna por referencia. Esto quiere decir que no se harían copias independientes del objeto a guardar, sino que permanecería tal cual y lo que se asignaría como dato es una referencia a ese único objeto.

Ahora, para investigar un poco sobre estas posibilidades, hemos creado un par de ejemplos un poco más complejos que hacen uso de los métodos data() y removeData(). Son ejemplos más avanzados, que hacen uso de algunas cosas que no hemos explicado todavía en este manual de jQuery. No obstante, vendrá bien verlos para aprender algunos usos de estas funcionalidades.

Para empezar, quiero mostrar una página de ejemplo donde existen tres enlaces y dos botones. Al pulsar cualquiera de los enlaces mostraremos el contenido de un dato almacenado en ellos con data(). Los botones, por su parte, servirán para almacenar contenidos en datos sobre esos enlaces. Además tendremos una capa con id="mensaje" que nos servirá para mostrar cosas por pantalla.

Podemos ver el ejemplo en marcha en una página aparte.

El código de los elementos HTML será el siguiente:

```
<a href="#" id="enlace1">Enlace 1</a>
<br>
<a href="#" id="enlace2">Enlace 2</a>
<br>
<a href="#" id="enlace3">Enlace 3</a>
<br>
<br>
<div id="mensaje">
Mensaje...
</div>
<br>
<button id="guardar">guardar "midato" con valor "mivalor" en todos los enlaces</button>
```

```
<br>
<button id="guardarenlace1">guardar "midato" con valor "otro valor" en el enlace 1</button>
```

Ahora veamos cómo aplicar eventos a los elementos de la página, para almacenar datos y mostrarlos.

Comencemos por el código de los eventos de los botones.

```
$("#guardar").click(function(evento){
   $("a").data("midato","mivalor");
   $("#mensaje").html('He guardado en todos los enlaces un dato llamado "midato" con el valor "mivalor"');
});
```

Con este código estamos almacenando datos en todos los enlaces. Cabe fijarse que con la función jQuery $("a") obtenemos un objeto jQuery donde están todos los enlaces de la página. Luego, al invocar data() sobre ese objeto, estamos almacenado ese dato en todos los enlaces existentes.

```
$("#guardarenlace1").click(function(evento){
   $("#enlace1").data("midato","otro valor");
   $("#mensaje").html('He guardado en el enlace1 un dato llamado "midato" con el valor "otro valor"');
});
```

En este otro código del evento click para el segundo botón, almacenamos "otro valor" sobre el dato de antes, pero sólo lo hacemos sobre el enlace 1, dado que hemos utilizado el selector $ ("#enlace1"), con el identificador único del primer enlace.

Y ahora podríamos ver el código para asignar un evento a todos los enlaces, para que al pulsarlos nos muestre lo que haya en el dato almacenado con data(), si es que hay algo.

```
$("a").click(function(evento){
   evento.preventDefault();
   valorAlmacenado = $(this).data("midato");
   $("#mensaje").html("En el enlace <b>" + $(this).attr("id") + "</b> tiene el dato 'midato' como " +
valorAlmacenado);
});
```

Como se puede ver, estamos creando un evento click, pero lo estamos haciendo sobre los tres enlaces que hay en la página a la vez, dado el selector utilizado en la función jQuery $("a"). Luego el código del evento será el mismo para los tres enlaces.

Lo primero que se hace es un evento.preventDefault() que permite que el enlace no tenga el comportamiento típico (ir a la URL del href). A continuación hacemos:

```
valorAlmacenado = $(this).data("midato");
```

Como se puede ver, se está extrayendo el valor almacenado en el enlace actual, que recibe el evento. Con $(this) obtenemos el objeto jQuery del elemento que ha recibido el evento, que es el enlace sobre el que se ha pulsado y no todos los enlaces. Con el método data("midato"), invocado sobre $(this), obtenemos el valor del dato "midato" almacenado en el enlace que fue pulsado solamente.

Luego se muestra un mensaje para indicar el valor que había en el dato. Pero claro, este código, como es común para todos los enlaces, tiene que acceder también a $(this) para saber qué enlace en concreto fue el que se pulsó. Para identificar el enlace se hace $(this).attr("id"), que devuelve el atributo "id" del enlace sobre el que se hizo clic.

A continuación se puede ver el código completo de esta página.

```
<html>
<head>
   <title>Ejemplos de uso de la función data del core de jQuery</title>
   <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
```

```
$(document).ready(function(){

  $("a").click(function(evento){
    evento.preventDefault();
    valorAlmacenado = $(this).data("midato");
    $("#mensaje").html("En el enlace <b>" + $(this).attr("id") + "</b> tiene el dato 'midato' como " +
valorAlmacenado);
  });

  $("#guardar").click(function(evento){
    $("a").data("midato","mivalor");
    $("#mensaje").html('He guardado en todos los enlaces un dato llamado "midato" con el valor "mivalor"');
  });

  $("#guardarenlace1").click(function(evento){
    $("#enlace1").data("midato","otro valor");
    $("#mensaje").html('He guardado en el enlace1 un dato llamado "midato" con el valor "otro valor"');
  });
});
</script>
</head>

<body>

<a href="#" id="enlace1">Enlace 1</a>
<br>
<a href="#" id="enlace2">Enlace 2</a>
<br>
<a href="#" id="enlace3">Enlace 3</a>
<br>
<br>
<div id="mensaje">
Mensaje...
</div>
<br>
<button id="guardar">guardar "midato" con valor "mivalor" en todos los enlaces</button>
<br>
<button id="guardarenlace1">guardar "midato" con valor "otro valor" en el enlace 1</button>

</body>
</html>
```

Si se desea, se puede ver el ejemplo en marcha en una página aparte. ht

Datos de tipo objeto asignados por referencia con data()

Sobre el punto que comentábamos antes, sobre los objetos Javascript que se asignan por medio de data(), que siempre se hace por referencia, hemos creado otro ejemplo, del que simplemente vamos a colocar un enlace para verlo en funcionamiento y su código. http://www.desarrolloweb.com/articulos/ejemplos/jquery/core/data3.html

El ejemplo es bastante similar al anterior, con la salvedad que se ha creado un par de acciones adicionales para almacenar en los elementos variables de tipo objeto.

