DESARROLLO DE APLICACIONES WEB

Tema 6.- Otras Tecnologías

AJAX y Fetch

Contenido

- Introducción
- Objeto XMLHttpRequest
- Pasos para realizar una petición AJAX
- Paso de atributos: petición GET y POST
- Ejemplos y Ejercicios

- Fetch

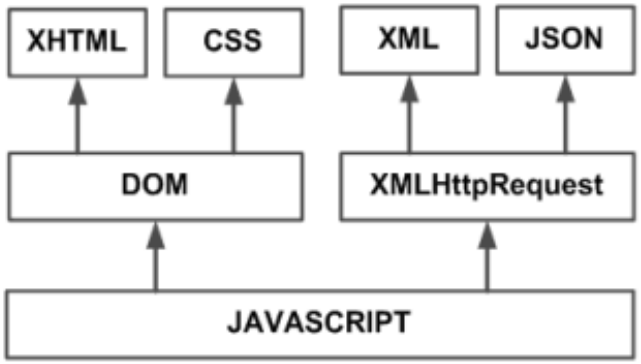
Introducción

- **AJAX** acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML*

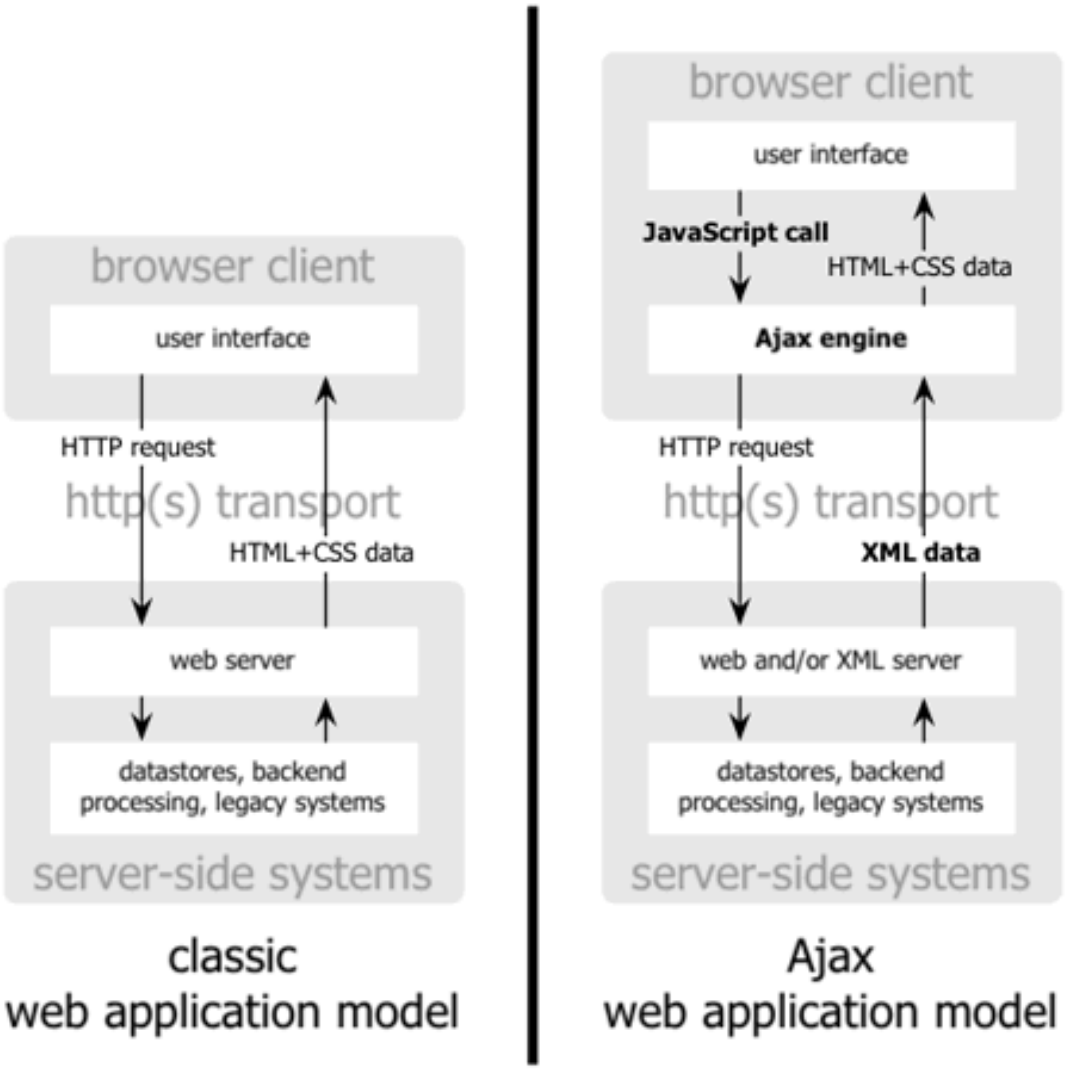
  JavaScript Síncrono y XML

  Permite realizar peticiones en segundo plano cuyo contenido es una parte del documento, sin tener que cargar toda la página, reduciendo, por lo tanto, el tráfico en la red.

- *Tecnologías*



Introducción



Objeto XMLHttpRequest

- Objeto del navegador: XMLHttpRequest

Atributos

- **readyState**: Valor numérico que determina el estado de la petición. Según el estado, este atributo pasa por los siguientes valores: 0 (sin iniciar), 1 (conectando), 2 (solicitando), 3 (procesando), 4 (completada)
- **onreadystatechange**: Atributo que determina la función que se ejecutará cada vez que se detecta un cambio en el estado de la petición.
- **responseText**: Atributo con la respuesta del servidor como una cadena de texto.
- **responseXML**: Atributo con la respuesta del servidor como un XML.
- **status**: Código de estado HTTP del servidor (200, 404, etc.)
- **statusText**: Código de estado HTTP del servidor como texto (OK, Not Found, etc)

Objeto XMLHttpRequest

- Objeto del navegador

Métodos

- **open("method", "url", tipo)**: Inicializa una petición con el método especificado (GET, POST) y la URL indicada, que puede ser asíncrona (por defecto, tipo=true) o síncrona (tipo=false).
- **send(content)**: envía la petición al servidor, donde content son los datos de la petición (null, para GET), en formato var1=val1&var2=val2.
- **abort()**: Cancela la petición actual.
- **setRequestHeader("header", "value")**: define un valor (value) para la cabecera (header).
