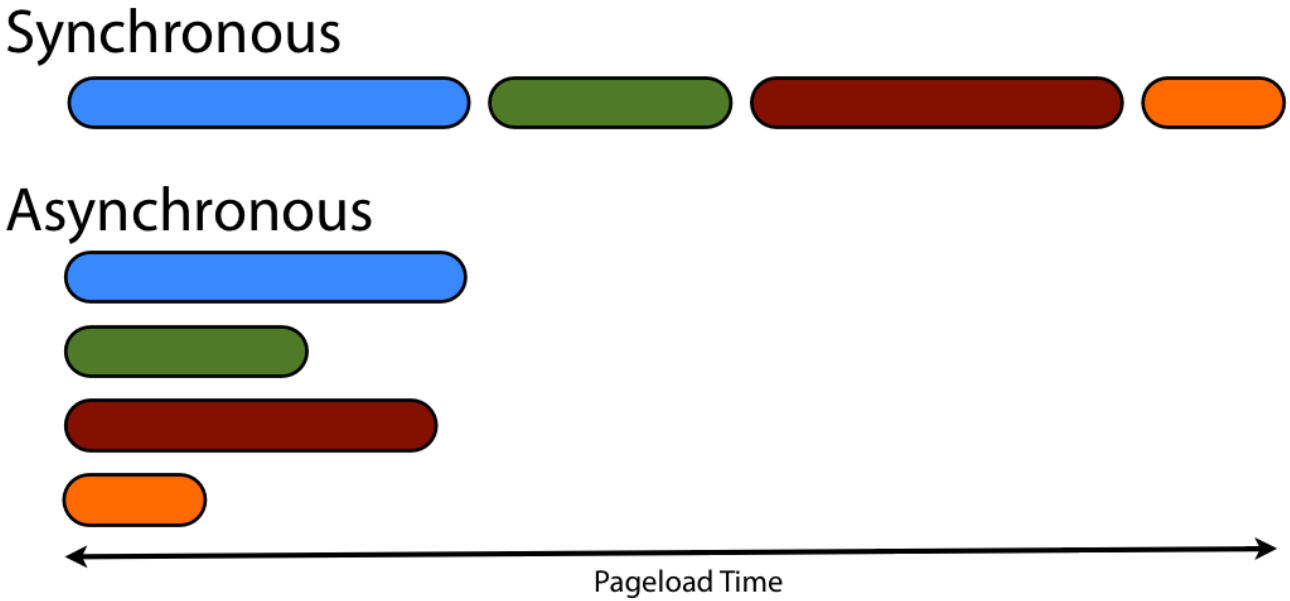


Utilización de mecanismos de comunicación asíncrona

Introducción a Ajax

Ajax:

HTML y CSS (Presentación)

DOM (Interacción y manipulación dinámica de información)

XML, XSLT, JSON (Intercambio y manipulación de información)

XMLHTTPRequest (Intercambio asíncrono de información)

Javascript (Unión de componentes)

Introducción a Ajax

Sin Ajax:

1. El usuario hace solicitud desde la interfaz.
2. Se hace una solicitud HTTP al servidor.
3. *El cliente tiene que esperar hasta recibir la respuesta (cambio de página)*
4. El servidor realiza la acción.
5. El servidor devuelve la página al cliente.

Con Ajax:

1. El usuario hace solicitud desde la interfaz.
2. Se hace una solicitud HTTP al servidor.
3. *El cliente sigue navegando por la página.*
4. Cuando el servidor realiza la acción muestra al cliente sin recargar la página.

Introducción a Ajax

Introducción a Ajax

Servidor web

- Apache, IIS, Nginx, ...

