TEMA 10. AJAX .

Desarrollo Web en Entorno Cliente.

Profesor: Juan José Gallego García

Índice :

AJAX

Ajax, **A**synchronous **J**avaScript **A**nd **X**ML (JavaScript Asíncrono y XML**)** es una técnica de desarrollo web para recuperar contenido, normalmente de un servidor de forma asíncrona.

- Puede leer contenido del servidor una vez que se ha cargado la página.
- Carga ese contenido sin tener que refrescar la página.
- Envía datos al servidor de forma asíncrona (en segundo plano).
- Aunque en su definición usa el término XML, igualmente puede transferir otros contenidos en formato TXT o JSON.

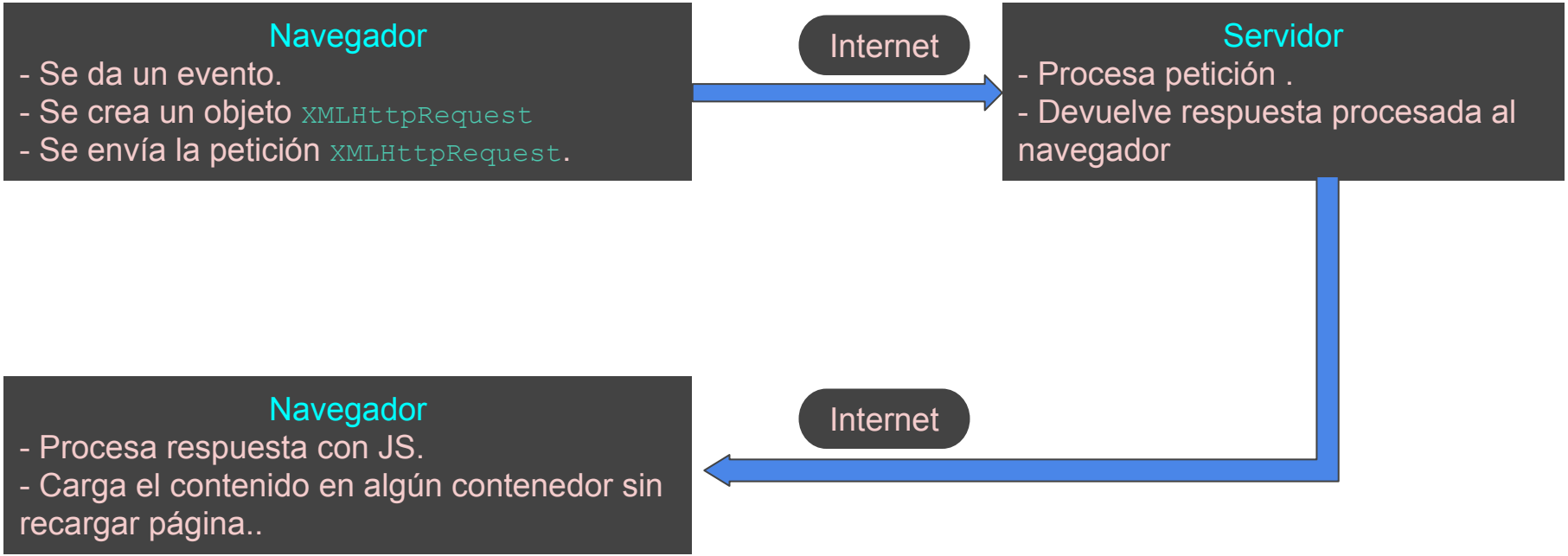
El siguiente ejemplo, lee una propiedad de un archivo JSON y la muestra en un contenedor (DIV) sin tener que cargar de nuevo la página. Dicho archivo JSON debe estar alojado en el mismo dominio (como ya vimos anteriormente), ya que por seguridad los navegadores prohíben las peticiones a otros dominios, excepto si se usan cabeceras CORS o JSONP .
Es necesario instalar los servicios del paquete XAMPP para simular servidor.

Ejemplo

```
<!DOCTYPE html>
<html lang="es">
    <meta charset="UTF-8">
<body>
    <button type="button" onclick="funAjax ()">Leer propiedad de archivo JSON</button>
    <div id="datos"> </div>
    <script>
    function funAjax() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
            let obj=JSON.parse(this.responseText);
            document.getElementById("datos").innerHTML = obj[0].nombre;
            }
        }
        xhttp.open("GET", "carteras.json", true);
        xhttp.send();
    }
    </script>
</body>
</html>
```

Archivo carteras.json

Funcionamiento de peticiones AJAX



Objeto XMLHttpRequest

Es el objeto clave para el intercambio de datos con el servidor.

Propiedades

```
onreadystatechange  // Define una función que se llama cuando hay un cambio en el estado de la petición.
readyState          // Mantiene el estado del objeto XMLHttpRequest:
                            0: solicitud o petición no iniciada.
                            1: conexión con servidor establecida.
                            2: petición recibida.
                            3: petición procesada.
                            4: petición finalizada y respuesta preparada.
responseText        // Devuelve la respuesta como una cadena.
responseXML         // Devuelve la respuesta como XML.
status              // Devuelve el código del estado de la petición procesada:
                            200: "OK".
                            403: "Forbidden".
                            400: "Not Found".
statusText          // Devuelve el texto del estado de la petición procesada.
```

Métodos

```
new XMLHttpRequest()     // Crea el objeto XMLHttpRequest.
abort ()                 // Cancela la petición actual.
getAllResponseHeaders()  // Devuelve la información de cabecera.
getResponseHeader()      // Devuelve información sobre una cabecera específica.
open(method, url, async, user, psw) // Especifica parámetros de la petición.
                                        method: el tipo GET o POST
                                        url: la ruta del archivo
                                        async: true (asíncrono) or false (síncrono)
                                        user: opcional, nombre de usuario
                                        psw: opcional, contraseña
send()                   // Envía la petición al servidor, usada para peticiones GET

send(cadena)             // Envía la petición al servidor, usada para peticiones POST
setRequestHeader()       // Añade el valor de una cabecera para ser enviada.
```

GET vs POST

Tenemos estos dos métodos para realizar la petición :

GET:

Este método es más sencillo y rápido que POST.

POST:

- No usa caché, por tanto actualiza los datos en el servidor.
- No tiene limitaciones de tamaño.
- Al usar la opción de usuario y contraseña, es más seguro.

Ejemplo GET

El siguiente ejemplo muestra cómo hacer una solicitud con AJAX, enviar una serie de parámetros por GET y recibir valores en JSON procesados por PHP desde el servidor, para cargarlos en un contenedor.

```
<!DOCTYPE html>
<html lang="es">
    <meta charset="UTF-8">
<body>
    <button type="button" onclick="funAjax ()">Envía parámetros por GET y recibe valores</button>
    <div id="datos"> </div>
    <script>
    function funAjax() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
            let obj=JSON.parse(this.responseText);
            document.getElementById("datos").innerHTML = `Desde servidor ${obj.nombre} de ${obj.ciudad}` ;
            }
        }
        xhttp.open("GET", "jsonGET.php?nombre=Juan&ciudad=Ubrique", true);
        xhttp.send();
    }
    </script>
</body>
</html>
```

--->

jsonGET.php

```php
<?php
// Para solicitudes de otros dominios.
header("access-control-allow-origin: *");
//.....................................
$nombre = $_GET['nombre'];
$ciudad = $_GET['ciudad'];
// Devuelve JSON
echo '{"nombre":"'.$nombre.'","ciudad":"'.$ciudad.'"}';
?>
```

