# 1 - Qué es AJAX?

AJAX son las siglas de *Asynchronous JavaScript And XML*. No es un lenguaje de programación sino un conjunto de tecnologías (HTML-JavaScript-CSS-DHTML-PHP/ASP.NET/JSP-XML) que nos permiten hacer páginas de internet más interactivas.

La característica fundamental de AJAX es permitir actualizar parte de una página con información que se encuentra en el servidor sin tener que refrescar completamente la página. De modo similar podemos enviar información al servidor.

La complejidad se encuentra en que debemos dominar varias tecnologías:

HTML

CSS

JavaScript

**DHTML** Básicamente debemos dominar todos los objetos que proporciona el DOM.

**XML** Para el envío y recepción de los datos entre el cliente y el servidor.

**PHP** o algún otro lenguaje que se ejecute en el servidor (ASP.Net/JSP/Python/NodeJS)

En este curso suponemos que domina las tecnologías mencionadas, en caso de no ser así recomiendo recorrer los cursos de HTML Ya, JavaScript Ya, CSS Ya, PHP Ya y DHTML Ya, luego todo lo nuevo que aparezca lo iremos explicando lentamente a través de ejemplos.

# 2 - Ventajas y desventajas de AJAX.

Ventajas

1. Utiliza tecnologías ya existentes.
2. Soportada por la mayoría de los navegadores modernos.
3. Interactividad. El usuario no tiene que esperar hasta que llegen los datos del servidor.
4. Portabilidad (no requiere plug-in como Flash)
5. Mayor velocidad, esto debido que no hay que retornar toda la página nuevamente.
6. La página se asemeja a una aplicación de escritorio.

Desventajas

1. Se pierde el concepto de volver a la página anterior.
2. Si se guarda en favoritos no necesariamente al visitar nuevamente el sitio se ubique dónde nos encontrábamos al grabarla.
3. La existencia de páginas con AJAX y otras sin esta tecnología hace que el usuario se desoriente.
4. Problemas con navegadores antiguos que no implementan esta tecnología.
5. No funciona si el usuario tiene desactivado el JavaScript en su navegador.
6. Requiere programadores que conozcan todas las tecnologías que intervienen en AJAX.
7. Dependiendo de la carga del servidor podemos experimentar tiempos tardíos de respuesta que desconciertan al visitante.

# 3 - Un ejemplo con AJAX.

Confeccionaremos un ejemplo donde veremos que aparecen muchos conceptos, no se preocupe si no los comprende en su totalidad ya que los mismos se verán en forma detallada a lo largo de este curso.

La idea fundamental de este ejercicio es conocer cómo debemos estructurar nuestras páginas y ver que introduce de nuevo el empleo de AJAX.

Confeccionaremos un problema muy sencillo, imaginemos que tenemos una lista de hipervínculos con los distintos signos del horóscopo y queremos que al ser presionado no recargue la página completa sino que se envíe una petición al servidor y el mismo retorne la información de dicho signo, luego se actualice solo el contenido de un div del archivo HTML.

Este problema se puede resolver muy fácilmente si refrescamos la página completamente al presionar el hipervínculo, pero nuestro objetivo es actualizar una pequeña parte de la página y más precisamente el div que debe mostrar los datos del signo seleccionado.

Si bien nuestra página solo contendrá los hipervínculos a los distintos signos en un caso real la página puede contener muchos otros elementos HTML con imágenes, otros hipervínculos etc. los cuales no deberán sufrir cambios (ni parpadeo) ya que solo se modificará el elemento div respectivo mediante DHTML.

Esta actualización parcial de la página tiene muchas ventajas:

* Reducimos el ancho de banda requerido al no tener que recuperar toda la página.
* Agilizamos la actualización de la página.
* Reducimos el parpadeo de la página.
* Hacemos más natural la navegación del sitio.

La mayoría de los problemas requieren los siguientes archivos como mínimo:

1. El archivo HTML (es la página que se ve en el navegador)
2. El archivo JS (contiene todas las rutinas JavaScript que permiten actualizar dinámicamente la página HTML (mediante DHTML) y las rutinas que permiten comunicarse con el servidor para el envío y recepción de información)
3. La hoja de estilo, es decir el archivo CSS
4. La página que contiene el script que se ejecuta en el servidor (en nuestro caso emplearemos el lenguaje PHP)

Comencemos a presentar los distintos archivos para resolver este problema:

pagina1.html

Esta página contiene HTML puro. Es importante notar que debemos incorporar los dos archivos externos .css y .js mediante los elementos HTML respectivos:

<script src="funciones.js"></script>

<link href="estilos.css" rel="StyleSheet" type="text/css">

La hoja de estilo solo tiene el objetivo de mejorar la presentación en la página de los doce hipervínculos de los signos del horóscopo. Puede probar de eliminar el archivo .css mediante el borrado del elemento link del archivo HTML y el problema debería continuar funcionando, por supuesto con una presentación mucho más pobre.

