Laboratorio III
AJAX

Clase 3 - p1

Maximiliano Neiner

Temas a Tratar

- AJAX

Temas a Tratar

- AJAX

Peticiones HTTP

- Una petición **HTTP** es como suele denominarse a la acción por parte del navegador de solicitar a un *servidor web* un documento o archivo, ya sea un fichero .html, una imagen, un archivo .js, etc.

- Garcias a dicha petición, el navegador puede descargar estos archivos, almacenarlos en un ***chaché*** temporal y, finalmente, mostrarlos en la página actual que lo haya solicitado.

AJAX (**A**synchronous **J**avaScript **A**nd **X**ML)

- No es un lenguaje de programación sino un conjunto de tecnologías que nos permiten hacer páginas de internet más interactivas.
- AJAX permite que las páginas sean capaces de actualizar partes en segundo plano, sin volver a cargar toda la página.
- La petición HTTP se realiza desde javascript, de forma transparente al usuario.
- AJAX se hizo popular en 2005 por Google, con Google Suggest.

Tecnologías

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

Nota: Actualmente, se utiliza más el formato JSON que el XML, pero por razones fonéticas se mantiene el término AJAX en lugar de AJAJ.

Métodos de petición AJAX

Existen varias formas de realizar peticiones HTTP mediante AJAX, las principales son:

- **XMLHttpRequest**: el más antiguo. Nativo
- **fetch**: Nuevo sistema nativo de peticiones basado en promesas. ECMAScript 6.

XMLHttpRequest (1/5)

- La piedra angular de AJAX es el objeto XMLHttpRequest y es utilizado para intercambiar datos con el servidor en segundo plano.
- Para enviar una petición al servidor, se utilizan los métodos ***open()*** y ***send()***.

```
xhttp.open("Método", "Url", Async);
```

- *Método*: Especifica el tipo de pedido (GET/POST).
- *Url*: Indica la ubicación del archivo en el servidor.
- *Async*: true (Asincrónico); false (Sincrónico).

XMLHttpRequest (2/5)

```
xhttp.send(); //Si es GET
xhttp.send("string"); //Si es POST
```

- Envía la petición al servidor.

```
let xhttp = new XMLHttpRequest();
xhttp.open("GET", "ajax_test.php", true);
xhttp.send();
```

- Recibe la respuesta desde el servidor.

```
console.log(xhttp.responseText);
```

XMLHttpRequest (3/5)

- Propiedades

Propiedad	Descripción
STRING .responseType	Define el tipo de respuesta de .response: json, Blob, etc. Por defecto, text.
OBJECT .response	Contenido parseado automáticamente basado en .responseType.
STRING .responseText	Respuesta de la petición como texto plano, o NULL si no se ha recibido.
STRING .responseURL	URL de la petición HTTP a realizar.
NUMBER .readyState	Número que indica en que estado se encuentra la petición (ver más adelante).
NUMBER .timeout	Milisegundos permitidos para realizar la petición HTTP. Por defecto, 0 (sin límite).
NUMBER .status	Código de error HTTP de respuesta de la petición.
STRING .statusText	Texto con el código de error de respuesta, legible para humanos.
BOOLEAN .withCredentials	Indica si la petición HTTP se está realizando con credenciales.

XMLHttpRequest (4/5)

- Métodos

Método	Descripción
Cabeceras HTTP	
`.setRequestHeader(` STRING `name,` STRING `value)`	Permite añadir la cabecera `name` con el valor `value` a la petición HTTP.
STRING `.getAllResponseHeaders()`	Obtiene todas las cabeceras HTTP de la respuesta de la petición.
STRING `.getResponseHeader(` STRING `name)`	Obtiene una cabecera HTTP concreta de la respuesta de la petición.
`.overrideMimeType(` STRING `mimetype)`	Permite modificar el MIME (tipo de fichero) de la petición.
Acciones	
`.open(` STRING `method,` STRING `url)`	Permite preparar una petición HTTP.
`.send()`	Envía la petición previamente preparada con `.open()`.
`.send(` OBJECT `body)`	Idem a la anterior, enviando datos en la petición.
`.abort()`	Cancela la petición enviada.

XMLHttpRequest (5/5)

- Eventos

Evento	Descripción
`abort`	Se dispara cuando una petición es cancelada.
`load`	Se dispara cuando una petición se ha completado correctamente.
`loadstart`	Se dispara cuando una petición comienza a cargar datos.
`loadend`	Se dispara cuando una petición termina (con error o sin ellos).
`error`	Se dispara cuando una petición sufre un error.
`timeout`	Se dispara cuando una petición agota el tiempo máximo.
`progress`	Se dispara (varias veces) cuando una petición recibe datos.
`readystatechange`	Se dispara cuando el valor `.readyState` cambia.

onreadystatechange (1/2)

- El evento ***readystatechange*** se dispara cada vez que cambia el valor de ***readyState***.
- La propiedad readyState mantiene el estado de *XMLHttpRequest*.
- La propiedad readyState indica el estado de la petición.

Valor	Constante	Descripción
0	XMLHttpRequest.UNSENT	Estado inicial. No se ha ejecutado aún .open().
1	XMLHttpRequest.OPENED	Se ha ejecutado open(), pero no se ha ejecutado aún .send().
2	XMLHttpRequest.HEADERS_RECEIVED	Se ha ejecutado send(). Cabeceras recibidas.
3	XMLHttpRequest.LOADING	Descarga de información en proceso.
4	XMLHttpRequest.DONE	Descarga finalizada.

onreadystatechange (2/2)

- **onreadystatechange**: Almacena una función (o el nombre de una función) que se invoca de forma automática cada vez que cambia de propiedad readyState.
- **status**: 200 –> OK; 404 –> No encontrado.


```
let xhttp = new XMLHttpRequest();
xhttp.open("GET", "ajax_test.php", true);
xhttp.send();

xhttp.onreadystatechange = () => {
   if(xhttp.readyState == 4 && xhttp.status == 200){
      console.log(xhttp.responseText);
   }
}
```

Demo

Ejercitación