Luego, al operar sobre esos datos de tipo objeto, comprobamos que en realidad sólo existe un objeto compartido por todos los elementos a los que fue asignado. Es decir, no se hicieron copias del objeto, sino que se asignaron en los datos simplemente su referencia.

Puede verse este ejemplo en marcha en una página aparte.

El código completo se puede ver a continuación.

```
<html>
```

```
<head>
  <title>Ejemplos de uso de la función data del core de jQuery</title>
  <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){

  $("a.enlacealmacenar").click(function(evento){
    evento.preventDefault();
    var valorAlmacenado = $(this).data("midato");
    var mensaje = "En el enlace <b>" + $(this).attr("id") + "</b> tiene el dato 'midato' como " + valorAlmacenado;
    var valorAlmacenado2 = $(this).data("miobjeto");
    mensaje += "<br>Además, he leído un dato llamado 'miobjeto' con valor " + valorAlmacenado2;
    $("#mensaje").html(mensaje);
  });

  $("#guardar").click(function(evento){
    evento.preventDefault();
    $("a").data("midato","mivalor");
    $("#mensaje").html('He guardado en todos los enlaces un dato llamado "midato" con el valor "mivalor"');
  });

  $("#guardarenlace1").click(function(evento){
    evento.preventDefault();
    $("#enlace1").data("midato","otro valor");
    $("#mensaje").html('He guardado en el enlace1 un dato llamado "midato" con el valor "otro valor"');
  });

  $("#guardarobjeto").click(function(evento){
    evento.preventDefault();
    $("a").data("miobjeto",$("#capapruebas"));
    $("#mensaje").html('He guardado todos los enlaces un dato llamado "miobjeto" con el valor un objeto que es el
objeto jquery de seleccionar la capa con id "capapruebas"');
  });

  $("#operarobjetoenlace1").click(function(evento){
    evento.preventDefault();
    $("#enlace1").data("miobjeto").html("cambio el html del objeto que hay en el dato 'miobjeto' del 'enlace1'");
  });

  $("#operarobjetoenlace2").click(function(evento){
    evento.preventDefault();
    $("#mensaje").html("Este es el HTML que hay en el objeto asociado a enlace2 en el dato 'miobjeto':<br>" + $
("#enlace2").data("miobjeto").html());
  });

});
</script>
</head>

<body>

<a href="#" id="enlace1" class="enlacealmacenar">Enlace 1</a>
<br>
<a href="#" id="enlace2" class="enlacealmacenar">Enlace 2</a>
<br>
<a href="#" id="enlace3" class="enlacealmacenar">Enlace 3</a>
<br>
<br>
<div id="mensaje">
Mensaje...
</div>
<br>

<ol style="line-height: 200%;">
<li>
```

```
<a id="guardar" href="#">guardar "midato" con valor "mivalor" en todos los enlaces</a>
</li>
<li>
<a id="guardarenlace1" href="#">guardar "midato" con valor "otro valor" en el enlace 1</a>
</li>
<li>
<a id="guardarobjeto" href="#">guardar "miobjeto" con una referencia a la capa de pruebas</a>
</li>
<li style="line-height: 100%;">
<a id="operarobjetoenlace1" href="#">Recuperar un objeto del enlace1 para hacer cosas con él
<SPAN style="font-size: 8pt; font-weight: bold">
PULSAR ESTE ENLACE SÓLO DESPUÉS DE HABER ALMACENADO EL OBJETO EN LOS ENLACES POR MEDIO DEL ENLACE DE ESTA LISTA MARCADO COMO 3)
</SPAN>
</a></li>
<li style="line-height: 100%;">
<a id="operarobjetoenlace2" href="#">Recuperar un objeto del enlace2 para hacer cosas con él
<SPAN style="font-size: 8pt; font-weight: bold">
PULSAR ESTE ENLACE SÓLO DESPUÉS DE HABER ALMACENADO EL OBJETO EN LOS ENLACES POR MEDIO DEL ENLACE DE ESTA LISTA MARCADO COMO 3)
</SPAN>
</a></li>
</ol>
<br>
<br>

<div id="capapruebas">
Este es el texto de una capa de pruebas... con id="capapruebas"
</div>

</body>
</html>
```

Hemos visto diversos ejemplos de uso de data() y removeData(), métodos básicos de jQuery. Puede que ahora no se les encuentre mucha utilidad, pero nos servirán para resolver problemas futuros y entender cómo funcionan diversos plugins o componentes más avanzados de jQuery.

Por lo que respecta al Core de jQuery, ya hemos visto diversas funcionalidades en desarrolloweb.com en artículos de este manual. Por ahora lo vamos a dejar por aquí, aunque hay diversos métodos del Core que no hemos llegado a ver. En los próximos artículos pasaremos página y comenzaremos a ver otros temas interesantes que nos permitirán explotar un poco más nuestra creatividad, poniendo en marcha utilidades más cercanas a lo que pueden ser nuestras necesidades del día a día.

*Artículo por **Miguel Angel Alvarez***

Selectores en jQuery

Como la propia palabra indica, los selectores son un mecanismo, disponible en jQuery, para seleccionar determinados elementos de la página. El selector no es más que una cadena de caracteres, creada bajo unas normas que veremos a continuación, con la que podemos referirnos a cualquiera o cualesquiera de los elementos que hay en una página.

Todo en jQuery pasa por utilizar los selectores, para acceder a los elementos de la página que deseamos alterar dinámicamente con Javascript. Hasta en los ejemplos más básicos del Manual de jQuery se tienen que utilizar selectores para acceder a los elementos que deseamos

alterar, así que inevitablemente, si has leído este manual hasta este artículo, los habrás utilizado ya.

En mi opinión, una de las cosas que más potentes de jQuery son los selectores, al menos comparando este framework Javascript con otros que conozco. Veremos en este artículo cómo utilizarlos y aprovecharnos de su potencia.

Para empezar, veamos un selector, para aclarar las ideas y refrescar la memoria. Cuando utilizamos la función jQuery (o función dólar) lo que pasamos como parámetro es el selector. La función jQuery devuelve justamente los elementos de la página que concuerdan con el selector enviado por parámetro.

```
$("p");
```

En esa llamada a la función jQuery, estamos pasando por parámetro una cadena "p" y como decía, esa misma cadena es el selector. En este caso, "p" es un selector que sirve para seleccionar todas las etiquetas P de la página, es decir, los párrafos.

Selectores básicos en jQuery

Los selectores, al menos los más básicos, son parecidos, o iguales, a los que se utilizan en CSS para seleccionar los elementos a los que se desean aplicar ciertos estilos. Como entiendo que todas las personas que intenten profundizar en el framework jQuery deben haber conocido CSS anteriormente, no habrá ningún problema con ellos.

Selector de etiquetas:
Simplemente indicamos la etiqueta a la que deseamos referirnos, es decir, la etiqueta que queremos seleccionar. Obtendremos con él todas las etiquetas de la página indicada en el selector.

```
$("h1") //selecciona todos los encabezados de nivel 1
```

Selector por identificador:
Sirven para seleccionar los elementos que tengan un identificador dado, que se asigna a las etiquetas a través del atributo id del HTML. Para utilizar este selector se indica primero el carácter "#" y luego el identificador de cuyo elemento se desee seleccionar.

```
$("#idelemento") //selecciona una etiqueta que tiene el atributo id="idelemento"
```

Selector por clase:
Podemos indicar el nombre de una clase (class de CSS) y seleccionar todos los elementos a los que se ha aplicado esta clase. Para ello, como en CSS, comenzamos colocando el carácter "." y luego el nombre de la clase que deseamos seleccionar.

```
$(".miclase") //selecciona todos los elementos que tienen el atributo class="miclase"
```

Selector por varias clases:
Si lo deseamos, podemos indicar varias clases CSS, para obtener todos los elementos que tienen esas clases aplicadas: todas al mismo tiempo. Esto se consigue comenzando por un ".", igual que los selectores de clases, y luego otro "." para separar las distintas clases que queremos utilizar en el selector.

```
$(".clase1.clase2") //selecciona los elementos que tienen class="clase1 clase2"
```

Selector asterisco "*":
Nos sirve para seleccionar todos los elementos de la página.

```
$("*") //selecciona todos los elementos que tiene la página
```

Concatenar varios selectores distintos:
Por último, podemos utilizar varios selectores, para obtener todas las etiquetas que cumplen uno de ellos. No hace falta que cumplan todos los selectores a la vez, sino con que uno de ellos concuerde es suficiente. Para ello colocamos todos los selectores que deseamos, separados por una coma ",".

```
$("div,p") //selecciona todos los elementos división y párrafo
$(".clase1,.clase2") //selecciona los elementos que tienen la clase "clase1" o "clase2"
$("#miid,.miclase,ul) //selecciona el elemento con id="miid", los elementos con class="miclase" y todas las listas UL
```

Conclusión sobre los selectores

Hasta este punto hemos visto los selectores básicos de jQuery, que nos servirán para hacer la mayoría de nuestros ejemplos y resolver también la mayor parte de las necesidades de selección de elementos que nos podamos encontrar en ejemplos reales. Sin embargo, el framework Javascript incluye una buena gama de selectores adicionales que pueden venirnos bien en algunos casos más concretos y que dejamos para próximos artículos.

Ahora, os recomendamos seguir el aprendizaje con el siguiente artículo, en el que pondremos en práctica los selectores que hemos conocido hasta el momento: Ejemplo para practicar con selectores de jQuery.

*Artículo por **Miguel Angel Alvarez***

Ejemplo para practicar con selectores en jQuery

En el artículo anterior explicamos lo que eran los selectores de jQuery y los tipos de selectores básicos. Ahora, para que podamos ver por la práctica cómo funcionan cada uno de los selectores, hemos creado un ejemplo donde podremos escribir varios selectores y ver cómo funcionan, es decir, qué elementos de la página se consigue seleccionar con cada uno.

En este ejemplo tenemos una página que tiene varias etiquetas y un formulario. En el formulario hay un campo de texto y un botón. En el campo de texto podemos escribir cualquier selector y pulsando luego el botón, mediante jQuery, hacemos que parpadeen los elementos que concuerdan con ese selector.

El ejemplo puede verse en una página aparte.

> **Aclaración:** Por cierto, comento una cosa que resulta para la mayoría debe resultar obvia, pero quizás alguien pueda cometer el error. En los ejemplos del artículo anterior, escribíamos los selectores entre comillas, porque un selector es una cadena de caracteres. Pero en este caso, en la página del ejemplo, en el campo de texto hay que escribir los selectores sin las comillas. Si ponemos las comillas en realidad sería como intentar hacer un selector que incluyese el carácter comillas ". Esto es porque en el propio campo de texto cualquier cosa que escribamos ya es una cadena de caracteres de por si.

Para hacer este ejemplo tenemos que utilizar varios métodos y funciones jQuery de los cuales, casi todos, ya hemos hablado a lo largo del manual.

Veamos el formulario que hemos creado en la página:

```
<form>
Selector: <input type="Text" name="camposelector" id="camposelector">
<input type="button" id="boton" value="Ver qué elementos seleccionas">
</form>
```

Como se puede ver, tiene un campo INPUT de texto al que le hemos puesto un identificador para referirnos a él mediante jQuery. Fijarse también el INPUT para hacer un botón, al que también le pusimos un identificador.

Ahora veamos el código Javascript empleado:

```
$(document).ready(function(){
  $("#boton").click(function(evento){
    var selectorEscrito = $("#camposelector").attr("value");
    if (selectorEscrito==""){
      alert("Escribe algo en el campo de texto")
    }else{
      elementosSeleccionados = $(selectorEscrito);
      elementosSeleccionados.fadeOut("slow", function(){
        elementosSeleccionados.fadeIn("slow");
      });
    }
  });
});
```

Con document.ready() indicamos una función a invocar cuando la página está lista para recibir acciones de programación que modifiquen su estructura.

Con $("#boton").click() indicamos una función a ejecutar cuando se hace clic sobre el botón.

var selectorEscrito = $("#camposelector").attr("value");

Nos sirve para acceder al atributo value del campo de texto, es decir, a lo que haya escrito dentro.

Si no hay nada escrito en el campo, muestro un mensaje de alerta, porque en este caso el selector cadena vacía no sería valido y recibiríamos un mensaje de error.

Si había algo en el campo, pues selecciono con jQuery los elementos de la página que corresponden con el selector escrito en el campo de texto. Eso se hace con la línea:

elementosSeleccionados = $(selectorEscrito);

Luego, sobre el elemento o elementos seleccionados, invoco el método fadeOut(), que sirve para ocultar elementos de la página. A fadeOut() le paso dos parámetros, uno es la velocidad con la que tiene que hacer el efecto y otro es una función callback, a ejecutar sólo en el momento que el efecto haya concluido. Eso es con la línea:

elementosSeleccionados.fadeOut("slow", function(){

Por último, en la función callback realizamos una llamada al método fadeIn() sobre el mismo objeto jQuery resultado de aplicar el selector anterior, que sirve para que los elementos ocultados se muestren de nuevo en la página. Esto último con la línea:

elementosSeleccionados.fadeIn("slow");

En resumen, ocultando y mostrando luego los elementos de vuelta conseguimos ese parpadeo. Si nos resulta extraño este código, recordamos que en el Manual de jQuery de desarrolloweb.com ya hemos publicado varios artículos que aclaran los puntos tratados en este ejemplo, como los efectos rápidos o las funciones callback.

Código completo del ejemplo de selectores

Escribimos aquí para acabar el código completo de este ejemplo de trabajo con selectores.

```
<html>
<head>
  <title>Título de la página</title>
<style type="text/css">
.rojo{
  color: #cc0000;
}
.verde{
  color: #00cc00;
}
.azul{
  color: #0000cc;
}
.fondogris{
  background-color: #cccccc;
}
body{
  font-family: verdana, arial, helvetica;
}
div{
  margin-bottom: 4px;
}
</style>

<script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
  $("#boton").click(function(evento){
    var selectorEscrito = $("#camposelector").attr("value");
    if (selectorEscrito==""){
      alert("Escribe algo en el campo de texto")
    }else{
      elementosSeleccionados = $(selectorEscrito);
      elementosSeleccionados.fadeOut("slow", function(){
        elementosSeleccionados.fadeIn("slow");
      });
    }
  });
});
</script>
</head>

<body>
<h1>Selectores en jQuery</h1>
<p>En esta página hay varias etiquetas. Ahora con este formulario puedes escribir un selector, para seleccionar algunas con jQuery, y luego pulsar el botón para ver qué elementos de la página has seleccionado.</p>
<form>
Selector: <input type="Text" name="camposelector" id="camposelector">
<input type="button" id="boton" value="Ver qué elementos seleccionas">
</form>

<p id="p1" class="rojo">Este es un párrafo con id="p1" y class="rojo"</p>

<p id="p2" class="verde">Este es un párrafo con id="p2" y class="verde" y aquí <i>meto una itálica</i></p>

<p id="p3" class="rojo fondogris">Este es un párrafo con id="p3" y class="rojo fondogris" (es decir, este elemento tiene aplicadas las clases "rojo" y "fondogris"</p>

<p id="p4">Este es un párrafo con id="p4", sin class</p>
```

```
<p>Este es un párrafo con sin id ni class</p>

<div id="div1">Esto es una división con id="div1"</div>

<div id="div2" class="rojo">Esto es una división con id="div2" y class="rojo" y aqui <b>meto una
negrita</b></div>

<div id="div3" class="verde fondogris">Esto es una división con id="div3" y class="verde fondogris"</div>

<div>Esto es una división sin id ni class</div>

<div class="azul">Esto es una división sin id, con class="azul"</div>

<b>Esto es una etiqueta b</b>

<i>Esto es una etiqueta i</i>
</body>
</html>
```

Dejamos de nuevo el enlace para ver este ejemplo en marcha y practicar con los selectores de jQuery.

*Artículo por **Miguel Angel Alvarez***

Videotutorial: Selectores en jQuery

Presentamos un videotutorial de jQuery en el que mostramos el uso de los selectores de jQuery. Este vídeo sirve para ilustrar el Manual de jQuery que estamos publicando en DesarrolloWeb.com.

Es el cuarto vídeo que venimos publicando de este popular Framework Javascript, que pretende mostrar por la práctica todos los detalles para aprender jQuery de una manera sencilla. Por tanto, existen algunos vídeos anteriores, que conviene haber visto antes como comenzar a programar con jQuery y los vídeos sobre la función $ de jQuery.

Los vídeos de jQuery se publican con 10 minutos de duración, para poder subirlos a YouTube, por lo que las explicaciones y los ejemplos pueden resultar un poco rápidos. Por ello, queremos advertir que para seguirlo es muy recomendable tener cierto nivel de Javascript, así como unos conocimientos sobre el propio jQuery, por lo que conviene haber leído el artículo del manual donde se trata los selectores en jQuery y también el ejemplo para practicar con selectores.

En el vídeo veremos qué es un selector y cómo utilizarlo para obtener referencias a elementos de la página, sobre los que luego podemos realizar acciones. Luego nos detendremos en los distintos tipos de selectores y mostraremos brevemente su funcionamiento: selectores de etiqueta, identificador, clase, unión de selectores, etc. Además, luego hacemos un ejemplo interesante para poder escribir diversos selectores en un campo de texto y ver qué etiquetas o elementos de la página se seleccionan.

Sin más introducción, os dejamos con este vídeo tutorial sobre los selectores de jQuery.

*Artículo por **Miguel Angel Alvarez***

Selectores de Jerarquía en jQuery

En los últimos artículos del Manual de jQuery hemos hablado sobre los selectores. Como ya dijimos, sirven para seleccionar elementos de la página con los que queremos trabajar desde Javascript por medio del framework. En concreto vimos los selectores básicos, con los que podremos resolver la mayoría de nuestras necesidades en cuanto a selección de elementos.