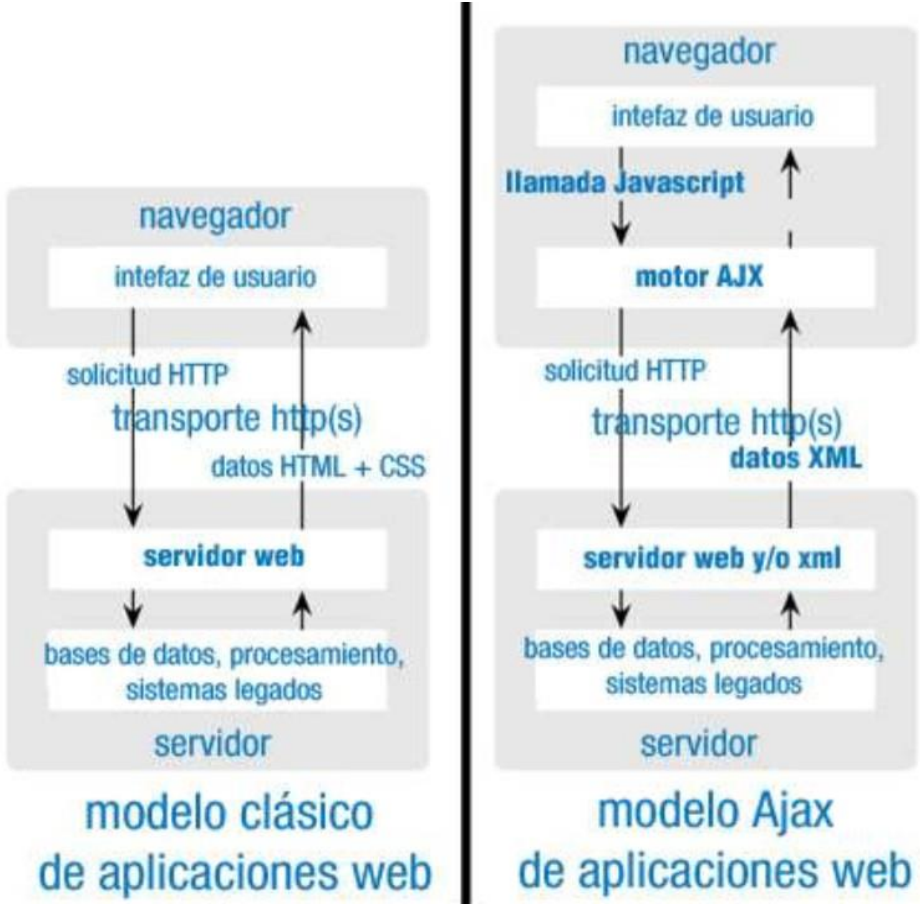
Servidor de bases de datos

- MySQL, PostgreSQL, NoSQL, ...

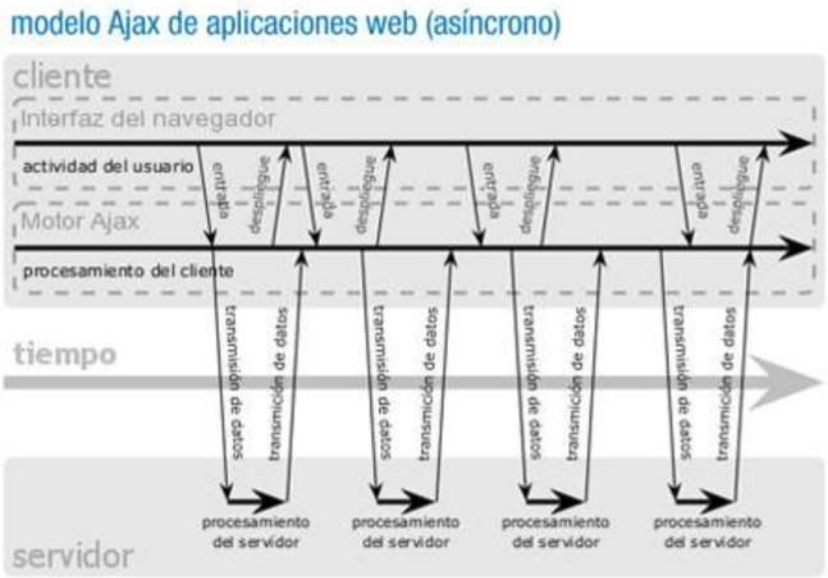
Lenguaje de servidor

- PHP, ASP, ...

Comunicación asíncrona



Comunicación asíncrona



El API XMLHttpRequest (XHR)

XMLHttpRequest (XHR):

- API de lenguajes de script del lado del cliente(ej. Javascript).
- Se utiliza en peticiones al servidor web (http o https).
- Los datos se pueden recibir en texto plano o XML y modificar el DOM sin modificar la página.
- Los datos también se pueden recibir en formato JSON y ser evaluados con Javascript.
- No es posible modificar páginas alojadas fuera del dominio desde el que se realiza la petición.

El API XMLHttpRequest (XHR)

Propiedades del objeto XMLHttpRequest (XHR):

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.

