Tutorial Básico de Ajax

Ajax no es un lenguaje exactamente su nombre viene dado por el acrónimo de Asynchronous JavaScript And XML y es posiblemente la mayor novedad en cuanto a programación web en estos últimos años.

El corazón de **Ajax** es el objeto XMLHttpRequest que nos permite realizar una conexión al servidor y al enviarle una petición y recibir la respuesta que procesaremos en nuestro código Javascript, estamos hablando del verdadero motor de **Ajax**, por ejemplo gracias a este objeto podemos desde una página HTML leer datos de una web o enviar datos de un formulario sin necesidad de recargar la página.
Puedes programar numerosas nuevas aplicaciones enfocadas desde una visión distinta como es el caso de este **paginador AJAX**

1. Basado en los estándares abiertos
2. Usabilidad
3. Válido en cualquier plataforma y navegador
4. Beneficia las aplicaciones web
5. No es dificil su utilización
6. Compatible con Flash
7. Adoptado por los "gordos" de la tecnología web
8. Web 2.0
9. Es independiente del tipo de tecnología de servidor que se utilice
10. Mejora la estetica de la web

La manera más fácil para comprender realmente la funcionalidad de Ajax es ver cómo funciona una aplicación web con Ajax y cómo una sin Ajax.

Sin Ajax

Se crearía una página con un formulario, cuando el usuario envia los datos del formulario se produce una conexión a la base de datos y se muestra por pantalla la página que el servidor devuelve, todo esto hace que se recargue la página ya sea saltando a una diferente o a ella misma, el usuario debe esperar una nueva carga de página despues de cada envío.

Es lento porque debe descargar la información **HTML** por duplicado.

Con Ajax

Utilizariamos un código Javascript que que crearía el mencionado objeto XMLHttpRequest al enviar el formulario, esta llamada se produce de forma asincrona lo que significa que se envían los datos y no se recarga la página, una vez el servidor responde una función Javascript es la que valora la respuesta del servidor, si esta respuesta es la deseada imprimeremos el texto que indique al usuario que sus datos fueron enviados correctamente.

El navegador no recarga la página, la experiencia desde el punto de vista del usuario es muy satisfactoria puesto que se asemeja a la respuesta del típico software de escritorio, ya no te planteas enlazar páginas sino enviar y recibir datos en una misma página que mediante funciones evalua las diferentes respuestas.

Es bastante más rápido puesto que no tiene que descargar de nuevo el código **HTML** de la página de confirmación del formulario.

Ejemplo objeto XMLHttpRequest (AJAX)

En primer lugar crearemos nuestro objeto ActiveX en IExplorer y un objeto nativo en el resto de navegadores que lo soportan, y es por ello que tendremos que ver qué objeto creamos, controlandolo con diferentes condiciones, con esto conseguimos que el navegador cree una instancia del objeto apropiado, dependiendo del navegador usado por el usuario.

```
function nuevoAjax(){
var xmlhttp=false;
try {
xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
} catch (e) {
try {
xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
} catch (E) {
xmlhttp = false;
}
}
if (!xmlhttp && typeof XMLHttpRequest!='undefined') {
xmlhttp = new XMLHttpRequest();
}
return xmlhttp;
}
```

Una vez hayamos comprendido y realizado este paso tendremos que tener en cuenta los métodos y propiedades que nos ofrece **AJAX**:

Métodos

abort() – Detiene la petición en curso.
getAllResponseHeaders() – Devuelve todas las cabeceras de la respuesta (etiquetas y valores) como una cadena.
getResponseHeader(etiqueta) – Devuelve el valor de la etiqueta en las cabecerasde la respuesta.
open(método,URL,asíncrona,nombre,password) – Abre una conexión con esa URL mediante ese metodo (GET o POST), aplicando los valores opcionales de la derecha (ahora se explicará).
send(contenido) – Envía el contenido al servidor.
setRequestHeader(etiqueta,valor) – Establece el valor de una etiqueta de las cabeceras de petición.

De está lista nos detendremos en el método open que es uno de los más utilizados y el que nos permitirá utilizar la mejor característica de Ajax que es la carga de datos externos a la página sin necesidad de recargar la misma.

Método Open

El método open prepara una conexión HTTP a través del objeto XMLHttpRequest con un método y una URL especificados.

```
XMLHttpRequest.open ( sMetodo, sURL [, bSincronia [, sUsuario [, sPwd ] ] ] );
```

sMetodo es la cadena que nos indicara el tipo de conexión (GET o POST)
sURL es la url a la que realizamos la petición
bSincronia es un campo booleano con el que podemos utilizar modo asíncrono o síncrono, si lo fijamos en "false" modo síncrono perderiamos las mejores características de **AJAX**, los datos sUsuario y sPwd son opcionales y sólo aplicables en caso de caída del servidor.

Al llamar a open el atributo readyState a 1, resetea los headers de envío y los devuelve atributos responseText, responseXML, status y statusText a sus valores iniciales

Propiedades

onreadystatechange – Contiene el nombre de la función que se ejecuta cada vez que el estado de la conexión cambie.
readyState – Estado de la conexión, puede valer desde 0 (no iniciada) hasta 4 (completado).
responseText – Datos devueltos por el servidor en formato cadena.
responseXML – Datos devueltos por el servidor en forma de documento XML que puede ser recorrido mediante las funciones del DOM (getEementsByTagName, etc).
status – Código enviado por el servidor, del tipo 404 (documento no encotrado) o 200 (OK).
statusText – Mensaje de texto enviado por el servidor junto al código (status), para el caso de código 200 contendrá "OK".

Conocer estas propiedades y métodos es algo muy útil a la hora de desarrollar aplicaciones utilizando Ajax debido a la gran ayuda que muchas de ellas ofrecen a la hora de depurar errores. Y nos da una mayor idea acerca de la potencia de esta conjunción de tecnologías.

Ejemplo de envío de datos "GET"

```
function cargarContenido(){
var t1, t2, contenedor;
contenedor = document.getElementById('contenedor');
t1 = document.getElementById('texto1').value;
t2 = document.getElementById('texto2').value;
ajax=nuevoAjax();
ajax.open("GET", "ejemploajax2.php?t1="+t1+"&t2="+t2,true);
ajax.onreadystatechange=function() {
if (ajax.readyState==4) {
contenedor.innerHTML = ajax.responseText
}
}
ajax.send(null)
}
```

Ejemplo de envío de datos "POST"

Unicamente debemos cambiar algunas cosas en nuestra función:
Adicción de una línea adicional: setRequestHeader que especifica qué tipo de datos llegarán al servidor. Cambio del parametro que especifica **el método a "POST"** y por último utilizaremos parametros para el "send".

```
function cargarContenido(){
var t1, t2, contenedor;
contenedor = document.getElementById('contenedor');
t1 = document.getElementById('texto1').value;
```

```
t2 = document.getElementById('texto2').value;
ajax=nuevoAjax();
ajax.open("POST", "ejemploajax2.php",true);
ajax.onreadystatechange=function() {
if (ajax.readyState==4) {
contenedor.innerHTML = ajax.responseText
}
}
ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
ajax.send("t1="+t1+"&t2="+t2)
}
```

Diferencia entre el método GET y POST

Es aconsejable elegir "GET" para aquellas peticiones en las que se soliciten pocos datos y "POST" para aquellas en las que sea necesario enviar información, especialmente si estos datos podrían superar los 512 bytes en total, puesto que por el método "GET" no podremos recibir la totalidad de los datos.