- **getAllResponseHeaders()**: devuelve las cabeceras de respuesta a la petición HTTP como pares key/valor.
- **getResponseHeader("header")**: devuelve una cadena con el valor de la cabecera especificado en "header".

AJAX en estado puro

Pasos:

- Establecer el elemento, el evento y la función que iniciará la llamada en segundo plano
- Crear instancia de XMLHttpRequest
- Establecer la función que atenderá la petición (que puede ser implementada como función anónima o como una función externa)
- Definir la petición (método y URL)
- Enviar la petición
- Atender la petición comprobando situación y estado de la petición e incluyendo el contenido recibido en el documento

AJAX en estado puro

Pasos:

- Establecer el elemento, el evento y la función que iniciará la llamada en segundo plano

```
<html>
..
<body>
..
<span id="resultado"></span>
...
…
<  ..    onclick= "peticionAjax();"
…
```

AJAX en estado puro

Pasos:

- Establecer elemento, evento y función que iniciará la llamada
- Crear instancia de XMLHttpRequest

```
var xhr;

if(window.XMLHttpRequest)  {  // Navegadores actuales
  xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject) {  // Antiguas versiones IE
  xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
if (xhr==null)
{ alert ("Your browser does not support AJAX!");
   return false;
}
```

AJAX en estado puro

Pasos:

- Establecer el evento y la función que iniciará la llamada en segundo plano
- Crear instancia de XMLHttpRequest (obtenemos xhr)
- Establecer la función que atenderá la petición

  xhr.onreadystatechange=atenderRespuesta;
- Definir la petición (método y URL)

  xhr.open(peticion,url,true);
- Enviar la petición

  xhr.send(cuerpo);
- Atender la petición comprobando situación y estado de la petición e incluyendo el contenido recibido en el documento

AJAX en estado puro

...