El API XMLHttpRequest (XHR)

Valor de la propiedad readyState:

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)

El API XMLHttpRequest (XHR)

Métodos del objeto XMLHttpRequest (XHR):

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

El API XMLHttpRequest (XHR)

Métodos del objeto XMLHttpRequest (XHR):

Método	Descripción
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 API XMLHttpRequest (XHR)

Métodos del objeto XMLHttpRequest (XHR):

El método open:

- **open(string metodo, string URL [,boolean asincrono, string usuario, string password]);**
- **Los dos primeros parámetros son obligatorios, los otros tres opcionales.**
- **Si se indica en el tercer parámetro "false", las peticiones se hacen síncronas.**
- **Realizar peticiones síncronas es contrario a la filosofía de Ajax.**

El método send:

- **En el parámetro se indica la información a enviar al servidor.**
- **Si se envían datos, puede ser una cadena, un array de bytes o un XML. Si no se envían datos, debe ser un valor null.**

El API XMLHttpRequest (XHR)

Ejemplos:

6_00_AJAX-HolaMundo.html
6_01_AJAX-HolaMundoMejorado.html

El API XMLHttpRequest (XHR)

Ejercicio U6T1 – Lector de ficheros:

Modifica el código de la página **6_01_AJAX-HolaMundoMejorado.html** para que haga lo siguiente:

- Crea un campo de texto en el que, nada más cargarse la página, se cargue la url de la misma.
- Crea un botón junto al campo de texto que se llame "Mostrar contenido" y que al hacer clic sobre él cargue en un textarea el contenido indicado en la url.
- Crea el textarea que inicialmente está vacío, pero que cargará el contenido de la url del campo de texto en él.

El API XMLHttpRequest (XHR)

Ejemplos:

6_01b_AJAX-fecha.html

El API XMLHttpRequest (XHR)

Ejercicio U6T2 - Localidad:

A partir de los códigos que hemos creado en clase deberás diseñar un programa que tenga las siguientes características:

- Una página con HTML que tenga un input de tipo texto y un botón: cuando el usuario introduzca el nombre de una localidad y pulse el botón obtendrá, en un div "resultado", un mensaje que indicará si la ciudad está incluida dentro de una lista de ciudades o no. El mensaje será rojo si no está incluida y verde en caso afirmativo.
- Una archivo en PHP que compruebe que la localidad recibida por parámetro está o no incluida dentro de una lista de 10 localidades (utiliza un array en PHP y recórrelo para comprobarlo).
- La petición debe realizarse de forma asíncrona, de modo que no se recargará la página, sino que se mostrará el resultado una vez finalizada la consulta al servidor.

Formatos para envío y recepción de la información

Recepción de datos en formato XML

- En la función iniciar() solicitamos que cargue, de manera asíncrona, el fichero PHP que devuelve los datos XML. Para ello usamos:
  - `cargarAsync("fichero.php");`
- Al recibir datos en formato XML debemos hacer la solicitud con la propiedad responseXML del objeto XHR.
  - `resultados = this.responseXML;`
- En la función estadoPetición realizamos el procesamiento del XML.

```
elementos = resultados.documentElement.getElementsByTagName("etiqueta_elementos");
for (i=0; i<elementos.length; i++)
```

- En caso de que haya elementos que puedan estar vacíos, debemos utilizar try- catch para evitar excepciones en Javascript.

Formatos para envío y recepción de la información

Ejercicio U6T3 - XML:

A partir de los ficheros que permiten procesar un XML, modifícalos de manera que el XML a procesar lo hayas creado tu mismo con los siguientes datos:

- Series: será el elemento principal del XML.
- Serie: contendrá los datos de una serie en concreto, que serán:
  - Título: nombre de la serie.
  - Cadena: nombre de la cadena que produce la serie (HBO, FX, etc.)
  - Director: nombre del director de la serie.
  - Año: año de estreno de la serie.
  - Terminada: podrá contener un valor "sí" o "no" en función si ha terminado o no su emisión.

Al procesar el XML se mostrarán todos los datos en una tabla. Tendrá las siguientes condiciones:

- El título, la cadena y el director: el título será negrita, y el director en cursiva.
- El año aparecerá en color rojo si la serie es anterior al año 2000, en amarillo si está entre el 2001 y el 2010 y en verde si es posterior al 2011. Estas variaciones se recogen en un archivo en CSS con reglas, como por ejemplo .rojo, .amarillo o .verde.
- En la celda "terminada" habrá una imagen determinada en caso de que en el XML se registre un Sí o un No.

Formatos para envío y recepción de la información

JSON

Formato para el intercambio de datos.

Alternativa a XML

Fácil uso en Javascript.