No obstante, en jQuery existen varios otros tipos de selectores, junto con algunos filtros, que hacen todavía más potente el framework de cara a acceder a las etiquetas o elementos que deseamos seleccionar. Vamos a ver en este artículo qué son los selectores de jerarquía y algunos ejemplos de uso.

Sabemos que la página está compuesta por etiquetas HTML que se meten unas dentro de otras, formando una jerarquía de etiquetas o de elementos. Los selectores de Jerarquía permiten utilizar la propia estructura de la página para acceder a unos elementos dados, que se seleccionan a través de la jerarquía existente de etiquetas en la página. Dentro de éstos, existen a su vez varias posibilidades, que hacen uso de criterios de descendencia, ascendencia, siguiente, anterior, etc.

Selector ancestor descendant:
Sirve para seleccionar elementos de la página que son descendientes de otro y que además se corresponden con un selector dado. Para este selector se indican dos datos, separados por un espacio. Primero el selector para definir el elemento o elementos antecesores y el segundo selector para definir el tipo de elementos que se tienen que seleccionar de entre los

descendientes.

```
$("p b") //selecciona todas las etiquetas B que hay dentro de las etiquetas P
$("p.parraforojo i") //selecciona todas las etiquetas I que hay dentro de los párrafos con clase "parraforojo".
$("table.mitabla td") //selecciona todas las etiquetas TD que hay en las tablas que tienen class="mitabla"
```

Selector parent > child:
Con el selectorr parent > child podemos acceder a elementos que sean hijos directos de otros. Para ello indicamos un selector como "parent" y un selector como "child". Nos seleccionará todos los elementos que son hijos directos de parent y que concuerdan con el selector child.

```
$("p > b") //selecciona todas las etiquetas B que son hijas directas de los párrafos.
$("#capa > *") //selecciona todas las etiquetas que son hijas directas del elemento con id="capa"
```

Nota: la diferencia entre "ancestor descendant" y "parent > child" es que este último sólo selecciona los hijos directos. Por ejemplo, en el HTML siguiente:

```
<p><b>Párrafo</b> que tiene alguna <b>negrita</b> e <span class="algo"><i>itálica</i></span> para seleccionar</p>
```

$("p > b") seleccionaría los mismos elementos que $("p b"), porque en este caso todas las etiquetas B son hijas directas de P.

Pero en el caso de la itálica (etiqueta I), que está metida dentro del párrafo, pero dentro también de un span, $("p i") seleccionaría la etiqueta I por ser descendiente de P, pero $("p > i") no seleccionaría la etiqueta I, por no ser hija directa de P.

Selector prev + next:
Con este selector conseguimos acceder a las elementos que están después de otros, es decir, a las etiquetas que concuerdan con el selector "next", que se abren después de cerrar las etiquetas que concuerdan con el selector "prev".

```
$("p.parraforojo + p") //Esto selecciona los párrafos que están después de cualquier párrafo que tenga la clase "parraforojo"
$("i + b") //selecciona todas las negritas (etiqueta B) que hay después de una itálica (etiqueta I)
```

Selector prev ~ siblings:
Selecciona los elementos hermanos que hay a continuación de los elementos que concuerden con el selector "prev", que son del tipo que se especifica con el selector "siblings". Los elementos hermanos son los que están en el mismo contenedor y se encuentran en el mismo nivel de jerarquía.

```
$("#miparrafo ~ table") //selecciona los elementos TABLE que son hermanos del elemento con id="miparrafo"
$("#a2 ~ div.clase") //selecciona los elementos hermanos del que tiene el id="a2" que sean etiquetas DIV con la class="clase".
```

Probando los selectores jQuery de Jerarquía

Hemos hecho un rápido script que prueba los selectores de jerarquía que están disponibles en jQuery. Es una simple página que tiene una serie de elementos y un script para seleccionar y alterar su estilo. Los elementos los vamos seleccionando con diversos tipos de selectores de Jerarquía que hemos visto en este artículo de DesarrolloWeb.com. El ejemplo tendría el siguiente código:

```
<html>
<head>
  <title>Probando </title>
  <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
```

```
$(document).ready(function(){
  //selectores ancestor descendant
  $("p i").css("color", "#66F");
  $("table.mitabla td").css("background-color", "#55ff00");

  //selectores parent > child
  $("p.parraforojo > b").css("color", "red");
  $(".mitd > *").css("border", "1px solid #ff9900");

  //selectores prev + next
  $("i + b").css("font-size", "40px");
  $(".elemlista + li").css("opacity", 0.2);

  //selectores prev ~ siblings
  $("#a2 ~ div.clase").css("font-size", "180%");
  $("#miparrafo ~ table").css("border", "3px dotted #dd6600");
});
</script>
</head>

<body>

<p class="parraforojo">
<i>Hola</i> <b>esto</b> es un <b>párrafo</b> rojo <i>donde</i> he <b>puesto</b> unas <b>negritas</b>
</p>
<p class="parraforojo">Otro <b>con</b> clase class="parraforojo" <span class="unspan"><b>(esto está dentro de
unspan B, no depende directamente -no child- del párrafo)</b></span></p>
<p>Hola <b>esto</b> es otro <b>párrafo </b>para <i>poner</i> otras <b>negritas</b></p>
<p>hola!!!</p>
<table border=1>
<tr>
  <td><i>Tabla cualquiera</i></td>
  <td>Esta tabla <b>no tiene</b> class de <b>CSS</b></td>
</tr>
</table>
<p id="miparrafo">Este es el párrafo con id="miparrafo"</p>
<table class="mitabla" border=1>
<tr>
  <td colspan=2>Esta tabla tiene una <b>clase CSS</b></td>
</tr>
<tr>
  <td><i>class="mitabla"</i></td>
  <td class="mitd">Y este <b>td</b> le he puesto <i>class="mitd"</i> <span>Una cosa<span>otra
cosa</span></span></td>
</tr>
</table>
<p><b>Párrafo</b> que tiene alguna <b>negrita</b> e <span class="algo"><i>itálica</i></span> para
seleccionar</p>

<div>
<div id="a1">hola</div> <div id="a2">dos</div> <div id="a3">3</div> <span>Cuatro (no es un div)???</span>
<div id="a4" class="clase">Cuatro de verdad</div>
</div>

<ul>
<li>Elem 1</li>
<li class="elemlista">Elem 2</li>
<li>Elem 3</li>
<li>Elem 4</li>
<li class="elemlista">Elem 5</li>
<li class="elemlista">Elem 6</li>
<li>Elem 7</li>
</ul>
</body>
</html>
```

Podemos ver el ejemplo en marcha en una página aparte.