- Atender la respuesta

```
// Se ejecutará en cada cambio de la situación de la petición
function atenderRespuesta() { // En este caso
    if (xhr.readyState == 4) { // cuando finalice la petición (4)
      if (xhr.status == 200) { // si ha sido correcta (200)
        // El contenido de la petición se le asigna al objeto resultado
        document.getElementById("resultado").innerHTML= xhr.responseText;
      }
    }
  }
```

AJAX en estado puro

```
…
var xhr

function peticionAjax() {

  xhr = new XMLHttpRequest();
  if (xhr==null)   {
    alert (”Tu navegador no soporta AJAX!");
    return false;
  }
  var url=”mifichero.jsp”;
  xhr.onreadystatechange=atenderRespuesta;
  xhr.open("GET",url,true);
  xhr.send(null);
}

function atenderRespuesta()  {
  if (xhr.readyState==4) {
     if (xhr.status == 200) {
        document.getElementById(”resultado").innerHTML=xhr.responseText;
      }
   }
}
…
```

AJAX: Parámetros petición GET

- Se añaden en la URL

Por ejemplo:

```
var valor1, valor2;
…
var url="miURL";
url=url + "?par1="
        + encodeURIComponent(valor1)
        + "&par2="
        + encodeURIComponent(valor2);
xhr.open("GET", url, true);
xhr.send(null);
…
```

AJAX: Parámetros petición POST

- Se añaden al cuerpo en el método send()

  Por ejemplo:

```
var valor1, valor2;
…
var url="miURL";
var cuerpo= "par1="
                + encodeURIComponent(valor1)
                + "&par2="
                + encodeURIComponent(valor2);
xhr.open("POST", url, true);
xhr.setRequestHeader("Content-Type",
                  "application/x-www-form-urlencoded");
xhr.send(cuerpo);
…
```

AJAX: Ejemplo 1

```
…
<body>

        <h1>Ejemplo AJAX</h1>

        <p onclick="peticion();">Pulsa aquí para hacer una llamada AJAX</p>

        <p id=”r"></p>

 </body>
…
```

```
<!– fichero contenidoAjax.jsp -->
Hola ${param.name} tu dni es ${param.dni}
```

AJAX: Ejemplo 1

```
function peticion() {
   var xhr = new XMLHttpRequest();// Obtiene el objeto XMLHttpRequest
    if (xhr !== null) {
      xhr.onreadystatechange = function() {
          switch (xhr.readyState) {
              case 1: alert("Objeto XHR creado para no definido con OPEN"); break;
              case 2: alert("Método OPEN definido pero no enviado SEND"); break;
              case 3: alert("Petición enviada pero no reibido respuesta"); break;
              case 4: alert("Respuesta servidor recibida");
                      if (xhr.status == 200) {
                              document.getElementById(”r").innerHTML = xhr.responseText;
                      }
            }
        }
     xhr.open("GET", "contenidoAjax.jsp?name=Pepe&dni=11111111", true);
     xhr.send(null);
    }
}
```

AJAX: Ejemplo 2

```
…
<form>
  <label for=”nombre">Nombre:</label>
  <input type="text" id=”nombre" name=”nombre" /><br/>
  <label for=”apellidos”<Apellidos:</label>
  <input type="text" id=”apellidos" name=”apellidos" /><br/>
  <label for="telefono">Teléfono:</label>
  <input type="text" id="telefono" name="telefono" /><br/>

  <input type="button" value=”Envia Petición” onclick=“peticion()” />
</form>
 …
<div id=”r"></div>
…
```

AJAX: Ejemplo 2

```
var xhr;
function peticion() {
  xhr = inicializa_xhr();
  if(xhr) {
    xhr.onreadystatechange = procesaRespuesta;
    xhr.open("POST", ”MiURL", true);
    var n= document.getElementById(”nombre");
    var a= document.getElementById(”apellidos");
    var t = document.getElementById("telefono");
    var c = “nombre=“+encodeURIComponent(n.value)  + “&apellidos=” +
          encodeURIComponent(a.value) + “&telefono=“ + encodeURIComponent(t.value);
    xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    xhr.send(c);
  }
}
```

AJAX: Ejemplo 2

```
function procesaRespuesta() {
  if (xhr.readyState == 4) {
    if (xhr.status == 200) {
      document.getElementById(”r").innerHTML = xhr.responseText;
    }
  }
}
```

Ejercicio

- Carrito de la compra con Almacenamiento Local y mostrar el carrito con AJAX

  - Crea una aplicación que contenga un listado de artículos
  - Estos articulos se pueden ir añadiendo a un carrito de la compra que se implementará con Almacenamiento local (sessionStorage)
  - Existirá un botón para mostrar el carrito que envíará mediante AJAX los elementos almacenados en local al servidor.
  - El servidor atenderá la petición mostrando una vista con el carrito

Fetch

- Fetch es una nueva API Javascript para realizar peticiones HTTP asíncronas utilizando promesas

  https://developer.mozilla.org/es/docs/Web/API/Window/fetch

  - Similitudes de Fetch con AJAX (XMLHttpRequest)
    - Realizan solicitudes asincrónicas HTTP.
    - Permiten varios formatos de datos, como JSON, XML, HTML, etc.
    - Puede configurar encabezados para enviar información adicional.

