

The page features a decorative design with three blue circles of varying sizes, each composed of concentric layers of different shades of blue. These circles are connected by thin, light blue lines that form a triangular shape. The circles are positioned in the upper and lower right areas of the page, while the text is on the left.

Manual de ajax en español By "ajaxman"

Usando ajax mediante el método
get para realizar peticiones de
manera transparente

Ajax y el objeto XMLHttpRequest

Escrito por Javier
16/07/2007

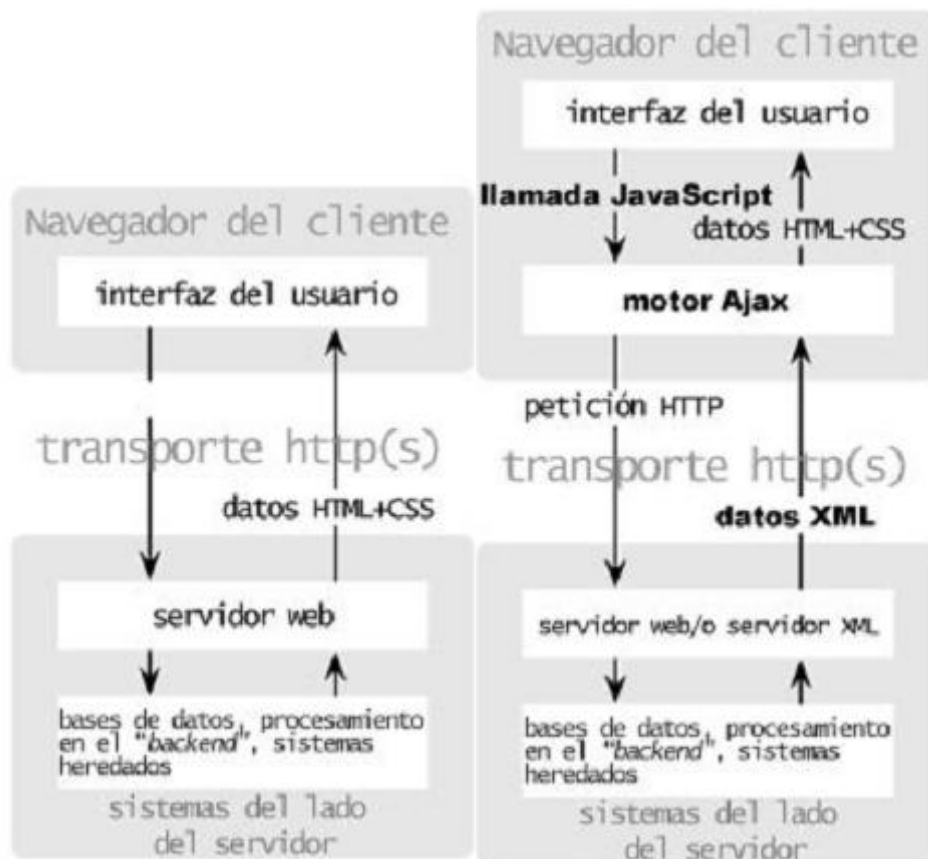
Ajax.

El 18 de Febrero de 2005, Jesse James Garrett, director de Estrategia para la experiencia del usuario y fundador de Adaptive Path, publico un artículo al cual se refería a ajax como:

Ajax: Un Nuevo acercamiento a las Aplicaciones Web

Exactamente se refería a una nueva forma de ver las aplicaciones web, ya no como sitios dinámicos, si no además como sitios interactivos, como la posibilidad de cargar paginas dinámicamente, sin tener que recargar toda la pagina (como un iframe), que es uno de los logros fundamentales de ajax.

Algo que debe quedar claro es que ajax no es una tecnología, un modulo de apache, un script, una extensión, si no es más bien una mezcla de tecnologías, que permiten crear aplicaciones web, increíbles y dinámicas.



Modelo Clásico Modelo Ajax

Ajax viene de la palabra Asynchronous JavaScript And XML, aunque también se podría referirse uno a este término como AJAH, AJAP, AJAJ, AJAA, Etc.

Traduciéndolo al español podremos decir que se trata de una mezcla de tecnologías:

- Uso de estándares XHTML y CSS
- Uso del Document Object Model(DOM)
- Interacción de datos usando XML, XSLT, HTML, hasta JS y Páginas Dinámicas (ASP, PHP, CGI, PYTHON, NET, Etc.).
- Y lo indispensable JAVASCRIPT, para juntar y organizar todo.

Ahora bien vayamos por partes.

¿Cómo demonios se usa ajax?

Para poder empezar a escribir páginas web con ajax, deberías cumplir con algunos requisitos básicos:

- Saber usar HTML
- CSS (que no es tan indispensable pero necesario)
- Javascript, básico
- y de preferencia el saber manejar mínimamente algún lenguaje de programación web como PHP, .NET, PYTHON, Etc.

Veremos a continuación como se crea una aplicación usando el modelo Ajax.

Primero necesitaremos hacer una página web que puede estar en un servidor local o remoto, esta la haremos en html.

Pagina_dinamica.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<title>Mi primer Script con Ajax</title>
<script language="JavaScript" type="text/javascript">
/**/
function _Ajax()
{</pre></div>
```

```
}
/*]]>*/
</script>
</head>
<body>
<form action="javascript:_Ajax();" method="get">
<input name="nombre" id="nombre" type="text"/>
<input type="button" name="aceptar" value="Aceptar"
onclick="_Ajax();" />
</form>
<div id="resultado">
</div>
</body>
</html>
```

Ahora vamos por partes, primero necesitamos agregarle a nuestra página web, un formulario con un "action" a "javascript", esto es necesario porque, cuando alguien le da "enter" y es el ultimo o único campo del formulario este se envía a la pagina que este, indicada en el action, por eso si, el action es un javascript, el formulario se enviara a la función.

```
<form action="javascript:_Ajax();" method="get">
```

Después, creamos un campo de tipo texto, con un id, recordemos que para poder implementar ajax, hay que saber identificar, la estructura básica del DOM.

```
<input name="nombre" id="nombre" type="text"/>
```

Y por ultimo un input, de tipo botón.

```
<input type="button" name="aceptar" value="Aceptar"
onclick="_Ajax();" />
```

Al cual, no es necesario agregarle un id, y le asignamos, un evento "onclick", con el cual llamamos a una función, en este caso no es necesario agregarle parámetros, pues no los necesita.

Recordemos que para el nombre de la función no es necesario que se llame _ajax, se puede llamar de cualquier forma, sin embargo es mejor nombrarla, con algún nombre que caracterice su funcionamiento.

Generamos la función que de momento, no tendrá mas contenido que este:

```
function _Ajax()  
{  
  
}
```

Para poder hacer peticiones remotas, es necesario del uso de un objeto, el XMLHttpRequest, que a pesar de ser un estándar no está bien implementado en algunos navegadores como, Internet Explorer, por ello es necesario hacer una función que genere correctamente este objeto.

```
function XMLHttpRequest(){  
var Object;  
if (typeof XMLHttpRequest == "undefined" )  
{  
if(navigator.userAgent.indexOf("MSIE 5") >= 0)  
{ Object= new ActiveXObject("Microsoft.XMLHTTP");}  
else  
{ Object=new ActiveXObject("Msxml2.XMLHTTP");}  
}  
else  
{ Object=new XMLHttpRequest();}  
return Object;  
}
```

y después lo instanciamos, de la siguiente manera,

```
var ajax=XMLHttpRequest();
```

Pero ¿Que quiere decir el código que acabamos de escribir?.

Vayamos por partes, el corazón de una petición remota, en este caso de ajax, se utiliza el objeto XMLHttpRequest, sin embargo, este objeto solo está definido en navegadores basados en netscape, como Mozilla, Firefox, Camino, Minimo(Navegador de Mozilla para Celular).

Y en navegadores, con motor de Internet Explorer, no está implementado este objeto nativamente, por ello es necesario crear una función que revise el tipo de navegador, y en base a ello nos dé el objeto correcto, de lo contrario, crear una nueva instancia de un objeto XMLHttpRequest sería tan fácil como:

```
var ajax=new XMLHttpRequest();
```

Ahora veamos línea por línea lo que realiza nuestro script.

```
function XMLHttpRequest(){ //Con esto creamos la función XMLHttpRequest(no es necesario que se llame asi)
```

```
var Object; //Agregamos la variable llamada objeto
if (typeof XMLHttpRequest == "undefined" ) //revisamos si no esta
definido el objeto nativamente(navegadores tipo mozilla)

{if(navigator.userAgent.indexOf("MSIE 5") >= 0) //Ahora revisamos si el
motor es mayor o igual a MSIE 5.0 (mayor que microsoft internet explorer
5.0)

{ Object= new ActiveXObject("Microsoft.XMLHTTP");} // Si es así creamos
un control activeX a partir de un objeto
ActiveXObject("Microsoft.XMLHTTP")

else //si no , o si es menor a MSIE 5.0 creamos otro control activeX

{ Object=new ActiveXObject("Msxml2.XMLHTTP");} // a partir de un
objeto ActiveXObject("Msxml2.XMLHTTP")

}
else // en cambio si el objeto estaba definido nativamente, solo lo
instanciamos

{ Object=new XMLHttpRequest();} //Instancia del objeto XMLHttpRequest

return Object; // Y retornamos el objeto creado

}
```

Y después solo lo instanciamos una variable a partir de la función como mencione al final de el código de la función.

También podemos definirlo mediante operadores ternarios, y minimizar el código el cual quedaría de la siguiente forma:

```
if ( typeof XMLHttpRequest == "undefined" )
XMLHttpRequest = function(){
return new ActiveXObject(
navigator.userAgent.indexOf("MSIE 5") >= 0 ?
"Microsoft.XMLHTTP" : "Msxml2.XMLHTTP"
);
};
```

y para instanciarlo seria de la siguiente manera:

```
var ajax=new XMLHttpRequest();
```

Ahora creamos la función que se encargara de hacer la petición a la otra página web:

```
function _Ajax()
{
var nombre=document.getElementById('nombre').value;
ajax.open("GET","datos_get.php?nombre="+nombre,true);
ajax.onreadystatechange=function(){
if(ajax.readyState==4)
{
var respuesta=ajax.responseText;
document.getElementById('resultado').innerHTML=respuesta;
}
}
ajax.send(null);
}
```

Veamos los puntos importantes de la función anterior:

El nombre de la función:

```
function _Ajax()
```

Obtenemos el valor del campo, el cual esta referenciado por un id, esta es una de las formas mas rápidas para acceder a un elemento en especifico, el cual puede ser un div, una celda, un campo de texto, etc...;

```
var nombre=document.getElementById('nombre').value;
```

Nota: Esta forma de acceder al DOM, funciona bien en la mayoría de navegadores, aunque en Navegadores con motor basado en Internet Explorer, se puede utilizar una función alternativa.

```
var nombre=document.all('nombre').value;
```

Esto sucede porque en Internet Explorer 6 aun hacen referencia al DOM de nivel 1 y con el metodo getElementById se hace referencia la DOM de nivel 2.

Ahora vayamos a ver la siguiente línea de código:

```
ajax.open("GET","datos_get.php?nombre="+nombre,true);
```

El objeto ajax tiene una serie de métodos, que harán posible que nuestra aplicación funcione, el primer método se llama open y abre una conexión entre nuestra página web y otra pagina(en este caso, la pagina se llama datos_get.php), este método usa 3 parámetros,

1. Método de envío de datos, GET o POST

2. El segundo parámetro es la pagina a la cual deseamos conectarnos, esta puede o no llevar datos, por ejemplo si usamos un método get, puede ser necesario envía variables, por este medio, si es por post, no es necesario.
3. Y por ultimo un valor booleano para saber si al conexión será Asíncrona o síncrona, y se define con las palabras reservadas o valores, true o false.

Al realizar una petición ajax es necesario revisar el estado en el que está en objeto, esto se refiere a , preguntar si la pagina que pedimos se cargo, si ajax aun está enviando datos, esperado respuesta, o si la pagina devuelta existe, entre otros.

En este caso debemos de referirnos al método onreadystatechange, de la siguiente forma:

```
ajax.onreadystatechange=function(){alert("revisando el estado del objeto ajax");}
```

Con este método nos referimos a una función la cual revisara el estado del objeto ajax, el cual consta de 5 estados.

1. No inicializado
2. Conexión establecida
3. Recibiendo respuesta
4. Procesando respuesta
5. Finalizado

Se puede comprobar estado por estado la petición, pero para nuestro ejemplo no es necesario, veamos la función a detalle:

```
ajax.onreadystatechange=function(){  
if(ajax.readyState==4)  
{  
var respuesta=ajax.responseText;  
document.getElementById('resultado').innerHTML=respuesta;  
}  
}
```

Con el método onreadystatechange generamos una función que comprueba si el estado es igual a 4, mediante el metodo ajax.readyState, (si se ha finalizado la petición), y posteriormente creamos una variable, llamada respuesta, la cual contiene la pagina solicitada, esta respuesta puede venir de dos formas.

```
ajax.responseText;
```

```
ajax.responseText;
```

El primero se refiere a que la respuesta será procesada como texto, por ejemplo si pedimos una pagina html, o php, asp, etc se puede utilizar este tipo de respuesta, sin embargo, si la pagina recibida es una XML es necesario usar la segunda opción.

La línea de código que sigue, simplemente le dice a javascript que inserte la respuesta en un elemento de la pagina web referenciado por algún id, puede ser una tabla, un textarea, una celda, un div, etc.

```
document.getElementById('resultado').innerHTML=respuesta;
```

Esta inserción se realiza mediante el método innerHTML, el cual también trabaja de dos formas.

```
document.getElementById('resultado').innerHTML=respuesta;
```

```
document.getElementById('resultado').innerText=respuesta;
```

Con la primera opción, la pagina que, hemos solicitado, se insertara como HTML, esto es si hemos recibido formularios, imágenes, etc.; Si lo que recibimos es solo texto , es recomendable usar innerText.

El ultimo método usado es.

```
ajax.send(null);
```

Este método permite enviar variables, cuando utilizamos el método post para enviar datos, si en cambio se usa el método get, simplemente se concatenan la variables al nombre de la pagina (como lo vimos en el método ajax.open) .

En este caso enviaremos, null, ya que no es necesario enviar más datos.

Ahora creamos la página datos_get.php

```
<?php
```

```
echo $_GET['nombre'] ;
```

```
?>
```

y listo, ya tenemos nuestra primera aplicación con ajax, a la vez que aprendimos de donde salió todo este rollo de ajax.

Puede encontrar este y otros artículos además de más información en la página web de www.Ajaxman.net