*Artículo por **Miguel Angel Alvarez***

Acceder y modificar atributos HTML desde jQuery

En este Manual de jQuery estamos recorriendo poco a poco la documentación del popular framework Javascript, para ofrecer a los lectores de DesarrolloWeb.com explicaciones detalladas de las clases y métodos disponibles. Le ha llegado el turno al método attr() que sirve para trabajar con los atributos de los elementos de la página. Este método, como muchos otros en jQuery tiene diferentes usos, dependiendo de los parámetros que le pasemos, pero siempre sirve para trabajar con los atributos HTML, como pueden ser title, height, width, href, value, etc.

El uso es bien simple. Dado un objeto jQuery, invocando el método attr() sobre él, podemos acceder a sus atributos, para recuperar sus valores, modificarlos o eliminarlos. Veremos los distintos usos conforme los parámetros que le pasemos.

Pero antes de empezar, vale la pena comentar que la información que encontraréis en este artículo se complementa con el siguiente texto, en el que veremos otros usos de la función attr().

Lectura de un atributo

El primer uso de attr() es para recuperar el valor de un atributo. En este caso, el método debe recibir una cadena con el nombre del atributo que queremos recuperar.

Ahora podríamos acceder a lo que hay escrito en el campo de texto de la siguiente manera:

```
$("#campotexto").attr("value")
```

Pero atención, en el caso que invoquemos el método attr sobre un objeto jQuery que contenga varios elementos a la vez, attr() en este caso devolvería el valor del atributo del primero de los elementos que haya en el objeto jQuery. Además, en caso que el elemento no tenga definido ese atributo al que se pretenda acceder, devolvería undefined.

Veamos un ejemplo, también simple, pero un poco más elaborado. Tenemos varios enlaces en la página, con este código HTML:

```
<a href="http://www.elpais.com" title="Diario El País">El País</a>
<br>
<a href="http://www.mozilla.org" title="Fundación Mozilla">Mozilla Fundation</a>
<br>
<a href="http://es.openoffice.org/" title="Siute de programas de oficina">Open Office</a>
```

Si hacemos algo como esto:

```
$("a").attr("title")
```

Obtendremos el valor del atributo title del primero de los enlaces. Como tenemos tres enlaces en la página, $("a") nos devolvería un objeto jQuery que contiene esos tres enlaces, pues recordar, que attr("title") devuelve el valor del atributo "title" del primero de los elementos del objeto jQuery. Ahora bien, si quisiéramos obtener el valor del atributo "title" de todos los elementos, tendríamos que hacer un recorrido a cada uno de los enlaces con el método each del core de jQuery http://www.desarrolloweb.com/articulos/core-each-jquery.html.

Veamos un ejemplo de una página completa que hace ese recorrido con each para recuperar todos los valores de los atributos title de los enlaces que haya en la página:

```
<html>
<head>
  <title>método attr</title>
  <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){

  $("a").each(function(i){
    var titulo = $(this).attr("title");
    alert("Atributo title del enlace " + i + ": " + titulo);
  });

});
</script>
</head>

<body>
<a id="enlace1" href="http://www.elpais.com" title="Diario El País">El País</a>
<br>
<a href="http://www.mozilla.org" title="Fundación Mozilla">Mozilla Fundation</a>
<br>
<a href="http://es.openoffice.org/" title="Siute de programas de oficina">Open Office</a>
</body>
</html>
```

Podemos ver el ejemplo en marcha en una página aparte.

Modificar un atributo

Ahora vamos a ver un uso de attr() en el que no leemos el atributo, sino que lo modificamos. En este caso la función recibe dos cadenas de texto, la primera con el nombre del atributo y la segunda con el nuevo valor que queremos asignar. Por ejemplo:

```
$('li').attr("type", "square");
```

Esto haría que todos los elementos de lista tengan un bullet de tipo cuadrado.

Si lo deseas, puedes ver el ejemplo en marcha en una página aparte.

Modificar varios valores de atributos a la vez

También podemos utilizar el método attr() pasando un objeto con pares atributo/valor. Esto sirve para modificar de una sola vez varios atributos sobre los elementos que haya en un objeto jQuery y si esos atributos no existían, simplemente los crea con los valores enviados en el objeto.

A estas alturas ya debemos saber crear variables con notación objeto, pero voy a dejar un ejemplo para que se pueda ver perfectamente este uso del método.

Imaginar que tenemos varios enlaces en la página, y que queremos modificar sus atributos, para todos los enlaces a la vez.

```
$('a').attr({
  'title': 'Title modificado por jQuery',
  'href': 'http://www.desarrolloweb.com',
  'style': 'color: #f80'
});
```

A partir de la ejecución de la sentencia anterior todos los title de los enlaces tendrán el valor "Title modificado por jQuery". Las URLs a las que enlazarán los link serán siempre la home de Desarrollo Web y además se les creará un estilo CSS para que sean de color naranja.

Podemos ver una página aparte con este ejemplo en marcha.

En el siguiente artículo veremos un ejemplo más elaborado sobre la modificación de atributos de elementos a través de la función attr(), en el que para obtener el valor del atributo a modificar utilizamos una función que pasamos también como parámetro a attr().