Podemos observar que cada hipervínculo solicita la misma página al servidor pero pasándole como parámetro un valor distinto, con esto podremos detectar en el servidor que signo eligió el operador.

El segundo archivo contiene las reglas de estilo que se definen para el archivo HTML:

estilos.css

No haremos un análisis de estas reglas ya que corresponden al tema 28 del curso de CSS Ya "Creación de un menú vertical configurando las pseudo clases", puede refrescar los conceptos allí. Inclusive si todavía conoce poco de CSS y no quiere estudiarlo por ahora puede anular el archivo no incorporándolo en la página HTML suprimiendo el elemento link.

Ahora viene uno de los puntos claves donde debemos prestar más atención, esto se encuentra en las rutinas JavaScript que debemos implementar para comunicarnos con el servidor, además de lo ya conocido de DHTML para añadir elementos HTML en forma dinámica.

Veamos el archivo en su totalidad y expliquemos en forma muy global (recuerde que a lo largo de este curso iremos profundizando todos estos conceptos de comunicación con el servidor):

funciones.js

En este punto si es indispensable haber realizado el curso de DHTML Ya para entender como registramos los eventos para los doce hipervínculos. Recordemos que siempre llamaremos a la función addEventListener.

Lo primero que se ejecuta es la llamada a la función inicializarEventos() inmediatamente luego que la página se ha cargado por completo en el navegador:

function inicializarEventos() {

for(var f=1;f<=12;f++) {

var ob=document.getElementById('enlace'+f);

ob.addEventListener('click',presionEnlace,false);

}

}

En esta función registramos el evento click para los doce enlaces de los signos del horóscopo. Para facilitar la codificación recordemos que todos tienen casi el mismo nombre, difieren por un número al final. Luego dentro de un for rescatamos la referencia a cada enlace y registramos el evento click indicando que se debe llamar a la función presionEnlace().

La función presión enlace:

function presionEnlace(e)

{

e.preventDefault();

var url=e.target.getAttribute('href');

cargarHoroscopo(url);

}

Primero desactivamos el evento por defecto para el hipervínculo, luego llama a la función cargarHoroscopo pasándole como referencia la url que contiene el hipervínculo.

Todo lo comentado hasta acá se estudió en cursos anteriores: HTML, JavaScript, CSS, DHTML.

Veamos ahora la función cargarHoroscopo:

var conexion1;

function cargarHoroscopo(url)

{

conexion1=new XMLHttpRequest();

conexion1.onreadystatechange = procesarEventos;

conexion1.open("GET", url, true);

conexion1.send();

}

Previo a la definición de esta función definimos una variable global llamada conexion1 que será utilizada en esta y la siguiente función.

La función recibe como parámetro la url a la que debe hacer la petición de datos.

Lo primero que hacemos es crear un objeto de la clase XMLHttpRequest (luego veremos que este objeto nos permite comunicarnos con el servidor de forma asincrónica):

conexion1=new XMLHttpRequest();

La propiedad onreadystatechange se inicializa con la referencia de una función que será la encargada de procesar los datos enviados por el servidor, veremos el código de esta función más adelante.

Seguidamente llamamos al método open que tiene tres parámetros:

Primero el método de envío de datos (GET o POST) Recordemos que si los datos se envían como parámetros (como es nuestro ejemplo) debemos indicar que utilizamos el método GET

El segundo parámetro es la url y la página que procesará los datos que le eviemos.

El tercer parámetro indicamos si se procesarán los datos de forma asíncrona (true) o síncrona (false)

Por último nos falta llamar al método send para que comience el proceso:

conexion1.send();

Nos queda explicar la función procesarEventos que se ejecuta cada vez que el objeto conexion1 de la clase XMLHttpRequest cambia de estado. Tengamos en cuenta que los estados posibles de este objeto son:

0 No inicializado.

1 Cargando.

2 Cargado.

3 Interactivo.

4 Completado.

Para conocer el estado del objeto debemos acceder a la propiedad readyState que almacena alguno de los cinco valores que enunciamos.

Nuestra función procesarEventos es:

function procesarEventos()

{

var detalles = document.getElementById("detalles");

if(conexion1.readyState == 4)

{

detalles.innerHTML = conexion1.responseText;

}

else

{

detalles.innerHTML = 'Cargando...';

}

}

Decíamos que cuando la propiedad readyState almacena 4 significa que todos los datos han llegado desde el servidor, luego mediante el método responseText recuperamos la información enviada por el servidor. Luego cualquier otro valor que contenga la propiedad readyState mostramos dentro del div el mensaje 'cargando...'.

Es seguro que muchas dudas han surgido de este primer pantallazo de AJAX, pero no se preocupe a medida que avancemos en el curso se irán aclarando e internalizando.

Pero todavía nos queda la página que contiene el programa en el servidor, en nuestro caso empleamos el lenguaje PHP (tener en cuenta que podemos emplear otro lenguaje de servidor para esto)

Veamos el código de esta página: