Hola amigos, soy Luis Anaya, y voy a explicarles un poco sobre AJAX, pues en pocas palabras y para que se entienda rápido, AJAX son varias tecnologías juntas, para poder utilizar esta mezcla de tecnologías es necesario saber usarlas todas (HTML o XHTML, CSS, JAVASCRIPT, XML, DOM y algún lenguaje de servidor –PHP, JSP, ASP, PERL, PHYTON, etc-.).

Buenos como es solo para iniciar y saber de se trata AJAX voy a dar la explicación y un ejemplo, este tutorial es muy básico, ideal para los que no han usado antes AJAX.

Lo que necesitamos para iniciar es:

1. Un editor de textos (bloc de notas, NotePad++, Dreamweaver, etc.)
2. Hacer nuestro pc un servidor local, para esto podemos instalar WAMP, XAMP o solo el Apache.

Ahora bien, AJAX hará la comunicación asíncrona con el servidor, o sea, lo hará en segundo plano, esto quiere decir no refrescaremos la página, solamente el “pedazo” o sección que necesitamos refrescar.

Es básico saber manejar el DOM (Document Object Model) para trabajar con AJAX, siempre acostumbrémonos a separar todo al hacer o desarrollar web, o sea, aparte el CSS, aparte el JAVASCRIPT, aparte el HTML y aparte el lenguaje de servidor, esto hará que nuestros sitios sean más fáciles de entender y mantener.

Como estoy explicando AJAX desde cero (espero me dé a entender, sino pregunten por el grupo o al email luigui.a@gmail.com), empezaré con el clásico “Hola Mundo”, que es lo más básico para empezar a entender todo esto.

1. Crearemos un documento txt, le pondré de nombre “inicio.txt”. En él escribiremos el famoso “Hola Mundo”.

Inicio.txt

```
Hola mundo, este es un primer script con ajax!!
```

2. Ahora crearemos el documento html

index.html

```
<!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>
<link rel="stylesheet" type="text/css" href="estilo.css" />
<script type=”text/javascript” src=”js/mi_script.js” ></script> //Separamos funcionalidad de contenido
<title>Hola Mundo con AJAX</title>
</head>
<body>
  <p>Este texto siempre se mira, y no se actualiza</p>
  <div id=”aqui”>Haz clic sobre mi</div>
  <p>Ese es el famoso Hola mundo, y se ha llamado del servidor con AJAX</p>
</body>
</html>
```

3. Ahora haremos el archivo javascript para utilizar AJAX, y aquí viene lo más difícil si es que no manejas bien JAVASCRIPT y el DOM, pero no te preocupes, te comentaré todo lo que haga. Algo que debe quedar claro es que el objeto XMLHttpRequest es el que permite crear la comunicación asíncrona con el servidor, o sea, este objeto es el que hace todo la comunicación con el servido en segundo plano.

mi_script.js

```
addEvent(window,"load",iniciar,false); // Esta función se ejecutará cuando el árbol del DOM se haya cargado por completo en tu navegador, en otras palabras, cuando la página cargue por completo esta función se ejecutará automáticamente, es por eso el "load".
//Los parámetros que le pasamos son el elemento que ejecutará el evento (window), luego el evento (load), la función que ejecutará (iniciar) y el burbujeo como le llaman algunos, o captura (false), este solo puede ser true o false.

function iniciar() {
        var obj = document.getElementById("aqui"); //Esto es DOM, hago referencia al objeto del documento cuyo ID sea "aqui", en nuestro caso el div que le puse id="aqui".
        addEvent(obj,"click",presion,false); //Igual que la primera función que se ejecutará, solo que esta se ejecutará al hacer click sobre el div "aqui".
}

var conexion; //Este será nuestro objeto que mantendrá la conexion y hará las peticiones
```

```
function presion() { //Función que se ejecutará cuando se presiones con el click sobre el div llamado "aqui"
	conexion = crearXMLHttp(); //Ahora conexion se convierte en una función para crear un objeto XMLHttpRequest
	conexion.onreadystatechange = procesar; //Ahora cuando cambie el estado de la conexion con el servidor se ejecutará esta función
	conexion.open("GET","inicio.txt",true); //esta función abre una página, en este caso abrirá el bloc de notas llamado inicio.txt, le decimos que lo hará con el método GET, luego el archivo a abrir, y el tercer parámetro indica que la comunicación es asíncrona, si fuera false este parámetro, la comunicación fuera síncrona y el navegador se congelara hasta que se terminara de ejecutar la petición al servidor, esto sirve solo cuando se necesita que es usuario no haga nada más mientras se ejecuta la petición.
	conexion.send(null); //Con esta función indicamos que no enviamos nada al servidor, se le tiene que pasar como parámetro el valor "null".
}

function procesar() { //Esta es la funcion que se ejecutará cuando el estado vaya cambiando de estado, que es eso de los estado? ya se los diré.
	var detalle = document.getElementById("aqui"); //Volvemos a hacer referencia al div id="aqui"
	if(conexion.readyState == 4) { //El estado que el servido devuelve es 0,1,2,3 y 4 (No inicializado, Cargando, Cargado, Interactivo, Completado) respectivamente, entonces cuando la conexion sea 4 (Completada) se pasará al siguiente if
		if(conexion.status == 200) { //Ahora el Status, este cuando sea 200 quiere decir que no hubo problemas, este no lo explico porque es más extenso, otro ejemplo es el 404, cuando una página no se ha encontrado en le servidor, creo que todos nos hemos encontrado más de algunas vez con este error mientras navegamos
			detalle.innerHTML = conexion.responseText; //Ahora decimos que en el div agregaremos la repuesta de la conexion, en este caso escribirá "Hola Mundo" en el div
		} else alert("Error: "+conexion.statusText); //En caso que no se llegue a dar el status = 200 indicaremos al usuario el cual es el error por medio de un alert.
	} else detalle.innerHTML = "Cargando..."; //Mientras el estado de la petición no sea 4 (o sea, 0,1,2 y 3) mostraremos en el div un texto que diga cargando
}

function addEvent(elemento, evento, funcion, captura) { //Esta es la función que se ejecutará al iniciar este script, como había dicho, se le indica el elemento que lo ejecuta, el evento que lo llama, la función que ejecutará y la captura para todo los navegadores que respetan los estándares
	if(elemento.attachEvent) { //Aquí verificamos que el navegador llama ejecuta una función attachEvent, los que no respetan los standares (Internet Explorer)
		elemento.attachEvent('on'+evento,funcion); //Si es el IE entonces al evento le anteponemos la palabra "on", así como cuando decimos onclick, onfocus, etc., luego le indicamos la función que ejecutará
		return true; //decimos que devuelva true para que se ejecute
```

```
        } else if(elemento.addEventListener) { //Detectamos un navegador que respeta los estándares FireFox, Opera, Safari, Chrome, etc...
                elemento.addEventListener(evento,funcion,captura); //Ahora le pasamos el evento que lo llama, la función a ejecutar y la captura (esto no lo explico aquí porque es otro tema)
                return true; //decimos que devuelva true para que se ejecute
        } else return false; // Sino es ninguno de los anteriores, ni el IE, ni otro que respete los estándares de la W3C decimos que no lo ejecute
}

function crearXMLHttp() { //Ahora viene el corazón de AJAX, el que hace la conexion asíncrona
        var xmlhttp = null; //decimos que el objeto xmlhttp es nulo
        if(window.ActiveXObject) { //Verificamos si es el IE 5, 5.5 o 6 el que quiere crear el objeto XMLHttpRequest, en este caso IE lo toma como un objeto ActiveXObject
                xmlhttp = new ActiveXObject("Microsoft.XMLHTTP"); //Si es el IE crea el objeto necesario
        } else if(window.XMLHttpRequest) { //Si es cualquier otro navegador, FireFox 1.0 o superior, Safari, Opera IE 7 o superior, Chrome, etc.
                xmlhttp = new XMLHttpRequest(); //Se crea el objeto XMLHttpRequest
        } return xmlhttp; //Devolvemos el objeto para que pueda ser utilizado
}
```

4. Por último podemos poner unos estilos a la página para ver más claro adonde se da clic, y que sea más vistosa la presentación, claro en este ejemplo, será muy sencillo es código.

estilo.css

```
#aqui {
  background-color:#18128B;
  border:1px solid #000;
  padding:10px;
  margin:5px;
  width:400px;
  text-align:center;
  color:#fff;
}
#aqui:hover {
  color:#18128B;
  background-color:#CCCCCC;
}
```

El código css es muy sencillo, así que no lo explicaré, se sobre entiende, pero si no entiendes o deseas aprender css avísame al grupo o al email que puse al inicio del tutorial.

Ahora solo queda correrlo desde el servidor, y verás que funciona bien. A lo mejor este tutorial no demuestre todo el potencial que tiene AJAX, pero da una idea de lo que puede hacer, como dije al principio, es un tutorial básico.

Junto al tutorial adjuntaré los archivos que se han creado para este tutorial, para que lo entiendas mejor te aconsejo que veas los archivos, ahí van comentados, si es posible verlo con un editor especializado (Notepad++, dreamweaver u otro), así lograrás distinguir entre los comentarios y el código.

He tratado de ser lo más explicativo que he podido, ya sabes, cualquier pregunta hazla, no te quedes con la duda. Hasta la próxima.