*Artículo por **Miguel Angel Alvarez***

Método attr() de jQuery, otros usos y removeAttr()

En el artículo anterior del Manual de jQuery ya comenzamos a explicar el método attr(), que pertenece al paquete de funciones para modificación de atributos de cualquiera de los elementos de una página web. En esta ocasión nos detendremos un uso adicionale del método attr(), que seguro nos resultarán útiles para mantener el control dinámico de los atributos de las etiquetas HTML, con sus correspondientes ejemplos. Este uso que nos faltaba por ver nos servirá cuando tenemos que asignar el valor de un atributo con la respuesta de una función Javascript.

Además veremos también en este artículo otro método relacionado que sirve para eliminar por completo un atributo de cualquier elemento de la página, el método removeAttr().

Asignar un valor de atributo procesado por una función

Podemos también enviar una función para procesar el valor que queremos asignar a un atributo. Para ello enviamos a attr() dos parámetros, el primero con el nombre del atributo y el segundo con la función que debe devolver el valor a asignar a dicho atributo.

Para ilustrar este uso de attr() mostraremos un ejemplo en el que desde jQuery accedemos a los elementos INPUT de la página que tienen la clase CSS "fecha" y le insertamos como texto a mostrar la fecha de hoy. Para obtener el día actual necesitamos procesar cierto código Javascript y para ello crearemos una función que devuelve la cadena de texto con la fecha.

```
$('input.fecha').attr("value", function(indiceArray){
  //indiceArray tiene el índice de este elemento en el objeto jQuery
  var f = new Date();
  return f.getDate() + "/" + (f.getMonth() +1) + "/" + f.getFullYear();
});
```

Para que se asimile mejor el uso de jQuery en una página, mostramos el código completo de este ejemplo.

```
<html>
<head>
  <title>método attr</title>
  <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){

  $('input.fecha').attr("value", function(indiceArray){
    //indiceArray tiene el índice de este elemento en el objeto jQuery
    var f = new Date();
```

```
    return f.getDate() + "/" + (f.getMonth() +1) + "/" + f.getFullYear();
  });

});
</script>
</head>

<body>
<form>
<input type="text" class="fecha">
<input type="text" class="nofecha">
<input type="text" class="fecha">
</form>

</body>
</html>
```

Si se desea, se puede ver en marcha el ejemplo en una página aparte.

Eliminar un atributo de uno o varios elementos con removeAttr()

Para acabar vamos a ver otro método distinto de los objetos jQuery, que sirve para borrar un atributo. Este sencillo método, llamado removeAttr(), simplemente recibe una cadena con el nombre del atributo que queremos eliminar y lo borra del elemento. Es decir, no es que se asigne un nuevo valor a un atributo, como ocurría con el método attr(), sino que ese atributo se borra por completo de la etiqueta, con lo cual no existirá en ningún caso, tomando el valor por defecto, si es que existe, que tenga configurado el navegador.

Para mostrarlo vamos a hacer un ejemplo en el que tenemos una celda de una tabla con nowrap, con lo que el texto de esa celda aparece todo en la misma línea. Luego quitamos el atributo y veremos que el texto de la celda se partirá en varias líneas. Esto lo hacemos simplemente enviando el valor "noWrap" al método removeAttr().

El código de este ejemplo es el siguiente.

```
<html>
<head>
  <title>método removeAttr</title>
  <script src="../jquery-1.3.2.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){

  $("#boton").click(function(i){
    $("td").removeAttr("noWrap");
  });

});
</script>
</head>

<body>
<table width="50">
<tr>
<td nowrap>
Esta celda tiene un nowrap, con lo que todo el texto se muestra en la misma línea!
Pero realmente la tabla mide 50 pixeles de anchura, luego tendrían que aparece varias líneas!
</td>
</tr>
</table>
```

```
<input type="Button" id="boton" value="Quitar nowrap">

</body>
</html>
```

Un detalle es que en la línea que se hace la llamada al método removeAttr("noWrap"), el nombre del atributo "noWrap" tiene que estar escrito con la "W" mayúscula para que funcione en Explorer.

Podemos ver el ejemplo de removeAttr() en marcha en una página aparte.

*Artículo por **Miguel Angel Alvarez***

Funciones CSS de jQuery para conocer el tamaño y posición de elementos

Entre las clasificaciones de funciones jQuery que existen diversas que sirven para controlar los atributos de CSS de los elementos de la página, ya sea para acceder a los valores actuales de los atributos CSS o para alterarlos. En artículos anteriores del Manual de jQuery pudimos conocer varias de estas funciones, por ejemplo en el artículo de Añadir y quitar clases CSS sobre elementos.

En este artículo vamos a ver otras de las funciones que pone a nuestra disposición jQuery para acceder a la posición de los elementos en la página y a sus dimensiones. Estas funciones, aunque estaría mejor llamarles métodos (ya que pertenecen al objeto jQuery), son meramente informativas, para saber dónde están posicionados los elementos dentro del documento y sus medidas internas y externas. Lo veremos con detalle en breve, pero antes quiero señalar para los despistados que si queremos alterar las propiedades CSS de un elemento de la página con jQuery recordemos que está disponible el método css(), que hemos visto anteriormente en repetidas ocasiones a lo largo de este manual, enviándole como primer parámetro el nombre del atributo CSS a alterar y como segundo parámetro el valor del mismo.

Ahora voy a dar un listado de los métodos nuevos que vamos a ver en este artículo, comenzando por los que sirven para conocer las dimensiones de un elemento.

Métodos innerWidth() e innerHeight():
Reciben un objeto jQuery y devuelven las dimensiones internas del primer elemento que haya en dicho objeto jQuery, esto es, la anchura y altura respectivamente del elemento contando el padding del elemento pero no el borde.

Métodos outerWidth() e outerHeight():
Reciben un objeto jQuery y devuelven las dimensiones externas del primer elemento de dicho objeto jQuery recibido por parámetro, esto es, la anchura y altura respectivamente del elemento contando el padding del elemento y su borde.

> **Nota:** Como podremos imaginarnos, si un elemento no tiene borde los valores de innerWidth e outerWidth serán exactamente los mismos, así como los valores de innerHeight y outerHeight.

