Manual Ajax práctico - Taller Ajax

Un manual que explica Ajax por la práctica, con una serie de ejemplos de códigos que usan javascript, HTML y XML para crear aplicaciones Ajax.

Autores del manual

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

Pablo Lecce
Programador autodidacta
http://www.rhosting.com.ar
(3 capítulos)

Andrés Fernández
http://www.disegnocentell.com.ar
(1 capítulo)

Damián Suárez
Impulsor de XiFOX.net
http://www.cabezaderaton.com.ar
(1 capítulo)

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

Fernando G. Muñoz
Desarrollador web, especialista en ASP.Net y desarrollos con arquitectura SOA.
http://marferdotnet.blogspot.es
(1 capítulo)

Enviar mediante POST y GET usando una sola funcion AJAX

Cuando se realiza una web que usa AJAX, el problema mas frecuente es tener que escribir una funcion para cada variable o conjunto de variables que se quiere pasar. Esto suma mucho peso al código de tu sitio web.

En este tutorial te contamos como crear una sola funcion que te permita pasar variables mediante GET y POST entre dos páginas web usando AJAX .

Esto aligerará mucho el peso de tus archivos javascript y de tus páginas ya que usarás una funcion para todo y no una para cada variable o conjunto de variables que desees pasar.

ANTES DE EMPEZAR

Este tutorial esta hecho para personas que saben cómo crear objetos AJAX, escribir funciones y pasarlas mediante AJAX por POST o GET. Tambien que tienen conocimientos sobre PHP y javascript. Si no es tu caso, por favor profundiza en dichos aspectos a fin de entenderlo.

EL CODIGO

Primero copio aqui el codigo completo, y luego pasaré a analizarlo.

```
<script>

function objetus(file) {

xmlhttp=false;

this.AjaxFailedAlert = "Su navegador no soporta las funcionalidades de este sitio y podria
experimentarlo de forma diferente a la que fue pensada. Por favor habilite javascript en su
navegador para verlo normalmente.\n";

this.requestFile = file;

this.encodeURIString = true;

this.execute = false;

if (window.XMLHttpRequest) {

this.xmlhttp = new XMLHttpRequest();

if (this.xmlhttp.overrideMimeType) {

this.xmlhttp.overrideMimeType('text/xml');

}

}

else if (window.ActiveXObject) { // IE

try {

this.xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");

}catch (e) {

try {

this.xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
```

```
} catch (e) {
this.xmlhttp = null;
}
}
if (!this.xmlhttp && typeof XMLHttpRequest!='undefined') {
this.xmlhttp = new XMLHttpRequest();
if (!this.xmlhttp){
this.failed = true;
}
}
}
return this.xmlhttp ;
}
function recibeid(_pagina,valorget,valorpost,capa){
ajax=objetus(_pagina);
if(valorpost!=""){
ajax.open("POST", _pagina+"?"+valorget+"&tiempo="+new Date().getTime(),true);
} else {
ajax.open("GET", _pagina+"?"+valorget+"&tiempo="+new Date().getTime(),true);
}
ajax.onreadystatechange=function() {
if (ajax.readyState==1){
document.getElementById(capa).innerHTML = "<img src='loadingcircle.gif' align='center'> Aguarde por
favor...";
}
if (ajax.readyState==4) {
if(ajax.status==200)
{document.getElementById(capa).innerHTML = ajax.responseText;}
else if(ajax.status==404)
{
capa.innerHTML = "La direccion no existe";
}
else
{
capa.innerHTML = "Error: ".ajax.status;
```

```
}
}
}
if(valorpost!=""){
ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
ajax.send(valorpost);
} else {
ajax.send(null);
}
}
</script>
```

Artículo por Pablo Lecce

Pasar valores por GET o POST mediante AJAX - Explicando el código

Bien, aqui te pasaremos a explicar el código de la funcion a fin que podas entenderlo mejor.

El codigo tiene dos funciones.

La primera es la función que carga el objeto AJAX propiamente dicho. Si bien es compleja, su explicacion no es objeto de este tutorial, y podes usar cualquier funcion para la carga del objeto xhttprequest que vengas usando previamente.

La funcion recibeid es la que se encarga de pasar valores entre páginas mediante AJAX, ya sean estos mediante GET o mediante POST.

Para ello usa 4 variables:

1. _pagina por donde le paso la url de la página que deseo cargar
2. valorget por donde le paso las variables get que deseo pasar
3. valorpost por donde le paso las variables post que deseo pasar
4. capa donde indico el DIV o la capa donde se cargará la página que se solicite mediante la funcion.

DESGLOSANDO LA FUNCION

¿Envio por GET o por POST?

Primeramente mediante el siguiente codigo

```
if(valorpost!=""){

ajax.open("POST", _pagina+"?"+valorget+"&tiempo="+new Date().getTime(),true);

} else {
```

```
ajax.open("GET", _pagina+"?"+valorget+"&tiempo="+new Date().getTime(),true);


}
```

La funcion determina el método que usará el objeto AJAX para enviar las variables a la página. Como sabes, si uno envia por metodo POST esto se hace de forma diferente a cuando envias mediante GET.

Adicionalmente sucede que si envias mediante GET y hay variables POST, las mismas no serán pasadas. Por ello la utilidad de este condicional es saber si hay variables POST que deben ser pasadas, setear el método a POST y sino dejarlo en GET.

La siguiente parte del código básicamente verifica los estados. Mientras la página esta siendo llamada carga una coqueta imagen de cargando, aunque podes reemplazarla por una frase si deseas.

Y una vez que recibe los resultados, los carga en la capa.

Finalmente la otra parte importante de la funcion

Mediante el siguiente condicional, se complementa el primer condicional, enviando los datos de la solicitud mediante POST o GET segun corresponda, con el codigo adecuado para ajax.send.

```
if(valorpost!=""){


ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");


ajax.send(valorpost);


} else {


ajax.send(null);


}
```

Artículo por Pablo Lecce

Enviar mediante POST y GET usando una sola funcion AJAX - Ejemplos de uso

Bien, ahora vamos a ver un par de ejemplos de uso de esta funcion.

El ejemplo mas simple es para pasar valores mediante GET. Para ello, por ejemplo, si usas un enlace el codigo debe lucir similar al siguiente:

```
<a href="javascript:recibeid ('http://www.misitio.com/mipagina.php',
'variableenviada=enviaste get,'','micapa')">Mi Enlace GET</a>
```

Si usas para enviar variables POST, tenes 2 opciones.

Si lo haces mediante enlace luciria similar a esta forma:

```
<a href="javascript:recibeid('http://www.misitio.com/mipagina.php',

'variablegetenviada=enviaste get','variablepostenviada=y enviaste post',

'micapa')">Mi Enlace POST</a>
```

Sin embargo para el envio mediante formulario hay 2 peculiaridades que debes conocer.

La primera es que en el tag de apertura del form debes incluir un return false, por ejemplo, debe lucir algo asi:

```
<form name="Miformulario" onSubmit="return false">
```

Y la segunda es que en el tag del boton debes incluir con un onclick la funcion y escribir las variables a pasar de un modo particular para que las tome.

Aqui un ejemplo:

```
<input name="Submit" type="submit" value="Enviar"

onClick="recibeid('http://www.misitio.com/mipagina.php',

'variablegetenviada=enviaste get','variablepostenviada1='+

Miformulario.campoparalavariablepostenviada1.value+'

&variablepostenviada2='+Miformulario.campoparalavariablepostenviada2.value+

'','micapa')" >
```

Podes ver este ejemplo funcionando haciendo click aqui

Bien, eso es básicamente todo. Resta que hagas tus propios experimentos con ella.

Desde ya que estamos abiertos a tus comentarios y mejoras.

Artículo por Pablo Lecce

Un ejemplo de Ajax sin XMLHttpRequest

Estoy harto de muchas cosas, pero estas dos son destacables:

1)Que la gente (sugestionada por las muchas herramientas creadas con apoyo en esta tecnología) se crea que Ajax sirve para todo... El otro día uno me dijo que Ajax era algo que servía para hacer drag and drop... Otro pensaba que era algo que servía para generar efectos de reflejo en imágenes... Y así podría seguir enumerando a algunos que piensan que mejora el gusto del café o que preguntan "cómo se te ocurre usar un detergente para hacer una web?".

2)Todos los websites de programación que enseñan a hacer combos relacionados, ya sea con javascript puro o con Ajax.

Y bueno, un poco para sumar yo también a este tema recurrente, y otro poco porque es un buen ejemplo de que a veces es mejor no usar Ajax sino herramientas similares, más sencillas y que obtienen el mismo resultado, les dejo este ejemplo, en el cual el evento onchange de un combo dispara una consulta al servidor, obtiene la respuesta que este le devuelve y la muestra en pantalla sin refrescar la página, es decir, de manera asíncrona, y, lo más importante, sin usar Ajax, sólo utilizando DOM javascript, algo que algunos llaman RPC javascript o Llamada a Procedimiento Remoto en javascript.

Este es el ejemplo y este es el código utilizado:

```
<?php
if(isset($_GET['p'])){
switch($_GET['sel']){
case '1':
$ret=array('Final del Juego','Rayuela','El Señor de loas Anillos');
break;
case '2':
$ret=array('rock','new age');
break;
case '3':
$ret=array('español','php','javascript');
break;
default:
echo 'document.getElementById("pp").innerHTML="<select name=\"dos\" id=\"dos\"></select>";';
exit;
}
$html='<select name=\"dos\" id=\"dos\">';
foreach($ret as $v)
$html.='<option value=\"'.$v.'\">'.$v.'</option>';
$html.='</select>';
echo 'document.getElementById("pp").innerHTML="'.$html.'";';
exit;
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>test</title>
<script>
function adjs(url){
oldsc=document.getElementById("old_sc");
if(oldsc)
document.getElementsByTagName('body')[0].removeChild(oldsc);
sc=document.createElement('script');
sc.id="old_sc";
sc.src=url+'&'+Math.random();
document.getElementsByTagName('body')[0].appendChild(sc);
}
</script>
</head>

<body>
<form id="form1" name="form1" method="post" action="">
<select name="uno" id="uno" onchange="adjs('?p&sel='+this.value)">
<option value="0">seleccionar</option>
<option value="1">libros</option>
<option value="2">música</option>
<option value="3">lenguaje</option>
</select>
<div id="pp"><select name="dos" id="dos">
</select></div>
</form>
</body>
</html>
```

Como ven, el corazón de esto es esta función:

```
<script>
function adjs(url){
oldsc=document.getElementById("old_sc");
if(oldsc)
document.getElementsByTagName('body')[0].removeChild(oldsc);
sc=document.createElement('script');
sc.id="old_sc";
sc.src=url+'&'+Math.random();
```

```
document.getElementsByTagName('body')[0].appendChild(sc);
}
</script>
```

Lo que hace la misma es incluir un nuevo elemento script en la página, cuyo atributo src es la ruta al archivo de proceso en el servidor.

Simple, sencilla, totalmente compatible con todos los navegadores modernos, y, gracias a que mediante una comprobación elimina los scripts incluídos en otras llamadas, con un consumo mínimo de recursos.

Otra ventaja frente a nuestro amigo XMLHttpRequest, aparte de su sencillez, es que puede trabajar con archivos que estén en otro dominio sin tener que apelar a ningún truco.

Artículo por Andrés Fernández

Ajax File Upload

Se ha escrito y discuto mucho sobre si es o no es factible utilizar AJAX para subir un archivo al servidor.

Por una necesidad personal yo también me vi involucrado en esta serie de discusiones. La finalidad de este artículo es que le sea útil a alguien; principalmente a mi.

Entonces ...
Al finalizar este pequeño tutorial deberías entender a la perfección este ejemplo.

Primer Error: Titulo Mal Empleado

Me voy a disfrazar de traductor en Inglés. Sería algo como Archivo Subido con AJAX. Esto, hoy por hoy, no se puede hacer. Es así y no hay vuelta que darle.
Entonces usted dirá ...
- heeee, si yo en gmail puedo subir archivos y lo hago con AJAX !!!.
Yo le respondo ...
- No sea bobo, yo también creía lo mismo.

Hay sobradas muestras que el objeto XMLHttpRequest no puede enviar archivos al servidor.
Entonces deberíamos cambiar el titulo del post. Sería así:

Ajax File Upload SIN Ajax

Frente a esta inmutable condición los desarrolladores han buscado una buena forma de simular scripts para subir archivos al servidor como si fuese con AJAX, y por lo menos conmigo lo lograron.

- Tenemos un formulario con un campo tipo file, el cual nos permitirá enviar el archivo a nuestro servidor PHP. La respuesta del servidor se hará en el IFRAME, actor principal de esta novela.
- El script ejecutado en nuestro server a causa del action del formulario es el encargado de copiar el archivo temporal en un espacio físico dentro del mismo.
- Luego, con un poco de ayuda de JS imprimiremos un mensaje correspondiente al estado final de nuestro script.

Primer Ejemplo - Subir un Archivo el Server

Utilizaremos tan solo dos scripts muy sencillos. Luego iremos mejorando y complementando el desarrollo hasta llegar a nuestro objetivo.

HTML:

```
<form method="post" enctype="multipart/form-data"
action="controlUpload.php"
target="iframeUpload">
<input type="hidden" name="phpMyAdmin" />
Archivo: <input name="fileUpload" type="file" />
<input type="submit" value="enviar">
<iframe name="iframeUpload"></iframe>
</form>
```

Es es un simple formulario HTML con un campo FILE y un marco flotante iframe denominado iframeUpload. Cuando enviamos el archivo al servidor ejecutaremos el script controlUpload.php y la respuesta del server se hará en nuestro iframe ya que apuntamos al mismo dentro de la etiqueta target en la declaración form.

