INTRODUCCION

La historia de AJAX está íntimamente relacionada con un objeto de programación llamado **`XMLHttpRequest.`**

Todas las aplicaciones realizadas con técnicas de AJAX deben **instanciar** en primer lugar el objeto `XMLHttpRequest`, que es el **objeto clave** que permite realizar **comunicaciones** con el servidor en segundo plano, sin necesidad de recargar las páginas. La implementación del objeto `XMLHttpRequest` depende de cada navegador, por lo que es necesario emplear una discriminación sencilla en función del navegador en el que se está ejecutando el código:

```
if(window.XMLHttpRequest) { // Navegadores que siguen los estándares
	peticion_http = new XMLHttpRequest();
}
else if(window.ActiveXObject) { // Navegadores obsoletos
	peticion_http = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Los navegadores que siguen los estándares (Firefox, Safari, Opera, Internet Explorer 7 y 8) implementan el objeto `XMLHttpRequest` de forma nativa, por lo que se puede obtener a través del objeto `window`. Los navegadores obsoletos (Internet Explorer 6 y anteriores) implementan el objeto `XMLHttpRequest` como un objeto de tipo ActiveX. Una vez obtenida la instancia del objeto `XMLHttpRequest`, se prepara la función que se encarga de procesar la respuesta del servidor. La *propiedad* **`onreadystatechange`** del *objeto* **`XMLHttpRequest`** permite indicar esta función directamente incluyendo su código mediante una **función anónima** o indicando una referencia a una función independiente.

```
peticion_http.onreadystatechange = muestraContenido;
```

El código anterior indica que cuando la aplicación reciba la respuesta del servidor, se debe ejecutar la función `muestraContenido()`. Como es habitual, **la referencia a la función se indica mediante su nombre sin paréntesis,** *ya que de otro modo se estaría ejecutando la función y almacenando el valor devuelto en la propiedad* `onreadystatechange`.

Después de **preparar** la aplicación para la **respuesta** del servidor, se realiza la petición HTTP al servidor:

```
peticion_http.open('GET', 'http://localhost/prueba.txt', true);
peticion_http.send(null);
```

Las instrucciones anteriores realizan el tipo de petición más sencillo que se puede enviar al servidor. En concreto, se trata de una petición de tipo `GET` simple que no envía ningún parámetro al servidor. La petición HTTP se crea mediante el método `open()`, en el que se incluye el tipo de petición (`GET`), la URL solicitada (`http://localhost/prueba.txt`) y un tercer parámetro que vale `true`.

Una vez creada la petición HTTP, se envía al servidor mediante el método `send()`. Este método incluye un parámetro que en el ejemplo anterior vale `null`. Más adelante se ven en detalle todos los métodos y propiedades que permiten hacer las peticiones al servidor. Por último, cuando se recibe la respuesta del servidor, la aplicación ejecuta de forma automática la función establecida anteriormente.

```
function muestraContenido() {
	if(peticion_http.readyState == 4) {
		if(peticion_http.status == 200) {
		alert(peticion_http.responseText);
		}
	}
}
```

La función `muestraContenido()` comprueba en primer lugar que se ha recibido la respuesta del servidor (mediante el valor de la propiedad `readyState`). Si se ha recibido alguna respuesta, se comprueba que sea válida y correcta (comprobando si el código de estado HTTP devuelto es igual a `200`). Una vez realizadas las comprobaciones, simplemente se muestra por pantalla el contenido de la respuesta del servidor (en este caso, el contenido del archivo solicitado) mediante la propiedad `responseText`.

Métodos y propiedades del objeto XMLHttpRequest

El objeto `XMLHttpRequest` posee muchas otras propiedades y métodos diferentes a las manejadas por la primera aplicación de AJAX. A continuación se incluye la lista completa de todas las propiedades y métodos del objeto y todos los valores numéricos de sus propiedades. Las propiedades definidas para el objeto `XMLHttpRequest` son:

Propiedad	Descripción
`readyState`	Valor numérico (entero) que almacena el estado de la petición
`responseText`	El contenido de la respuesta del servidor en forma de cadena de texto
`responseXML`	El contenido de la respuesta del servidor en formato XML. El objeto devuelto se puede procesar como un objeto DOM
`status`	El código de estado HTTP devuelto por el servidor (`200` para una respuesta correcta, `404` para "No encontrado", `500` para un error de servidor, etc.)
`statusText`	El código de estado HTTP devuelto por el servidor en forma de cadena de texto: "OK", "Not Found", "Internal Server Error", etc.

Los valores definidos para la propiedad `readyState` son los siguientes:

Valor	Descripción
`0`	No inicializado (objeto creado, pero no se ha invocado el método `open`)
`1`	Cargando (objeto creado, pero no se ha invocado el método `send`)
`2`	Cargado (se ha invocado el método `send`, pero el servidor aún no ha respondido)
`3`	Interactivo (se han recibido algunos datos, aunque no se puede emplear la propiedad `responseText`)
`4`	Completo (se han recibido todos los datos de la respuesta del servidor)

Los métodos disponibles para el objeto `XMLHttpRequest` son los siguientes:

Método	Descripción
`abort()`	Detiene la petición actual
`getAllResponseHeaders()`	Devuelve una cadena de texto con todas las cabeceras de la respuesta del servidor
`getResponseHeader("cabecera")`	Devuelve una cadena de texto con el contenido de la cabecera solicitada
`onreadystatechange`	Responsable de manejar los eventos que se producen. Se invoca cada vez que se produce un cambio en el estado de la petición HTTP. Normalmente es una referencia a una función JavaScript
`open("metodo", "url")`	Establece los parámetros de la petición que se realiza al servidor. Los parámetros necesarios son el método HTTP empleado y la URL destino (puede indicarse de forma absoluta o relativa)
`send(contenido)`	Realiza la petición HTTP al servidor
`setRequestHeader("cabecera", "valor")`	Permite establecer cabeceras personalizadas en la petición HTTP. Se debe invocar el método `open()` antes que `setRequestHeader()`

El método ***open***() requiere dos parámetros (método HTTP y URL) y acepta de forma opcional otros tres parámetros. Definición formal del método `open()`:

```
open(string metodo, string URL [,boolean asincrono, string usuario, string password]);
```

Por defecto, las peticiones realizadas son asíncronas. Si se indica un valor `false` al tercer parámetro, la petición se realiza de forma síncrona, esto es, se detiene la ejecución de la aplicación hasta que se recibe de forma completa la respuesta del servidor.

No obstante, las peticiones síncronas son justamente contrarias a la filosofía de AJAX. El motivo es que una petición síncrona *congela* el navegador y no permite al usuario realizar ninguna acción hasta que no se haya recibido la respuesta completa del servidor. La sensación que provoca es que el navegador se ha *colgado* por lo que no se recomienda el uso de peticiones síncronas salvo que sea imprescindible.

Los últimos dos parámetros opcionales permiten indicar un nombre de usuario y una contraseña válidos para acceder al recurso solicitado. Por otra parte, el método `send()` requiere de un parámetro que indica la información que se va a enviar al servidor junto con la petición HTTP. Si no se envían datos, se debe indicar un valor igual a `null`. En otro caso, se puede indicar como parámetro una cadena de texto, un array de bytes o un objeto XML DOM.