Métodos offset() y position():
Ambos métodos devuelven la posición de un elemento en la página. Reciben un objeto jQuery y devuelven la localización del primer elemento que haya en ese objeto jQuery. La posición siempre se indica como valor de retorno del método por medio de un objeto que tiene dos

atributos, "top" y "left", indicando los píxeles que está separado de la esquina superior izquierda del documento. La diferencia entre estos dos métodos es que offset() indica la posición del elemento real, teniendo en cuenta los márgenes del elemento, lo que suele ser más útil. Por su parte, position() indica la posición donde habría sido posicionado el elemento si no tuviera márgenes, lo que a menudo no es la posición real.

Nota: Para acceder a los valores top y left del objeto de retorno podemos hacer algo así:

```
posicionReal = $("#idelemento").offset();
alert(posicionReal.top);
alert(posicionReal.left);
```

Función que muestra las dimensiones de un elemento

Por hacer unas pruebas con estos métodos, vamos a comenzar creando una función que muestra en una caja de alerta las dimensiones de un elemento cuyo selector se envíe por parámetro. A la función enviaremos el selector y luego con jQuery mostraremos sus valores de anchura y altura, tanto de la parte interior del elemento (innerWidth e innerHeight), como del elemento completo con su borde (outerWidth y outerHeight).

```
function dimensionCapa(capa){
capa = $(capa);
var dimensiones = "";
dimensiones += "Dimensiones internas: " + capa.innerWidth() + "x" + capa.innerHeight();
dimensiones += "\nDimensiones externas: " + capa.outerWidth() + "x" + capa.outerHeight();
alert(dimensiones);
}
```

Como decíamos, las dimensiones externas toman en cuenta el borde del elemento, si es que tiene, y las dimensiones internas no toman en cuenta el posible borde.

Función para mostrar la posición de un elemento

Ahora vamos a hacer una función similar a la anterior para mostrar un ejemplo de uso de las funciones position() y offset(). Esta función recibe un selector y muestra la localización de este elemento, tal como me la devuelven los métodos position() y offset().

```
function posicionCapa(capa){
capa = $(capa);
var posicion = "";
posicion += "Posición relativo al documento:\nLEFT: " + capa.offset().left + "\nTOP:" + capa.offset().top;
posicion += "\n\nPosición si no tuviera margen:\nLEFT: " + capa.position().left + "\nTOP:" + capa.position().top;
alert(posicion);
}
```

Si invocamos esta función sobre un elemento cualquiera que no tenga margen, las dos posiciones devueltas por position() y offset() serán las mismas, pero si aplicamos un margen a ese elemento, el elemento cambiará de lugar en la página y entonces el valor de offset() también cambirá, pero no el de position().

Ejemplo completo sobre los métodos de dimensiones y posición de elementos

Las dos funciones anteriores las podemos ver en marcha en un ejemplo que hemos creado

para poder explicar mejor todos los métodos comentados en este artículo de DesarrolloWeb.com.

En el ejemplo simplemente se realizan las acciones para averiguar las posiciones y dimensiones de un par de elementos de la página. Además, tenemos un par de botones para alterar el CSS de los elementos dinámicamente y así volver a ver sus posiciones y dimensiones y comprobar cómo han cambiado.

Realmente no sirve de mucho el ejemplo, pero al menos esperamos que resultará bastante didáctico. Podemos verlo en marcha en una página aparte.

Ahora el código de este ejemplo, que no debería resultar muy complicado si hemos seguido el manual de jQuery hasta este punto.

```
<html>
<head>
<title>Funciones CSS en jQuery</title>
<script src="../jquery-1.4.1.min.js"></script>
<script type="application/x-javascript">
function dimensionCapa(capa){
capa = $(capa);
var dimensiones = "";
dimensiones += "Dimensiones internas: " + capa.innerWidth() + "x" + capa.innerHeight();
dimensiones += "\nDimensiones externas: " + capa.outerWidth() + "x" + capa.outerHeight();
alert(dimensiones);
}
function posicionCapa(capa){
capa = $(capa);
var posicion = "";
posicion += "Posición relativo al documento:\nLEFT: " + capa.offset().left + "\nTOP:" + capa.offset().top;
posicion += "\n\nPosición si no tuviera margen:\nLEFT: " + capa.position().left + "\nTOP:" + capa.position().top;
alert(posicion);
}
$(document).ready(function(){
$("#botondimensiones").click(function(){
dimensionCapa("#capa1");
});
$("#botonposicion").click(function(){
posicionCapa("#capa1");
});
$("#botontamano").click(function(){
$("#capa1").css("width", 200);
});
$("#botonmargen").click(function(){
$("#capa1").css("margin", 20);
});
$("#botondimensionesc2").click(function(){
dimensionCapa("#capa2");
});
$("#botonposicionc2").click(function(){
posicionCapa("#capa2");
});

});

</script>

</head>
<body>
<h1>Funciones CSS en jQuery de dimensiones y posición</h1>
<p>Probando funciones de localización de elementos en la página...</p>
<div id="capa1" style="padding: 24px; background-color: #ffccdd; float: left; border: 2px dotted #666;">
<h2>capa1:</h2>
Voy a crear esta capa para ver lo que mide y donde está posicionada.
```

```
</div>
<br style="clear: both;">
<div style="margin: 10px;">
<button id="botondimensiones" type="button">Dimensiones de capa1</button>
<button id="botonposicion" type="button">Posicion de capa1</button>
<button id="botontamano" type="button">Cambiar tamaño capa1</button>
<button id="botonmargen" type="button">Cambiar margen capa1</button>
</div>

<div style="margin: 10px;">
<button id="botondimensionesc2" type="button">Dimensiones de capa2</button>
<button id="botonposicionc2" type="button">Posicion de capa2</button>
</div>

<br>
Desplaza la página hacia abajo para ver la capa2...
<br>
<br>
...
<br>
<div id="capa2" style="background-color:#ccc; border-bottom: 5px solid #999; margin-left: 10px;">
Esta capa está muy hacia abajo!!
</div>
</body>
</html>
```

Para acabar, podemos ver este script funcionando en una página aparte.

*Artículo por **Miguel Angel Alvarez***