PHP:

```
// Script Que copia el archivo temporal subido al servidor en un directorio.
echo '<p>Nombre Temporal: '.$_FILES['fileUpload']['tmp_name'].'</p>';
echo '<p>Nombre en el Server: '.$_FILES['fileUpload']['name'].'</p>';
echo '<p>Tipo de Archivo: '.$_FILES['fileUpload']['type'];
$tipo = substr($_FILES['fileUpload']['type'], 0, 5);
// Definimos Directorio donde se guarda el archivo
$dir = 'archs/';
// Intentamos Subir Archivo
// (1) Comprovamos que existe el nombre temporal del archivo
if (isset($_FILES['fileUpload']['tmp_name'])) {
// (2) - Comprovamos que se trata de un archivo de imágen
if ($tipo == 'image') {
// (3) Por ultimo se intenta copiar el archivo al servidor.
if (!copy($_FILES['fileUpload']['tmp_name'], $dir.$_FILES['fileUpload']['name']))
echo '<script> alert("Error al Subir el Archivo");</script>';
}
else echo 'El Archivo que se intenta subir NO ES del tipo Imagen.';
}
else echo 'El Archivo no ha llegado al Servidor.';
```

Cuando se ejecuta el script, este intenta rescatar los datos del archivo a subir. Para eso se vale del array global de PHP $_FILES. Si no estas muy familiarizado recomiendo que leas esta sección del manual oficial para entender su funcionamiento y sus particularidades.

En las primeras líneas presentamos el nombre del archivo temporal generado por nuestro motor PHP, el segundo es el nombre real del archivo y el tercero es el tipo de archivo.
Si estos valores están definidos evidentemente el archivo llego al server. Estas líneas mas adelante no las utilizaremos; ahora sólo las usamos para realizar el seguimiento del script.
Asignamos a $tipo el tipo de archivo para poder controlar que sea una imágen. Se cortan los primeros 5 caracteres y si todo es correcto $tipo tendrá el valor image. Si no es una imagen tendrá un valor distinto. Este control se puede hacer de diversas formas; nosotros utilizamos esta.

Luego, intentamos copiar el archivo temporal en forma definitiva en algún sector físico en nuestro servidor. Se verifica en forma anidada definición de nombre temporal de archivo, tipo de archivo y si el motor concretó la copia.
En este ejemplo lo hacemos en la carpeta archs/.
Si está todo bien podemos pasar al segundo paso.
En este link podemos probar el script. Podemos ver los archivos subidos al server en este link.

Aclaraciones.
Hay que definir permisos de escritura de PHP en el directorio a copiar el archivo. En nuestro caso es la carpeta archs.

Armando el Circo

Empecemos a modificar un poco los archivos anteriores.
En primer lugar vamos a eliminar el botón submit Enviar y vamos a disparar el formulario con una orden en JS.

JavaScript:

```
onchange="javascript: submit()"
```

Vamos a ocultar el iframe.

JavaScript:

```
style="display:none"
```

Al final del archivo agregamos una línea que nos dirige a un simple script que nos muestra los archivos subidos al server; más que nada para que puedas corroborar la funcionalidad.

HTML:

```
<a href="verArchivos.php">Ver Archivos</a>
```

Esta línea no se presenta en el script.

HTML:

```
  <form method="post" enctype="multipart/form-data"
action="controlUpload2.php" target="iframeUpload">
Archivo: <input name="fileUpload" type="file" onchange="javascript: submit()" />
<br /><iframe name="iframeUpload" style="display:none"></iframe>
</form>
```

Y al script controlUpload.php le vamos a agregar alertas de control (también con JS por ahora). Además vamos a eliminar sentencias que se han vuelto innecesarias al ocular el iframe.

PHP:

```
// Script Que copia el archivo temporal subido al servidor en un directorio.
$tipo = substr($_FILES['fileUpload']['type'], 0, 5);
// Definimos Directorio donde se guarda el archivo
$dir = 'archs/';
// Intentamos Subir Archivo
// (1) Comprovamos que existe el nombre temporal del archivo
if (isset($_FILES['fileUpload']['tmp_name'])) {
// (2) - Comprovamos que se trata de un archivo de imágen
if ($tipo == 'image') {
// (3) Por ultimo se intenta copiar el archivo al servidor.
if (!copy($_FILES['fileUpload']['tmp_name'], $dir.$_FILES['fileUpload']['name']))
echo '<script> alert("Error al Subir el Archivo");</script>';
else
echo '<script> alert("El archivo '.$_FILES['fileUpload']['name'].' se ha copiado con
Exito");</script>';
}
else echo '<script> alert("El Archivo que se intenta subir NO ES del tipo Imagen.");</script>';
}
else echo '<script> alert("El Archivo no ha llegado al Servidor.");</script>';
```

controlUpload2.php | Ver

Ya se empieza a parecer a un verdadero AJAX FILE UPLOAD. Que siga el circo !. Show must go on.

Finalmente un sencillo Script

Con estas últimas modificaciones tendremos la base final para poder implementar nuestro FILE UPLOAD con un comportamiento muy similar a gmail.

Al formulario lo vamos a contener en un div con id formUpload.

HTML:

```
<div id="formUpload">
```

Al archivo upload3.php, aparte del formulario de envío le vamos a agregar una sencilla función resultadoUpload (estado, file) realizada en JS que, dependiendo del código que nos 'envíe' controlUpload.php en las variales estado y file (ahora vemos como ...) imprimirá en dicho div un mensaje de respuesta al intento de subida.

JavaScript:

```
function resultadoUpload(estado, file) {
var link = '<br /><br /><a href="upload3.php">Subir Archivo</a> - <a href="verArchivos.php">Ver
Imagenes</a>';
if (estado == 0)
var mensaje = 'El Archivo <a href="archs/' + file + '" target="_blank">' + file + '</a> se ha subido
al servidor correctamente' + link;
if (estado == 1)
var mensaje = 'Error ! - El Archivo no llego al servdor' + link;
if (estado == 2)
var mensaje = 'Error ! - Solo se permiten Archivos tipo Imagen' + link;
if (estado == 3)
var mensaje = 'Error ! - No se pudo copiar Archivo. Posible problema de permisos en server' + link;
document.getElementById('formUpload').innerHTML=mensaje;
}
```

La impresión de nuestro mensaje en el div lo hace en la última linea de la función.

JavaScript:

```
document.getElementById('formUpload').innerHTML=mensaje;
```

Este sitio tiene una buena descripción del conjunto de funciones getElementBy*. De todas formas, con un poco de tu gran astucia podrás encontrar muchos ejemplos de su funcionamiento.