  - Diferencias de Fecth con AJAX
    - Mayor simplicidad y legibilidad con el uso de Promesas.
    - Admite solicitudes entre dominios.
    - Conveniente para el uso de API que manejan datos JSON.
    - Permite más opciones de configuración, como encabezados, método, modo, caché, etc.

Fetch

- Una promesa (Promise) es un objeto que representa la terminación correcta o el fracaso de una operación asíncrona

```
const promesa = operacionAsincrono();
promesa.then(exitoCallback, falloCallback);
```

- Ejemplo Fetch

```
const promise = fetch("/url");

promise.then(function(response) {
  /* ... */
});
```

Fetch

- Uso de Fetch

  https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Using_Fetch

```
const options = {
  method: "GET“       // headers, body, credentials
};

fetch('https://jsonplaceholder.typicode.com/users', options)
  .then(function (response) {
            if (response.ok) {
               const data = response.text()  // .json(), .blob(), etc.
               data.then( datos => { /** Procesar los datos **/ })
            } else { /* Procesar Respuesta no OK **/  }
  })
. catch (function (error) {
    /* Procesar error petición no completada */
})
```

- Tutorial: https://lenguajejs.com/javascript/peticiones-http/fetch/

Fetch – async/ await - Petición GET

```
<button onclick="verhora()">Refrescar</button>
 <p>Hora conexión: <script>document.write(new Date())</script></p>
 <p>Hora refresco: <span id="hora"></span></p>
```

```
async function verhora() {
    let response = await fetch("hora.jsp"); // GET
    let p = document.getElementById("hora");
    if (response.ok) { // si el HTTP-status es 200-299
        let data = await response.text();
            p.innerHTML = data;
    } else {
        // Procesar error
        p.innerHTML = "Error al refrescar";
    }
}
```

```
 <%@page import="java.time.LocalDateTime"%>
<%
   LocalDateTime locaDate = LocalDateTime.now();
   int hours  = locaDate.getHour();
   int minutes = locaDate.getMinute();
   int seconds = locaDate.getSecond();
   String h = String.format("%02d:%02d:%02d", hours, minutes, seconds);
%>
<%= h %>
```

Fetch – Petición POST

```
<button type="button" onclick="request_post()">Post 1</button>
```

```
async function request_post() {
   let datos = new URLSearchParams("nombre=Pepe&pass=pw")
   let response = await fetch("post.jsp", {
         method: 'POST',
         body: datos
    }); // POST
    if (response.ok) { // si el HTTP-status es 200-299
        const data = await response.text();
        console.log(data);
    } else {
         // Procesar error
         console.log("Error al refrescar");
    }
}
```

```
Hola ${param.nombre}
```

Fetch – Petición POST

```
<button type="button" onclick="request_pos2t()">Post 2</button>
```

```
async function request_post2() {
        let formData = new FormData();
        formData.append("nombre", "Pepe");

        let datos = new URLSearchParams(formData);

        let response = await fetch("post.jsp", {
                method: 'POST',
                body: datos
         }); // POST
         if (response.ok) { // si el HTTP-status es 200-299
                const data = await response.text();
                console.log(data);
          } else {
                // Procesar error
                console.log("Error al refrescar");
            }
         }
```

```
Hola ${param.nombre}
```

Fetch – Petición POST

```
<form id="f">
        Nombre <input type="text" name="nombre" /> <span id="s1"></span><br />
        <button type="button" onclick="return request_post3();">Post 3</button>
</form>
```

```
async function request_post3() {
      let f = document.getElementById("f");
      let  s1 = document.getElementById("s1");
      let formData = new FormData(f);

      let datos = new URLSearchParams(formData);

      let response = await fetch("post.jsp", {
            method: 'POST',
            body: datos
       }); // POST
       if (response.ok) { // si el HTTP-status es 200-299
            const data = await response.text();
            console.log(data);
             s1.innerHTML = data;
        } else {
             console.log("Error al refrescar");
         }
}
```

```
Hola ${param.nombre}
```