Ejemplo POST

En el siguiente se puede ver cómo hacer uso del método POST (con AJAX) para enviar y recibir datos al servidor. En este caso hay que establecer en la cabecera el tipo de dato que se va a enviar como parámetro, y en lugar de enviarlos junto con la llamada al archivo PHP, se envía en la llamada del método send('cadena_parámetros')

```html
<!DOCTYPE html>
<html lang="es">
    <meta charset="UTF-8">
<body>
    <button type="button" onclick="funAjax ()">Envía parámetros por POST y recibe valores</button>
    <div id="datos"> </div>
    <script>
    function funAjax() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
            let obj=JSON.parse(this.responseText);
            document.getElementById("datos").innerHTML = `Desde servidor ${obj.nombre} de ${obj.ciudad}` ;
            }
        }
        xhttp.open("POST", "jsonPOST.php", true);
        xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        xhttp.send("nombre=Juan&ciudad=Ubrique");
    }
    </script>
</body>
</html>
```

--->

jsonPOST.php

```php
<?php
// Para solicitudes de otros dominios.
header("access-control-allow-origin: *");
//.....................................
$nombre = $_POST['nombre'];
$ciudad = $_POST['ciudad'];
// Devuelve JSON
echo '{"nombre":"'.$nombre.'","ciudad":"'.$ciudad.'"}';
?>
```

Ejemplo: llamada a varias funciones

En el ejemplo que viene a continuación se puede ver cómo ejecutar distintas funciones con el mismo objeto XMLHttpRequest. Para ello se pasan como parámetros la URL y la función a ejecutar. En este caso sólo se usa una función fun1( ) pero igualmente se haría para cualquier número de funciones.

```
<body>
    <button type="button" onclick="funAjax ('jsonGET.php?nombre=Juan&ciudad=Ubrique',fun1)">
        Envía parámetros por GET y recibe valores</button>
    <div id="datos"> </div>
    <script>
    function funAjax(url,fun) {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                fun(this);
            }
        }
        xhttp.open("GET", url, true);
        xhttp.send();
    }

    function fun1 (xhttp){
        let obj=JSON.parse(xhttp.responseText);
        document.getElementById("datos").innerHTML = `Desde servidor ${obj.nombre} de ${obj.ciudad}` ;
    }
    </script>
</body>
```

```
<form id='formu'>
    Nombre: <input name='nombre'><br>
    Apellido: <input name='apellido'><br>
    <input type='submit'>
</form>
<div id="datos"></div>
<script>
    var formu = document.getElementById('formu');
    formu.onsubmit = function(e) {
        e.preventDefault(); // Para no enviar el formulario a otra página.
        var solicitud = new XMLHttpRequest();
        solicitud.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                let obj=JSON.parse(this.responseText);
                document.getElementById("datos").innerHTML = `Desde servidor ${obj.nombre} de ${obj.ciudad}`;
            }
        }
        solicitud.open('POST', 'jsonPOST.php',true);
        var formData = new FormData(document.getElementById('formu'));
        formData.append('ciudad', 'Ubrique'); // Campo añadido.
        solicitud.send(formData);
    }
</script>
```

Ejemplo: Objeto FormData

En este ejemplo se usa el objeto **FormData** para enviar los datos de un formulario HTML con AJAX.
Se puede añadir al formulario un nuevo campo mediante código antes de enviarlo.

Ejemplo: Petición a archivo XML

En el siguiente enlace se muestra un ejemplo que usa el objeto XMLHttpRequest para realizar una petición, en este caso, a un archivo XML.

ajaxXML.html

baseCD.xml

Como mejora en los últimos estándares de JS se ha añadido el método **fetch ( )** que facilita las solicitudes o peticiones al servidor de forma asíncrona, basado en lo que se denominan **promesas** (promises) que estudiaremos en el siguiente tema.
Por otro lado, también hay que mencionar que una complementación de AJAX son los WebSockets empleado principalmente para comunicación bidireccional en aplicaciones de tiempo real como un chat.

Bibliografía

- LibrosWeb
- W3Schools
- Developer Mozilla Docs