Ahora solo resta modificar controlUpload.php para que en vez de ejecutar una ventana alert () (como en el ejemplo anterior) simplemente ejecute el código JavaScript para llamar a la función resultadoFile () enviándole los datos correspondientes al intento de subida del archivo.
El punto llamativo es como accedemos desde el iframe oculto a la función que se encuentra en la página que lo contiene. Utilizamos la palabra reservada de JS parent.

JavaScript:

```
<script>parent.resultadoUpload(estado, file);</script>
```

Las otras modificaciones son para mejorar estéticamente las páginas. También se utilizan dos scrips en php. verArchivos.php para ver los archivos subidos y eliminar.php que no requiere demasiada explicación.

- upload3.php | ver | ejecutar
- controlUpload3.php | ver
- verArchivos.php | ver | ejecutar
- eliminar.php | ver
- Conjunto de todos los scripts upLoader.tar.gz y upLoader.rar

Finalizando

Este humilde tutorial se hizo más largo de lo que hubiese querido. Voy a seguir trabajando para mejorar el funcionamiento y evolucionarlo. Queda pendiente una barra de progreso (ahí creo que no podemos escapar de AJAX) y otras formas de aplicación.
Tomen al mismo y a todo el código como una guia par desarrollar esta pequeña aplicación. Faltan muchos controles de

errores, de seguridad, etc. El código fuente presentado no es idéntico a los archivos que se pueden descargar; esto es simplemente por una cuestión de prolijidad.
Disculpen mi desorden semántico y gramatical; como la mayoría, soy atrevido al jugar el papel de tutor. Nada más lejos de mis fines. Simplemente lo mío es ayudar como así también me han ayudado.

Artículo por Damián Suárez

Uso de Ajax muy sencillo con jQuery

Es muy fácil desarrollar Ajax en jQuery. En este artículo veremos el ejemplo más sencillo de Ajax con el framework Javascript 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

Vemos más posibilidades de Ajax en jQuery, modificando un ejemplo anterior, para crear un mensaje de carga mientras que el usuario espera la respuesta Ajax del servidor.

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

Vídeo sobre cómo realizar un script Ajax utilizando el framework Javascript jQuery. En este videotutorial mostraremos el proceso paso a paso al alcance para cualquier programador.

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

Ajax con Mootools, ejemplo sencillo

Ejemplo de programación Ajax con Mootools. Mostramos un texto traído con Ajax en una caja de alerta.

Vamos a ofrecer unas notas para aprender a programar con Ajax usando el framework Javascript Mootools, a través de la realización de un sencillo ejemplo. El ejercicio consiste en crear un enlace que, al pulsarlo, realizará una conexión con Ajax para traerse los contenidos de un archivo HTML, para mostrarlos en una caja de alerta.

Para los que no sepan lo que es Mootools, debemos referirles a nuestro Manual de Mootools. En este artículo vamos a dar por sabidas muchas cosas de Mootools, pero esperamos que con unos mínimos conocimientos se pueda seguir el ejemplo. La versión del framework que estamos utilizando en estos momentos es Mootools 1.2.

Para realizar estos ejemplos sencillos de Ajax utilizaremos la clase Request de Mootools. Dicha clase es como un envoltorio del XMLHttpRequest de Javascript, que se utiliza para crear conexiones Ajax.

Para realizar una conexión Ajax tenemos que instanciar un objeto de la clase Request, que se crea indicando una serie de opciones y eventos, que nos servirán para configurar la conexión y especificar las acciones a realizar durante el proceso de ejecución de la conexión. En este primer caso veremos un reducido juego de opciones y eventos de los que podemos utilizar en la clase.

Podemos ir viendo el objetivo a conseguir en este ejercicio.

Código HTML

Antes que nada, veamos el código HTML que necesitaríamos para crear un enlace, luego modificaremos su comportamiento con Mootools.

```
<a href="javascript:void(0)" id="mienlace">Pulsa este enlace</a>
```

El enlace tiene dos cosas que comentar. Primero el href="javascript:void(0)", que sirve para que el enlace no haga nada al pulsarse. Luego id="mienlace", que sirve para darle un nombre a ese enlace para poder referirnos a él desde Javascript.

Configurar una conexión Ajax con Request de Mootools

Ahora veamos qué se debería hacer para configurar una conexión conAjax mediante la clase Request de Mootools.

```
var prueboRequest = new Request({
   method: 'get',
   url: 'archivo-html-solicitud.html',
   onRequest: function() { alert('Acabas de iniciar una solicitud por Ajax!'); },
   onSuccess: function(texto, xmlrespuesta){ alert('Respuesta:\n' + texto);},
   onFailure: function(){alert('Fallo');}
}).send();
```

Como se puede ver, se ha creado un objeto prueboRequest que es una instancia de la clase Request. Para configurar la conexión se envían una serie de opciones:

method:
Sirve para definir que el método de conexión. En nuestro ejemplo será por GET

url:
Con esto indicamos la URL a la que queremos conectar con Ajax, para traernos el contenido de la misma. Simplemente indicamos la ruta relativa donde está el archivo al que nos conectamos. Por cierto que el archivo 'archivo-html-solicitud.html' debemos haberlo creado en nuestro servidor. Tal como está la ruta al archivo se entiende que se ha colocado en el mismo directorio que el script.

onRequest:
Esto sirve para definir un evento en la conexión, que se ejecutará cuando se inicie la conexión con el archivo. La función escrita a continuación del onRequest contiene el código que de va a ejecutar al inicio de la conexión Ajax. En nuestro ejemplo es un simple alert() de Javascript.

onSuccess:
Esto sirve para especificar un evento, que se ejecutará en caso que la conexión se realice con éxito. A continuación se debe escribir el código de la función que se ejecutará al desatarse el evento, que tiene la particularidad de recibir dos parámetros. El primero es el código HTML del archivo traído por Ajax. El segundo contendrá la respuesta XML del Request. En

nuestro ejemplo la función simplemente muestra en una caja de alert() de Javascript el contenido del código HTML.

onFailure:
Con este otro evento definimos qué hacer en el caso que la conexión con Ajax haya dado un error.

También debemos fijarnos en el último .send(), que es un método que se llama sobre el objeto Request creado. Este método sirve para poner en marcha la conexión Ajax.

Asociar esa conexión a un evento de clic en el enlace

Hasta ahora tenemos un enlace HTML y una instaciación de un objeto de la clase Request. Pero tenemos que conectar estos dos elementos, para que al pulsar el enlace se cree la conexión. Para ello tenemos que añadir un evento al enlace y debemos realizarlo así en Mootools:

```
//función que se va a ejecutar cuando esté listo el dom
window.addEvent('domready', function(){
   //añado un evento al enlace para ejecturar al hacer clic
   $('mienlace').addEvent('click', function(evento){
      //creo y ejecuto un request como se ha visto anteriormente
   });
});
```

El ejemplo sencillo de conexión Ajax con Mootools completo

El código completo del ejemplo es el siguiente.

```
<html>
<head>
   <title>Request de Mootools</title>
   <script src="mootools-1.2-core-yc.js" type="text/javascript"></script>
</head>

<body>
Request de Mootools, un envoltorio de XMLHttpRequest.
<p>
<a href="javascript:void(0)" id="mienlace">Pulsa este enlace</a>

<script>
window.addEvent('domready', function(){
   //función que se va a ejecutar cuando esté listo el dom

   //añado un evento al enlace para ejecturar al hacer clic
   $('mienlace').addEvent('click', function(evento){
      //creo un request
      var prueboRequest = new Request({
         method: 'get',
         url: 'archivo-html-solicitud.html',
         onRequest: function() { alert('Acabas de iniciar una solicitud de contenidos por
Ajax!'); },
         onSuccess: function(texto, xmlrespuesta){ alert('Respuesta:\n' + texto);},
         onFailure: function(){alert('Fallo');}
      }).send();
   });
});
</script>
</body>
</html>
```

Podemos ver el ejemplo en funcionamiento en una página aparte.

Últimas aclaraciones para que este ejemplo Ajax-Mootools funcione

Ahora bien, conviene decir un par de cosas para terminar de aclarar este ejemplo y que cualquiera de vosotros pueda ponerlo en marcha en su ordenador o en su servidor web.

Como se puede haber entendido por las explicaciones anteriores, para la puesta en marcha de este ejemplo necesitamos dos páginas distintas. Una es la que tiene el código fuente del ejercicio y otra página que es la que se conecta por Ajax para mostrar sus contenidos. El archivo con el código Javascript lo podemos llamar de cualquier manera, pero el archivo que nos traemos por Ajax debe llamarse 'archivo-html-solicitud.html', para que la ruta de la conexión realizada en el Request

funcione. Ambos archivos deben estar en el mismo directorio.

Además, para que funcione correctamente la conexión con Ajax debemos poner los archivos en un servidor Web. Si intentamos colocar los archivos en un directorio cualquiera de nuestro disco duro y luego abrir la página con código del ejemplo en un navegador, al intentar hacer la conexión http por Ajax, nos dará un error. Si abrimos directamente el archivo no se podrá realizar la conexión con Ajax. Por lo tanto debemos pasar por un servidor web para que funcione. Para ello debemos de hacer una de estas dos cosas:

1. Podemos tener un servidor web instalado en local en nuestro ordenador. Entonces simplemente colocamos los archivos en un directorio de publicación y accedemos a ellos a través de localhost. Algo como http://localhost/mootools-ajax/request-alert.html
2. Colocar los archivos en un servidor de hosting que tengamos contratado. Para ello los subimos por FTP a nuestro servidor como habitualmente habremos hecho.

Esto es todo. Espero que este primer ejemplo de Ajax con Mootools haya podido entenderse bien. Ahora puedes seguir leyendo y aprendiendo con unas modificaciones a este script, para mostrar los contenidos Ajax en el propio cuerpo de la página.

Artículo por Miguel Angel Alvarez

Ejemplo 2 de Ajax utilizando Mootools

Mejoramos el ejemplo anterior de Ajax con Mootools, para que el contenido traído con Ajax se muestre en una capa div.

Acabamos de ver el primer ejemplo de Ajax con Mootools. Ahora vamos a publicar unas modificaciones bien sencillas, simplemente para hacer que los contenidos traídos por Ajax se muestren en la misma página, pero en el propio cuerpo de la página y no en una caja de alerta. El ejemplo es bien parecido, así que antes de leer esto, debemos haber leído el anterior taller de Ajax con Mootools.

Como decía, la idea es realizar una conexión con Ajax, así que la única diferencia es que, al mostrarse el contenido del archivo, se mostrará en una capa en el mismo contenido de la página., sin recargarla, por supuesto. Así que, de la conexión que hacemos con el objeto de la clase Request, lo único que cambia es lo que va a pasar en caso que la conexión se realice con éxito (Evento onSuccess).

Antes de continuar, si se desea, puede verse en ejemplo en marcha.

Código HTML

El HTML que necesitaremos para el ejemplo debe contener además un contenedor para mostrar la respuesta Ajax.

```
<a href="javascript:void(0)" id="mienlace">Pulsa el enlace</a>
<p>
<div id="cajaactualizacion">
Aquí voy a actualizar el texto!
</div>
```

El enlace no ha cambiado, lo que se puede ver a continuación es que tenemos un DIV llamado "cajaactualizacion", que es donde mostraremos el contenido del archivo.

Código Javascript relativo al Ajax con Mootools

El código es exactamente igual, pero se ha utilizado una función distinta en el evento onSuccess.

```
var nuevoRequest = new Request({
   method: 'get',
   url: 'archivo-html-solicitud.html',
   onRequest: function() { alert('Empezando la solicitud con Ajax!'); },
   onSuccess: function(texto, xmlrespuesta){ $('cajaactualizacion').set('html',texto);},
   onFailure: function(){alert('Error!');}
}).send();
```

Como se decía, lo único que debemos entender, de manera adicional a lo que ya vimos en el ejemplo del artículo anterior, es esto:

```
$('cajaactualizacion').set('html',texto);
```

Simplemente se está haciendo una llamada a un método del elemento $('cajaactualizacion'), que es el DIV donde se iban a escribir los resultados de la conexión Ajax. A este $('cajaactualizacion') le pasamos el método set, pasando a su vez los parámetros "html", para decir que queremos cambiar el innerHTML de la capa, y luego la variable texto, que contiene el texto del archivo traído por Ajax.

El código del segundo ejemplo de Ajax y Mootools

```
<html>
<head>
   <title>Request de Mootools</title>
   <script src="mootools-1.2-core-yc.js" type="text/javascript"></script>
</head>

<body>
Clase Request de Mootools, un envoltorio de XMLHttpRequest. Para Ajax!
<p>
<a href="javascript:void(0)" id="mienlace">Pulsa este enlace</a>
<p>
<div id="cajaactualizacion">
Aquí voy a actualizar el texto!
</div>

<script>
window.addEvent('domready', function(){
   //función a ejecutar cuando esté listo el dom

   $('mienlace').addEvent('click', function(evento){
      var nuevoRequest = new Request({
         method: 'get',
         url: 'archivo-html-solicitud.html',
         onRequest: function() { alert('Empezando esta solicitud Ajax!'); },
         onSuccess: function(texto, xmlrespuesta){ $('cajaactualizacion').set('html',texto);},
         onFailure: function(){alert('Error!');}
      }).send();

   });

});
</script>
</body>
</html>
```

Podemos ver en marcha el ejemplo en una página aparte.

Atentos a las otras explicaciones y recomendaciones adicionales tratadas en el artículo anterior!, que hay cosas que no hemos comentado aquí por haberse dado por entendidas.

Artículo por Miguel Angel Alvarez

Vídeo: Ajax con Mootools, ejemplos sencillos

Vídeo en el que mostraremos dos ejemplos muy sencillos de Ajax con el framework Javascript Mootools.

Ajax es muy simple si utilizamos un framework Javascript. En este vídeo de corta duración veremos cómo realizar un pequeño script, ayudados por Mootools, que nos servirá para hacer una conexión Ajax para traernos un contenido.

El vídeo muestra el proceso desde el inicio, que sería la descarga del framework Javascript y la inclusión en una página web. Luego se hace un script con Javascript que, haciendo uso de las librerías de Mootools, se conecta con el servidor para traerse un contenido sencillo. En concreto vamos a hacer dos ejemplos, uno extremadamente sencillo y otro un poco más

complejo, que hace uso de algunas características de las conexiones Ajax que nos permite Mootools.

En el ejemplo tendremos una capa y un enlace. Al pulsar el enlace se conectará por Ajax para traerse un contenido que está en un fichero aparte. En el primero obtendremos el archivo tal cual y en el segundo vamos a utilizar un tipo de conexión más elaborada que nos permitirá enviar un dato por POST a la página que se solicitará por Ajax.

Ambos ejemplos espero que sirvan para ilustrar el modo con el que se puede hacer una solicitud Ajax. Aunque están explicados desde cero, será interesante que el lector sepa lo que es Mootools o cómo utilizarlo para hacer Ajax. Todo esto está explicado en DesarrolloWeb.com en el Manual de Mootools y en los artículs de ejemplos por Ajax en Mootools.

Os dejamos con el vídeo de una duración de 10 minutos, para poder subirlo a YouTube. Espero que las explicaciones no sean demasiado rápidas para que se pueda entender bien. (mirar en la web)

Código de los ejemplos

Dejo por aquí el código de estos dos ejemplos, por si alguien lo quiere copiar y pegar para probar todo esto.

Ejemplo1:

```
<html>
<head>
   <title>ej1</title>
   <script src="mootools-1.2.3-core-yc.js" type="text/javascript"></script>
   <script>
   window.addEvent("domready", function(){
      //para ejecutar cuando está lista la página
      $("enlaceajax").addEvent("click", function(evento){
         evento.stop();
         $("capaajax").load("contenido-ajax.html");
      });
   });
   </script>
</head>

<body>


<div id="capaajax"></div>
<br>
<br>
<a href="#" id="enlaceajax">Enlace para cargar contenido Ajax</a>
</body>
</html>
```

El ejemplo lo hemos publicado en DesarrolloWeb.com en una página aparte.

Ejemplo2:

```
<html>
<head>
   <title>ej2</title>
   <script src="mootools-1.2.3-core-yc.js" type="text/javascript"></script>
   <script>
   window.addEvent("domready", function(){
      //para ejecutar cuando está lista la página
      $("enlaceajax").addEvent("click", function(evento){
         evento.stop();
         //$("capaajax").load("contenido-ajax.html");
         var miAjax = new Request.HTML({
            url: "otro-ajax.php",
            method: "post",
            data: "cosa=valor",
            update: $("capaajax")
         });
         miAjax.send();
      });
```

```
   });
   </script>
</head>

<body>


<div id="capaajax"></div>
<br>
<br>
<a href="#" id="enlaceajax">Enlace para cargar contenido Ajax</a>
</body>
</html>
```

Puede verse en marcha.

Para el que lo desee, he colocado un ZIP con el código de los ejemplos Ajax.

Esto debe colocarse en un servidor web

Ya lo mencionamos en el vídeo: las conexiones por Ajax se realizar por HTTP, por lo que necesitamos publicar el ejemplo en un servidor web para que funcione.

Podemos colocarlo en local, en un servidor que podamos tener instalado en nuestro ordenador, o en remoto, en un espacio de alojamiento web.

Si queréis instalar un servidor en vuestro ordenador os recomiendo probar con Wamp

. Como os digo, cualquier espacio de hosting, también os servirá para probar estos archivos, que tendríais que subir por FTP al servidor.

Mencionar además que el segundo ejemplo envía datos por POST y nosotros los recibimos a través de un archivo PHP. Por ello, os vendrá bien tener un servidor que soporte PHP, para que el segundo ejercicio pueda mostrat el contenido que se envía por POST.

Espero que estas prácticas de Ajax con Mootools sirvan de utilidad para los lectores de DesarrolloWeb.com

Artículo por Miguel Angel Alvarez

Ajax en Prototype

Tutorial en el que mostraremos cómo hacer solicitudes Ajax utilizando el framework Javascript Prototype, que tiene varias clases interesantes para hacer aplicaciones Ajax rápidamente.

A estas alturas probablemente todo el mundo debe conocer lo que es Ajax, o al menos las personas que leen este artículo asumimos que lo saben y que desean realizar sus propias páginas haciendo uso de Ajax de una manera sencilla. Los Frameworks Javascript nos vienen como anillo al dedo para simplificar nuestras tareas de desarrollo y en concreto Prototype ofrece diversas utilidades que nos permitirán construir scripts Ajax en pocos minutos.

Lo más complicado de hacer una aplicación web Ajax es que cada navegador tiene sus particularidades a la hora de hacer solicitudes HTTP desde Javascript para traer contenidos del servidor que se pueden mostrar sin actualizar la página entera. La tarea de hacer un script Cross-Browser es sin duda larga y tediosa, por ello Ajax permaneció por un tiempo a disposición únicamente de programadores avanzados. Sin embargo, con la llegada de los frameworks Javascript, cualquier persona con un conocimiento medio de este lenguaje, puede encarar sus objetivos en pocos minutos y sin complicaciones derivadas de las distintas plataformas.

En este artículo vamos a dedicar un tiempo a mostrar el proceso de creación de scripts sencillos en Javascript que hacen solicitudes Ajax, utilizando Prototype, el popular framework Javascript.

Nota: en DesarrolloWeb.com ya explicamos cómo hacer scripts Ajax con otros frameworks, como pueden ser Mootools o jQuery. Si conocemos otros frameworks Javascript, veremos que con Prototype las cosas son bastante parecidas.

Descargar Prototype e incluir el script en nuestra página

Como un paso previo a realizar cualquier cosa con Prototype, tenemos que descargar el framework e incluirlo en la página.

Podemos hacer el download de la versión más reciente de Prototype en su página web: http://www.prototypejs.org

Tendremos que obtener un archivo con extensión js, que contiene todo el código de las librerías de Prototype. Una vez descargado lo podremos incluir en nuestra página con una etiqueta SCRIPT que colocaremos en la cabecera de la página:

```
<script src="prototype-1.6.0.3.js" type="text/javascript"></script>
```

Clase Ajax y Ajax.Updater de Prototype

En Prototype existe una clase llamada **Ajax** que contiene todo lo que necesitaremos para hacer llamadas asíncronas al servidor. Realmente esta clase Ajax no se puede utilizar por si misma, por ser abastracta, sino que tenemos que usarla a través de distintas clases que heredan de ella, con las que podremos hacer cualquier tipo de script según nuestras necesidades. Lo bueno es que todo lo que hagamos funcionará en los navegadores más comunes de la misma manera, con lo que no tendremos que complicarnos en hacer un código diferente para cada tipo de plataforma de cliente. Además, podremos ahorrar muchas líneas de código al hacer las solicitudes y otras acciones en torno de éstas.

Esta clase Ajax tendría una clase heredada con la que podemos hacer un ejemplo de uso bien simple. Se trata de la clase Ajax.Updater, con el que hacer una solicitud por Ajax y actualizar a la vez el contenido HTML de cualquier elemento de la página. Veamos un ejemplo:

```
new Ajax.Updater('contenidoajax', 'recibir-x-ajax.html');
```

Con esta única línea de código hacemos todo el trabajo, de recibir un archivo por Ajax y mostrar los contenidos en un lugar de la página. En concreto recibiremos el archivo "recibir-x-ajax.html" y actualizaremos el elemento con identificador (atributo id del HTML) "contenidoajax".

Ahora veamos un código de una página que realizaría una simple solicitud Ajax:

```
<html>
<head>
   <title>Recibir contenido por Ajax</title>
   <script src="../prototype-1.6.0.3.js" type="text/javascript"></script>
<script>
function recibirAjaxActualizar(){
   new Ajax.Updater('contenidoajax', 'recibir-x-ajax.html');
}
</script>
</head>

<body>
<div id="contenidoajax">Elemento a actualizar con la solicitud Ajax</div>
<p>
- <a href="javascript: recibirAjaxActualizar();">Pulsame para recibir un contenido con Ajax y
actualizar un elemento de la página</a>
</body>
</html>
```

Podemos ver este código en marcha en una página aparte.

Nota: la solicitud Ajax viaja por el protocolo HTTP, por lo que este ejemplo debe estar alojado en un servidor web para que funcione. Si colocas el archivo tal cual en tu disco duro y lo ejecutas no funcionará, puesto que tienes que solicitarlo a través de cualquier servidor web, ya sea uno que puedas tener instalado en tu ordenador o bien un espacio de alojamiento web del que dispongas en Internet.

En el siguiente artículo veremos otros casos de uso de Ajax con Prototype que complementarán las presentes explicaciones.

Artículo por Miguel Angel Alvarez

Crear scripts Ajax personalizados en Prototype

Algunas guías adicionales para hacer scripts con Ajax con el framework Javascript Prototype.

En el pasado artículo de DesarrolloWeb.com vimos cómo construir scripts Ajax extremadamente sencillos con Prototype. A continuación ampliaremos las explicaciones, para enseñar a personalizar un poco más las solicitudes Ajax y tus scripts, para hacer cosas que seguramente necesitarás para mejorar la experiencia de usuario en tus páginas web.

Para comenzar, estaría bien comentar que la clase "padre" de Prototype para realizar conexiones HTTP asíncronas se llama **Ajax**, pero que esta clase es abstracta y para utilizarla debemos trabajar con alguna de las clases que heredan de esta. La clase Ajax general permite algunos tipos de personalización de la solicitud muy útiles, los cuales sirven también para las solicitudes Ajax que podemos hacer cuando utilizamos las clases heredadas, como Ajax.Updater o Ajax.Request. Por lo que la mayoría de las explicaciones que veremos sirven para cualquier tipo de conexión Ajax en Prototype.

En el ejemplo del artículo anterior vimos un caso de uso de Ajax.Updater y ahora vamos a mostrar cómo utilizar otra clase más básica llamada Ajax.Request, que es exactamente igual, salvo que ésta no actualiza la información de la respuesta automáticamente en la página.

Por ejemplo veamos el siguiente código de uso de la clase Ajax.Request:

```
new Ajax.Request('recibir-parametros.php',{
   method: 'get',
   parameters: {miparametro: 'El valor!', otro: 99},

   onSuccess: function(respuesta){
      alert(respuesta.responseText);
   }
});
```

Como se puede ver, el constructor de la clase Ajax.Request recibe como primer parámetro la URL que se va a solicitar con Ajax y como segundo parámetro un objeto con todas las propiedades y funciones para personalizar la solicitud.

En este caso hemos especificado diversos valores como:

- method: con un valor "get" / "post", que es para indicar cómo se van a enviar los datos a la URL, por método get o post.
- parameters: con una lista de los parámetros que queremos enviar con la solicitud Ajax, en notación de objeto.
- onSuccess: que es una función con lo que queremos hacer cuando la solicitud se haya procesado con éxito. En este caso simplemente mostramos en una caja de alerta la respuesta recibida por la solicitud. Este método onSuccess es un evento que soporta la clase Ajax y que se ejecuta cuando la solicitud se ha producido y se ha recibido la respuesta correctamente.

Podemos ver un script que hace uso de esta clase para hacer una solicitud personalizada al servidor por medio de Ajax:

```
<html>
<head>
   <title>Recibir contenido por Ajax</title>
   <script src="../prototype-1.6.0.3.js" type="text/javascript"></script>
<script>
function recibirAjax(){
   new Ajax.Request('recibir-parametros.php',{
      method: 'get',
      parameters: {miparametro: 'El valor!', otro: 99},

      onSuccess: function(respuesta){
         alert(respuesta.responseText);
      }
   });
}
```

```
</script>
</head>

<body>
- <a href="javascript: recibirAjax();">Pulsame para recibir un contenido con Ajax</a>
</body>
</html>
```

También necesitaremos crear una página, a la que llamamos por Ajax, que recibe los parámetros enviados en la solicitud. En este caso a página es PHP, pero podríamos utilizar cualquier otra tecnología de programación del lado del servidor.

El código de esa página sería parecido a esto:

```
<?
echo "La respuesta!. He recibido: " . $_GET["miparametro"] . " / " . $_GET["otro"];
?>
```

Ahora podemos ver esta página en marcha.

Personalizar el script Ajax con el típico mensaje de carga

Ahora veamos cómo realizar una solicitud Ajax en el que nos muestre el típico mensaje de "Cargando...", para avisar al usuario que se ha producido una solicitud y que tiene que esperar a recibir la respuesta.

Para ello simplemente tenemos que modificar un poco el script anterior, para crear el comportamiento que muestre el mensaje de carga. Por variar un poco, en este caso utilizaremos Ajax.Updater, de una manera muy similar.

```
$('cargando').update("Cargando...");
new Ajax.Updater('contenidoajax', 'recibir-parametros.php', {
   method: 'get',
   parameters: {miparametro: 'lo que sea', otro: 58},

   onSuccess: function(){
      $('cargando').update("");
   }
});
```

En este caso hemos creado una primera línea de código, justo antes de la solicitud Ajax, que actualiza el texto del elemento con identificador "cargando". En ese elemento colocamos el mensaje de carga. Nosotros hemos actualizado el contenido colocando un texto, pero podríamos haber colocado una imagen, ya que update() acepta cualquier código HTML.

Luego se codifica la solicitud Ajax, en la que hemos colocado en el evento onSuccess una línea de código para eliminar el mensaje de carga una vez hemos recibido una respuesta correcta del servidor, utilizando el mismo método update() sobre el elemento de la página deseado.

A continuación se puede ver el código completo de este ejemplo:

```
<html>
<head>
   <title>Recibir contenido por Ajax</title>
   <script src="../prototype-1.6.0.3.js" type="text/javascript"></script>
<script>
function recibirAjax(){
   $('cargando').update("Cargando...");
   new Ajax.Updater('contenidoajax', 'recibir-parametros.php', {
      method: 'get',
      parameters: {miparametro: 'lo que sea', otro: 58},

      onSuccess: function(){
         $('cargando').update("");
      }
   });
}
</script>
</head>

<body>

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

```
<p>
- <a href="javascript: recibirAjax();">Pulsame para recibir un contenido con Ajax</a>
<p>
<div id="cargando"></div>
</body>
</html>
```

Si se desea, podemos poner en marcha el ejemplo en una página aparte.

Con esto hemos visto lo sencillo que es hacer Ajax con Prototype. Pero sin duda, lo más útil e interesante es que con Prototype podemos construir scripts Ajax que son compatibles con todos los navegadores, sin necesidad de preocuparnos por las diferentes particularidades de cada uno.

Artículo por Miguel Angel Alvarez

Detección de JavaScript con Ajax.net

En este articulo mostramos como podemos detectar si el cliente web esta ejecutando Javascript, lo que nos ayuda a hacer webs accesibles y plenamente funcionales.

A la hora de desarrollar páginas web avanzadas, muchos de nosotros nos hemos hemos tenido que enfrentar al problema de la accesibilidad. Casi siempre, en ese camino, nos hemos encontrado con la disyuntiva de la usabilidad frente a la accesibilidad. Inevitablemente nos hacemos esta pregunta ¿Hacer páginas web accesibles está reñido con la usabilidad?

Quizás la anterior pregunta sería muy larga de analizar y posiblemente cada uno de nosotros, según nuestras propias experiencias, tengamos una respuesta. Sin embargo, en este artículo vamos a centrarnos en una cuestión un tanto más específica, pero muy relacionada con la anterior: ¿Qué puedo hacer para continuar usando Javascript y sin embargo la web sea accesible?

La realidad, a la hora de hacer un desarrollo web , nos lleva a entender que hacer un site accesible no es complicado, siempre que se sigan unas normas estrictas de diseño e implementación. Pero, cuando además se quiere dotar al proyecto de una buena experiencia de usuario, la cosa cambia.

Hasta ahora hemos visto equipos de desarrollo que realizan dos portales , uno accesible y otro no accesible, de tal manera que según se activa o desactiva el Javascript la web va redirigiendo a una versión u a otra. Para nosotros esta solución tiene una gran desventaja, además de la necesidad de duplicar nuestro volumen de trabajo, pues dicha alternativa conlleva el mantenimiento de dos portales en paralelo, lo que puede crear incongruencias. Aunque tiene la ventaja de que la página es plenamente funcional y es plenamente accesible.

La otra solución es la de poner etiquetas noscript y dotar de esa funcionalidad a la web…

La solución que hoy proponemos trata de unir las etiquetas noscript con una nueva técnica que combina el Ajax y métodos de página, para poder detectar si está o no activo el motor Javascript en el navegador del usuario.

En lo primero que podemos pensar es en utilizar el objeto Request.Browser.JavaScript, sin embargo, esto trae problemas ya que solo dice si el navegador lo admite , pero no nos dice si está activo (además de que ese objeto está en desuso y acceder a él nos arrojará un bonito Warning).

El código (en c#) de la solución propuesta es el siguiente:

```
public static bool JavascriptActivo { get; set; }
#region Eventos del formulario
protected void Page_Load(object sender, EventArgs e)
{
JavascriptActivo = false;

}

#endregion

[WebMethod]
public static void TieneJavascript()
```

```
{
JavascriptActivo = true;
}
```

Por su parte, en el aspx tendríamos que utilizar el siguiente código:

```
<asp:ScriptManager ID="ScriptManager1" runat="server" EnablePageMethods="True">
</asp:ScriptManager>

<script language="javascript" type="text/javascript">
PageMethods.TieneJavascript();
</script>
```

Con esto le estamos diciendo que queremos utilizar el método de página "TieneJavaScript", el cual activa la propiedad estática que define si utiliza Javascript. Y por último, en cada postback le decimos que la ponga a false (que no utiliza Javascript). En realidad, el poner esta propiedad a false es para detectar si mantiene activo el Javascript.

La explicación es sencilla, según el ciclo de vida primero se ejecuta el onload, y luego se renderiza la página, es entonces cuando se ejecuta el Javascript, que está funcionando por Ajax, y por lo tanto no realiza refresco de la página.

Si utiliza Javascript se hace la llamada al método de página y todo transcurre como se espera, si no, simplemente se actualiza la variable y se muestra la página sin Javascript.

Conclusión:

El 28 de diciembre del 2007 se publicó la Ley de Medidas de Impulso de la Sociedad de la Información. En la misma se obliga a los entes públicos a la eliminación de barreras de accesibilidad de sus sitios. Esta es la razón principal por la que todo desarrollador, debe aprender las reglas para realizar sus desarrollos web de forma accesible. Entre estas normas destacamos la imposibilidad de ejecución de Javascript, o mejor dicho, la necesidad de crear una opción mediante la cual se pueda hacer lo mismo que se implementa con Javascript, pero con programación a través del servidor.

Hasta ahora las medidas que se han tomado, pasaban por la duplicidad de código, redirigiendo a páginas accesibles y dejando el portal no accesible a disposición de usuarios sin limitaciones.

En nuestra solución damos un remedio a los inconvenientes de la situación anterior, ya que al ejecutar el script a descargase la página detectamos en las primeras fases si se esta ejecutando o no, y en sucesivos postback estaremos seguros de cual es la situación actual de navegador.

Si a eso le sumamos que por defecto el estado del Javascript del navegador cliente será estable nos da un maravilloso juego de probar una vez (en la página de inicio) y guardar dicho estado en la session de usuario.

Todo esto nos lleva a una amplia gama de posibilidades, pero siempre con la ventaja de no tener que duplicar el códigoS

Artículo por Fernando G. Muñoz