Puede ser leído por cualquier lenguaje de programación

Formatos para envío y recepción de la información

JSON

Nombre/Par de valores

```
"Nombre" : "Chema"
```

Valores

Número (entero o float)
String (entre comillas simples)
Booleano (true o false)
Array (entre corchetes)
Objeto (entre llaves)
Null

Objetos

```
{ "NombreFruta": "Manzana", "Cantidad": 20 }
```

Arrays

```
{
  "Frutas": [
    { "NombreFruta": "Manzana", "cantidad": 10 },
    { "NombreFruta": "Pera", "cantidad": 20 },
    { "NombreFruta": "Naranja", "cantidad": 30 }
  ]
}
```

Formatos para envío y recepción de la información

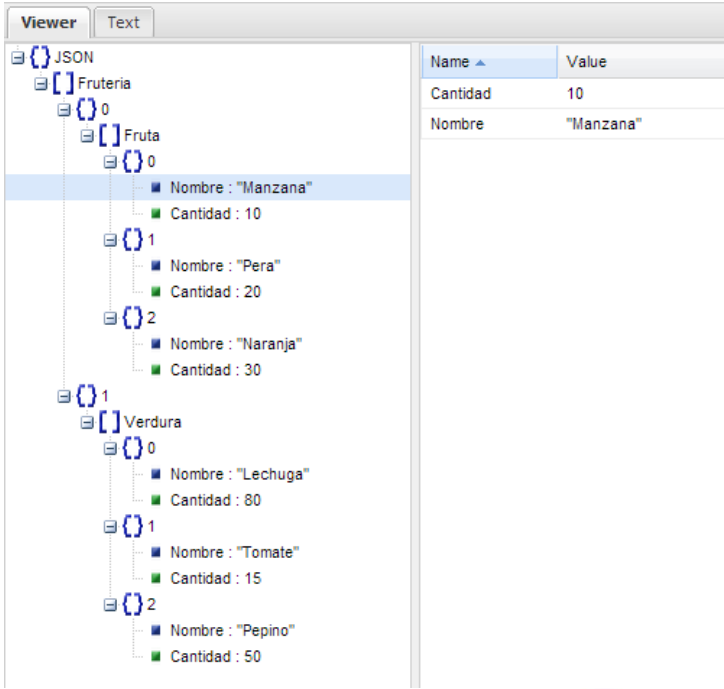
JSON

```
{
  "Fruteria": [
    {
      "Fruta": [
        { "Nombre": "Manzana", "Cantidad": 10 },
        { "Nombre": "Pera", "Cantidad": 20 },
        { "Nombre": "Naranja", "Cantidad": 30 }
      ]
    },
    {
      "Verdura": [
        { "Nombre": "Lechuga", "Cantidad": 80 },
        { "Nombre": "Tomate", "Cantidad": 15 },
        { "Nombre": "Pepino", "Cantidad": 50 }
      ]
    }
  ]
}
```

Formatos para envío y recepción de la información

JSON

```
{
  "Fruteria": [
    {
      "Fruta": [
        { "Nombre": "Manzana", "Cantidad": 10 },
        { "Nombre": "Pera", "Cantidad": 20 },
        { "Nombre": "Naranja", "Cantidad": 30 }
      ]
    },
    {
      "Verdura": [
        { "Nombre": "Lechuga", "Cantidad": 80 },
        { "Nombre": "Tomate", "Cantidad": 15 },
        { "Nombre": "Pepino", "Cantidad": 50 }
      ]
    }
  ]
}
```



JSON Viewer

Formatos para envío y recepción de la información

Recepción de datos en formato JSON

- En la función iniciar() solicitamos que cargue, de manera asíncrona, el fichero PHP que devuelve los datos XML. Para ello usamos:

```
cargarAsync("fichero.php");
```

- Al recibir datos en formato JSON debemos hacer la solicitud con la propiedad .responseText del objeto XHR y la parseamos con JSON.parse teniendo en cuenta que devolverá un array.

```
resultados = JSON.parse(this.responseText);
```

- En la función estadoPetición realizamos el procesamiento del JSON. Para el ejemplo de las frutas:

```
texto = "<table border=1><tr><th>Fruta</th><th>Cantidad</th></tr>";
let frutas = mijson[0].Fruta;
for (let i = 0; i < frutas.length; i++) {
  fruta = frutas[i];
  texto += "<tr><td>" + fruta.Nombre + "</td><td>" + fruta.Cantidad + "</td></tr>";
}
texto += "</table>";
```

- En caso de que haya elementos que puedan estar vacíos, debemos utilizar try-catch para evitar excepciones en Javascript.

Formatos para envío y recepción de la información

Ejercicio U6T4 – JSON:

Crea un archivo json equivalente al XML que creaste en el ejercicio anterior y comprueba su funcionamiento con el JSON Viewer (http://jsonviewer.stack.hu/).

Basándote en el ejemplo de la frutería, procesa el JSON de manera que obtengas el mismo resultado que obtuviste en el ejercicio anterior (la tabla de las series con las especificaciones dadas).

Recuerda que solamente tendrás que modificar los siguientes archivos:

- datosjson.php: donde modificarás el XML por el JSON.
- index.js: donde procesarás los datos teniendo en cuenta que se trata de un archivo JSON.

END

prof.jduran@iesalixar.org