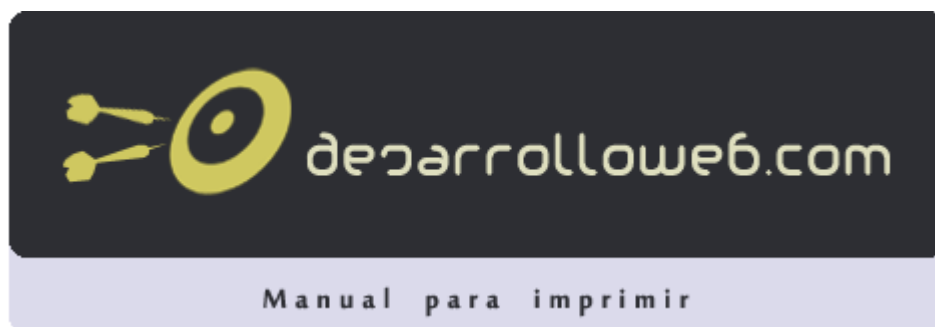


# Manual Ajax práctico

## Taller 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)

**Damián Suárez**  
Impulsor de XiFOX.net  
<http://www.cabzaderaton.com.ar>  
(1 capítulo)

**Andrés Fernández**  
<http://www.disenocentell.com.ar>  
(1 capítulo)

## Enviar mediante POST y GET usando una sola funcion AJAX

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.encodeURIComponent = 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'> Espere 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**

El código tiene dos funciones.

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

La función recibe 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 función.

### **DESGLOSANDO LA FUNCION**

#### **¿Envío por GET o por POST?**

Primeramente mediante el siguiente código

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

```
} else {
```

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

```
}
```

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

Adicionalmente sucede que si envías 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 puedes 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 función

Mediante el siguiente condicional, se complementa el primer condicional, enviando los datos de la solicitud mediante POST o GET según corresponda, con el código 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**

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.campoparalavariabilepostenviada1.value+'  
&variablepostenviada2='+Miformulario.campoparalavariabilepostenviada2.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**

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 los 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 incluidos 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**

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: Título 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 título 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 al 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 imagen
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 llevo al server. Estas líneas mas adelante no las utilizaremos; ahora sólo las usamos para realizar el seguimiento del script. Asignamos a `$tipo` el tipo de archivo para poder controlar que sea una imagen. Se cortan los primeros 5 caracteres y si todo es correcto `$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 ocultar 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 imagen
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 variables 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
Imágenes</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 servidor' + 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 línea 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 scripts 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 guía para 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](#)*