DESARROLLO WEB EN ENTORNO CLIENTE

U.D. 6: Utilización de mecanismos de comunicación asíncrona

Objetivos

- Mecanismos de comunicación asíncrona en las aplicaciones Web.
- Tecnologías asociadas con la técnica AJAX.
- Formatos de envío y recepción de información asíncrona.
- Llamadas asíncronas.

Mecanismos de comunicación síncrona

- En un proceso habitual, el cliente es el que inicia el intercambio de información solicitando datos al servidor que responde enviando un flujo de datos al cliente.

- Así, **si** el usuario ingresa en una página Web introduciendo una URL en el navegador, **esperará la respuesta del servidor** hasta que el código HTML llegue por completo y se dibuje la página solicitada. En ese caso, se está utilizando un mecanismo de comunicación **síncrona**: el cliente ha enviado una petición y permanece bloqueado esperando la respuesta del receptor.

Mecanismos de comunicación asíncrona

- El mecanismo de comunicación **asíncrona** recarga en **segundo plano** una **parte** de la página Web, dejando **desbloqueado** el resto.

- El cliente que envía una petición no permanece bloqueado esperando la respuesta del servidor.

- Esto ayuda a que las aplicaciones Web tengan una interactividad similar a las aplicaciones de escritorio.

Mecanismos de comunicación asíncrona

- Ejemplos

```
<script>
    //Ejemplo acciones síncronas
    //ACCION 1
    console.log("Inicio del codigo síncrono");
    //ACCION 2
    for (let i=0; i<10; i++){
        console.log(i);
    }
    //ACCION 3
    console.log("Fin del codigo síncrono");


    //Ejemplo acciones asíncronas
    //ACCION 1
    console.log("Inicio del codigo síncrono");
    //ACCION 2 setTimeout es una función que funciona en segundo plano
    setTimeout(function(){
        for (let i=0; i<10; i++){
        console.log(i);
        }
    }, 0); // 0 sec de retraso en el inicio

    //ACCION 3
    console.log("Fin del codigo síncrono");
```

```
<script>
    //1.
    /*function avanzaFila(){
        setTimeout(function(){
            console.log("Tu turno ha llegado");
        }, 5000);
    }

    avanzaFila();*/

    //2. callback es una función que pasas como variable y la lanzas
    //en el transcurso de las instrucciones de la def de la funcion
    function avanzaFila(callback){
        setTimeout(function(){
            console.log("Tu turno ha llegado");
            callback();
        }, 5000);
    }

    function mujerTeLlama(){
        console.log("Te presentas a tu turno");
    }

    console.log("Llegas a la fila y haces el trato");
    avanzaFila(mujerTeLlama);
    console.log("Te vas a por café");
```

*Funciones *callback* en javascript

Mecanismos de comunicación asíncrona

- Funciones *callback* en javascript: es una función que se pasa a otra función como un argumento, que luego se invoca dentro de la función externa para completar algún tipo de rutina o acción.

```
function saludar(nombre) {
  alert("Hola " + nombre);
}

function procesarEntradaUsuario(callback) {
  var nombre = prompt("Por favor ingresa tu nombre.");
  callback(nombre);
}

procesarEntradaUsuario(saludar);
```

El ejemplo es una callback síncrona, ya que se ejecuta inmediatamente.

Las callbacks a menudo se utilizan para continuar con la ejecución del código después de que se haya completado una operación asíncrona — estas se denominan devoluciones de llamada asíncronas.

AJAX

- Necesidad: aplicaciones Web interactivas.
- Solución: nuevo uso a tecnologías como XML, CSS o DOM.
- En 2005 J.J. Garrett habla por primera vez sobre AJAX (*Asynchronous JavaScript And XML*): **JavaScript asíncrono y XML**.
- Se suprimen los efectos secundarios de las recargas, como la pérdida del contexto, la ubicación del *scroll* o las respuestas más lentas.

AJAX - Ventajas

- Ajax no es una función. Es una técnica de desarrollo de aplicaciones Web que permite la creación de aplicaciones interactivas.
- Se minimizan las comunicaciones entre el cliente y el servidor, realizándose de manera asíncrona.
- Mejora la experiencia de usuario.
- Alta compatibilidad. Soportado por la mayoría de plataformas Web.
- Permite el scroll "infinito".
- Los buscadores pueden sugerir palabras clave mientras se escribe la búsqueda.

AJAX

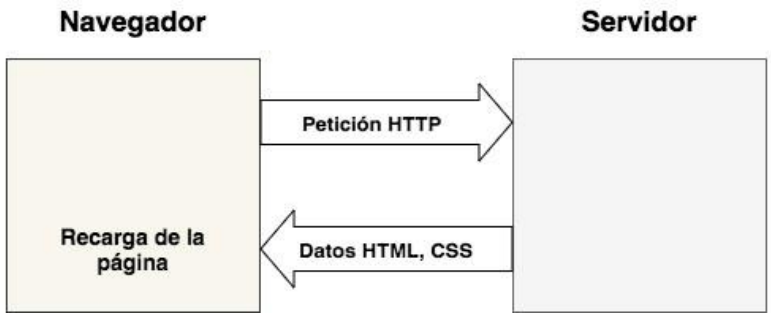
Ajax es una técnica de desarrollo que permite actualizar ciertas partes de una página web, mejorando la interactividad y la experiencia de usuario. Cuando estás navegando en una web tradicional y cambias de página, se producen las siguientes acciones:

1. Se envía una petición al servidor para obtener la nueva página.
2. El servidor gestiona la petición y envía la página (archivos HTML, CSS, imágenes, etc.).
3. Se muestra la información en la navegador.

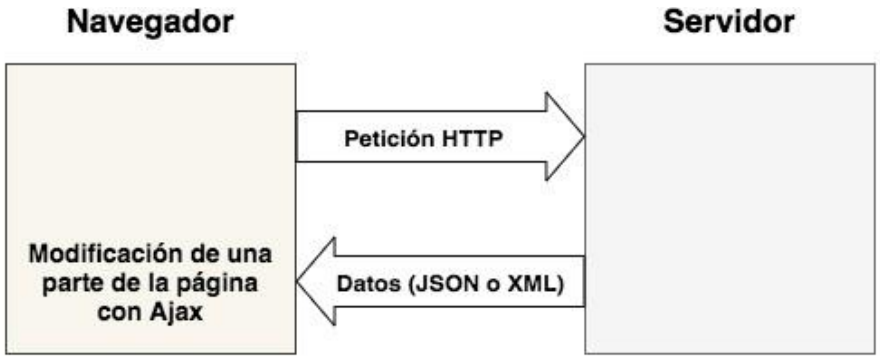
Esta ha sido la forma de funcionar desde el principio de la Web, pero con la aparición de Ajax, es posible recargar sólo una parte de la página y no interrumpir constantemente la navegación:

1. En primer lugar, se envía una petición al servidor con el objetivo de obtener los datos necesarios.
2. El servidor procesa los datos demandados y envía una respuesta.
3. Se reciben los datos y se muestran en una parte de la página, sin recargarla en su totalidad.

La figura siguiente muestra un resumen de los dos modos de funcionamiento:



AJAX



Los dos modos de funcionamiento de una web: tradicional y con Ajax

Aunque el funcionamiento tradicional sigue siendo válido en la mayoría de los casos, se dan muchas situaciones en las que el uso de Ajax es muy útil. Por ejemplo, se puede utilizar Ajax para validar datos de un formulario en tiempo real, para mostrar sugerencias como hace google en su buscador, o para cargar imágenes en una galería cuando el usuario se va desplazando hacia abajo.

Elección de AJAX - Inconvenientes

- Uso de nuevas tecnologías y mayor complejidad.
- Las aplicaciones o sitios Web utilizan más recursos del servidor.
- Problemas de SEO. Los buscadores tienen más dificultad al analizar el código.
- El comportamiento considerado lógico por el usuario al utilizar la funcionalidad de "volver a la página anterior" no es reproducido de la misma manera.
- AJAX es un medio, no un fin. No se recomienda en formularios simples, Webs básicas, consultas complejas a BBDD, etc.

Tecnologías en AJAX

- XHTML y CSS para una presentación basada en estándares.
- DOM para la interacción y la visualización dinámica de datos.
- XML y XSLT para el intercambio y transformación de datos entre cliente y servidor.
- XMLHttpRequest para la recuperación asíncrona de los datos entre cliente y servidor.
- JavaScript como elemento de unión del resto de tecnologías y marco de trabajo del lado del cliente.

Tecnologías involucradas: XHTML y CSS

- XHTML es un HTML estándar especificado mediante un documento XML. XHTML es más riguroso con su estructura:
  - Los valores de los atributos, siempre entre comillas:
    - Incorrecto: <td colspan=2>
    - Correcto: <td colspan="2">
  - Los nombres de elementos y atributos deben ir en minúsculas:
    - Incorrecto: <IMG SRC="dibujo.png" /IMG>
    - Correcto: <img src="dibujo.png" /img>
  - No está permitida la minimización de atributos (se usa el nombre del atributo como valor):
    - Incorrecto: <textarea readonly>
    - Correcto: <textarea readonly="readonly">

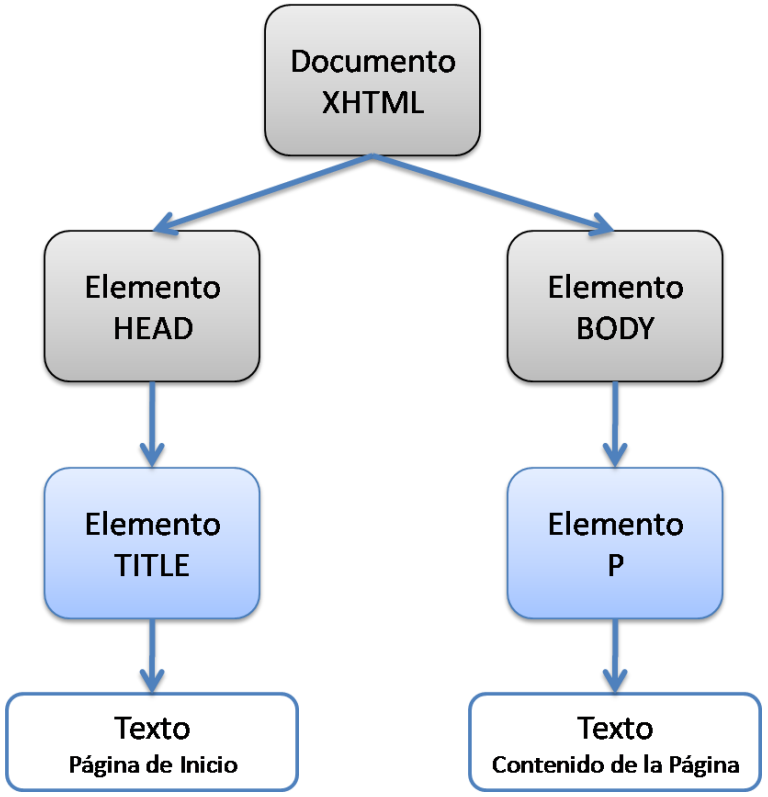
Tecnologías involucradas: XHTML y CSS

- Un cambio en el CSS es reflejado instantáneamente en todas las páginas que utilizan algún elemento de presentación identificado en dicha hoja de estilo.

- Si a esto le sumamos la capacidad de poder ser leído y modificado mediante el DOM, vemos la gran importancia que tiene en las aplicaciones con AJAX.

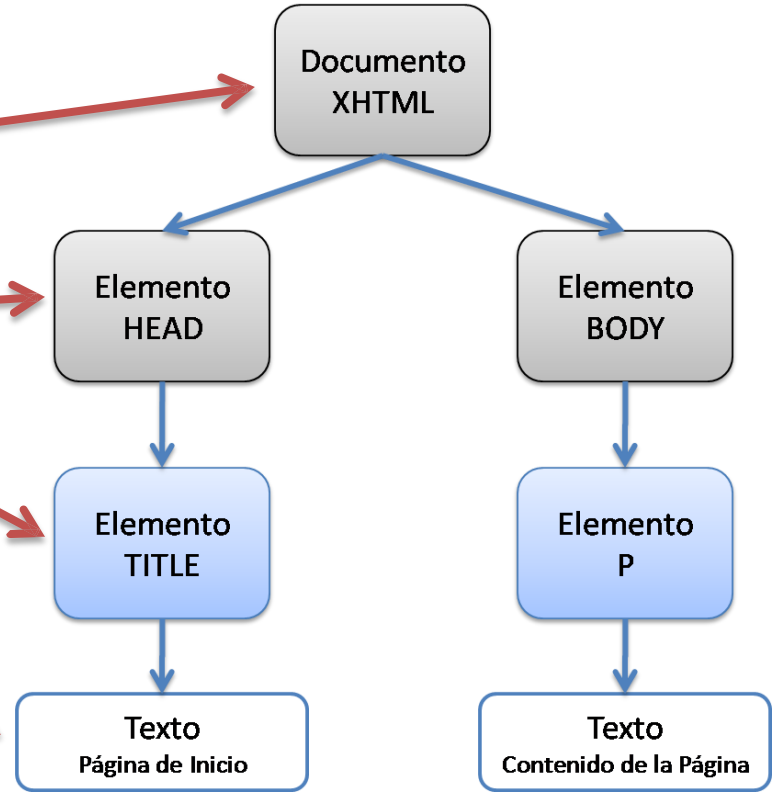
Tecnologías involucradas: DOM

- El *Document Object Model* (DOM) es una representación de la página Web en una estructura de jerarquía de árbol.
- Es esencialmente una API para representar documentos XHTML.
- Todas las partes de la página están accesibles desde el cliente y no es necesario utilizar tecnologías del lado del servidor.



Tecnologías involucradas: DOM

```
<html
   xmlns="http://www.w3.org/1999/xh
   tml">
 <head>
  <title>Página de Inicio</title>
 </head>
 <body>
  <p>Contenido de la Página</p>
 </body>
</html>
```



Tecnologías involucradas: DOM

La fortaleza de esta representación jerárquica está en su grado de estandarización y en que todos los navegadores lo utilicen. El DOM es un estándar del World Wide Web Consortium (también conocido como W3C www.w3.org), se considera un estándar que deben seguir todos los navegadores para representar las páginas Web. Esto garantiza que, modificando el color del borde de un botón mientras paso el cursor sobre el elemento, funcione de la misma manera tanto en Chrome como en Edge, Firefox, etc. Mediante el DOM y un lenguaje de script del lado del cliente (el más utilizado es JavaScript) podemos agregar o modificar elementos del árbol DOM con la particularidad de que aparecerán inmediatamente en el navegador.

Tecnologías involucradas: JavaScript

- El lenguaje de scripting JavaScript es el más utilizado actualmente en los navegadores.
- Es un lenguaje orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Tecnologías involucradas: JavaScript

- Utilización principal es en el lado del cliente.
- Permitiendo mejoras en la interfaz de usuario, páginas web dinámicas y accediendo al árbol DOM para realizar modificaciones instantáneas del código fuente.
- El código JavaScript se ejecuta en el propio navegador y puede ir formando parte del propio código HTML de dicha página.

Tecnologías involucradas: JavaScript
Ejemplo de JavaScript

```
<head>
  <script>
   function cambiarAltura() {
    if (novedades.style.height == "150px") {
      novedades.style.height = "25px";
    } else {
      novedades.style.height = "150px"; }
   }
  </script>
 </head>
 <body>
  <div id="novedades" style="height:150px;
   border:1px;">
  <a href="#" onClick="cambiarAltura()">X</a>
  <p>Alerta: La línea de bus 25 cambia su
   recorrido</p>
  </div>
 </body>
```

Tecnologías involucradas: JavaScript
Ejemplo de JavaScript

Analizando el código fuente anterior, tenemos una función JavaScript que al ser ejecutada realiza el cambio de tamaño del elemento *div*. Comprobamos mediante una sentencia condicional *if* el tamaño actual del elemento *div* cuyo identificador es "novedades". De hecho, modificamos directamente al atributo altura *height* del elemento novedades.

La sentencia "novedades.style.height" hace referencia al elemento *div* que tiene el atributo *id* con el valor "novedades", este *div* también tiene un atributo *style* y dentro de este atributo la propiedad *height* de CSS. Lo que hacemos aquí es cambiar esta propiedad CSS.

Si analizamos el resto del código se puede ver el mecanismo de llamada de la función JavaScript. Dentro del elemento *div* tenemos un enlace, que dispara la función JavaScript utilizando el atributo *onclick.*

Tecnologías involucradas: XML

- El lenguaje XML es utilizado para describir y estructurar datos.
- Alrededor de XML encontramos otras tecnologías como Xquery y XSLT.
- Los navegadores contienen funcionalidades internas para trabajar con documentos XML.

Tecnologías involucradas: XML

- XQuery proporciona los medios para extraer y manipular información de documentos XML, o de cualquier fuente de datos que pueda ser representada mediante XML como, por ejemplo, bases de datos relacionales o documentos ofimáticos.

- XSLT (EXtensible Stylesheet Language Transformations) es un lenguaje para transformar un documento XML en otro documento XML, texto plano, etc.

Tecnologías involucradas: XML
Ejemplo de XML

```
<ciudadano>
  <nombre>Pepe</nombre>
  <edad>34</edad>
  <domicilio>calle Alcalá 1</domicilio>
  <estudios>
    <estudio>primario</estudio>
    <estudio>secundario</estudio>
    <estudio>universitario</estudio>
  </estudios>
</ciudadano>
```

Tecnologías involucradas: JSON

- JSON (JavaScript Object Notation).
- Formato ligero para el intercambio de datos.
- Es una manera de almacenar información.
- Fue pensado en un primer momento como una alternativa a XML.

Tecnologías involucradas: JSON

- XML tienen una gran cantidad de información extra asociada a su estructura.

- En cambio, JSON está constituido por dos estructuras:

  - Una colección de pares de nombre-valor.
  - Una lista ordenada de valores.

- Para obtener datos estructurados desde el servidor que se van a transformar en diferentes formatos (fichero PDF, HTML, etc.) se recomienda usar JSON.

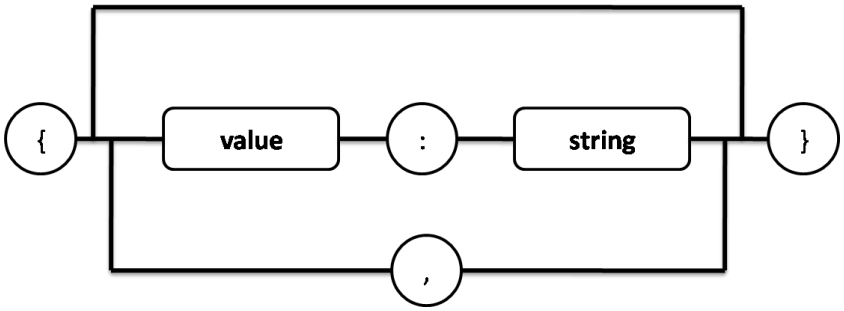
Tecnologías involucradas: JSON Object

- El elemento base de la sintaxis es el *object.*

- Está conformado por un conjunto desordenado de pares nombre-valor.

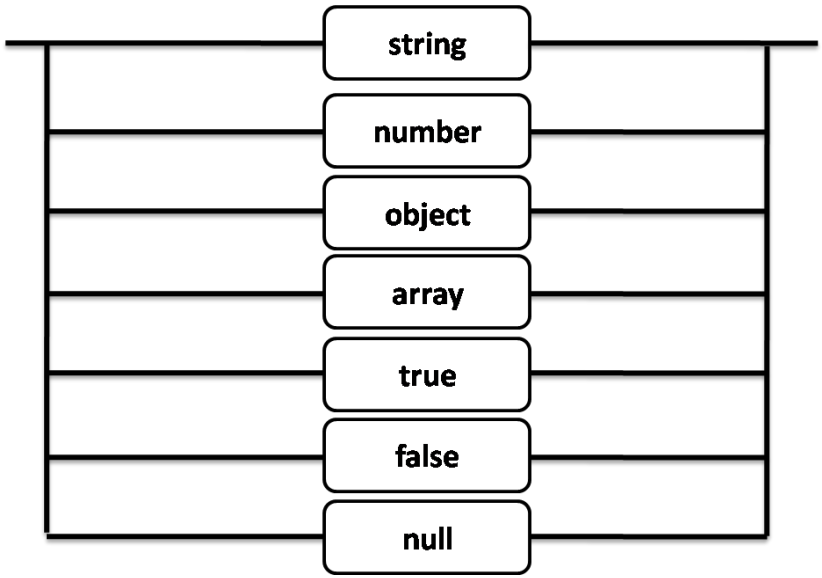
- Un objeto comienza con una llave de apertura y finaliza con una llave de cierre.

- Cada nombre es seguido por dos puntos, estando los pares nombre-valor separados por una coma.



Tecnologías involucradas: JSON Array

- Un *array* es una colección de elementos *values.*
- Comienza por un corchete izquierdo y termina con un corchete derecho.
- Los elementos *value* se separan por una coma.



Tecnologías involucradas: JSON String y Number

- Un *string* es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape.

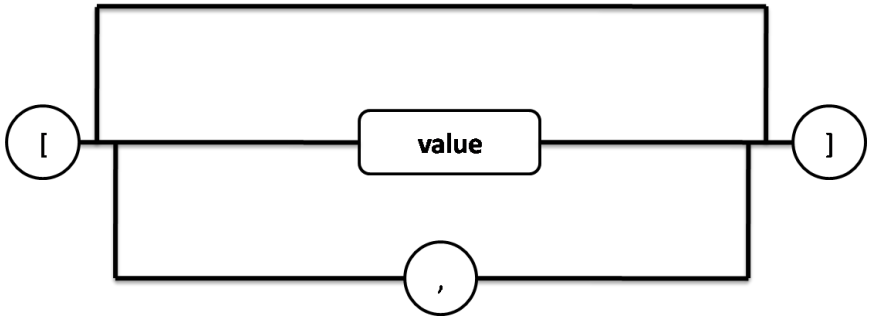
- Una cadena de caracteres es parecida a una cadena de caracteres de C o Java.

- Un elemento *number* es similar a un número de C o Java, excepto que no se usan los formatos octales y hexadecimales.

- Por último, los espacios en blanco pueden insertarse entre cualquier par de símbolos.

Tecnologías involucradas: JSON Value

- A su vez un elemento *value* puede ser una cadena de caracteres o *string* con comillas dobles, un *number*, los valores booleanos *true* o *false*, *null*, un *object* o un *array.*



- Estas estructuras pueden anidarse.

Tecnologías involucradas: JSON
Ejemplo de JSON

```
{
   'nombre':'Pepe',
   'edad':34,
   'domicilio':'calle Alcalá 1',
   'estudios':['primario','secundario','universitario']
}
```

Tecnologías involucradas: Objeto XMLHttpRequest

- Aparece a partir de Internet Explorer 5 en la forma de un control ActiveX llamado XMLHttp.

- Se fueron transformando en un estándar de facto en navegadores como Firefox, Safari, Opera, etc.

- Actualmente el objeto **XMLHttpRequest** se encuentra descrito por el World Wide Web Consortium y sirve como una interfaz con la que se realizan peticiones a servidores Web.

Tecnologías involucradas: Atributos del objeto XMLHttpRequest

Atributo	Descripción
readyState	Devuelve el estado del objeto como sigue: 0 = sin inicializar, 1 = abierto, 2 = cabeceras recibidas, 3 = cargando y **4** = completado.
responseBody	Devuelve la respuesta como un array de bytes.
responseText	Devuelve la respuesta como una cadena.
responseXML	Devuelve la respuesta como XML. Esta propiedad devuelve un objeto documento XML, que puede ser examinado usando las propiedades y métodos del árbol DOM.
status	Devuelve el estado como un número (ej. **200** = "OK", 404 = "Not Found").
statusText	Devuelve el estado como una cadena (ej. "Not Found").

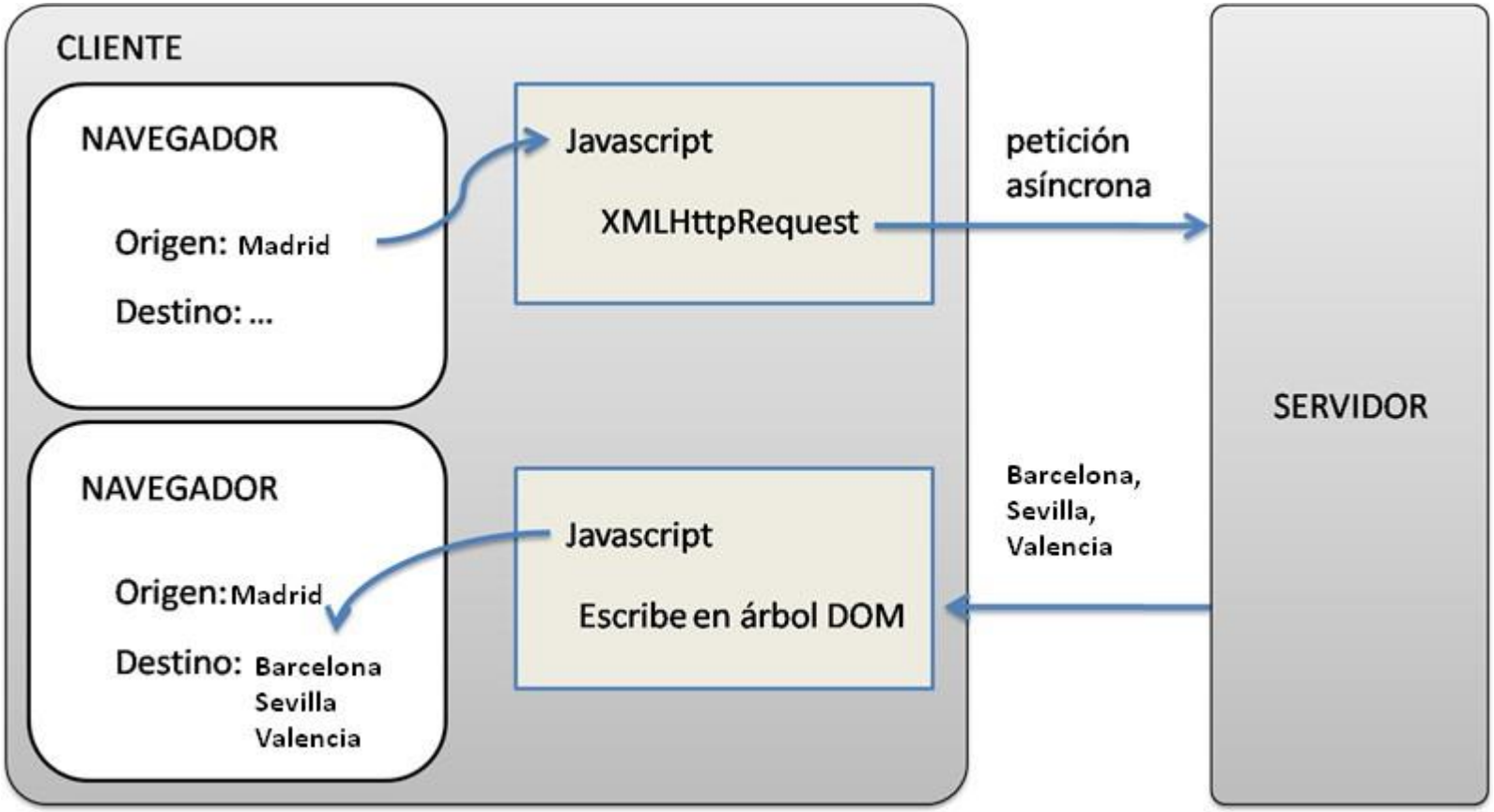
Tecnologías involucradas: Métodos del objeto XMLHttpRequest

Métodos	Descripción
`abort()`	Cancela la petición en curso.
`getAllResponseHeaders()`	Devuelve el conjunto de cabeceras HTTP como una cadena.
`getResponseHeader(cabecera)`	Devuelve el valor de la cabecera HTTP especificada.
`open` `( método, URL [, asíncrono [, nombreUsuario [, clave]]] )`	Especifica el método, URL y otros atributos opcionales de una petición. El parámetro de **método** puede tomar los valores "GET", "POST" o "PUT". El parámetro **URL** puede ser una URL relativa o completa. El parámetro **asíncrono** especifica si la petición será gestionada asíncronamente o no.
`send``([datos])`	Envía la petición.

Tecnologías involucradas: Propiedades del objeto XMLHttpRequest

Propiedades	Descripción
`onreadystatechange`	Evento que se dispara con cada cambio de estado.
`onabort`	Evento que se dispara al abortar la operación.
`onload`	Evento que se dispara al completar la carga.
`onloadstart`	Evento que se dispara al iniciar la carga.
`onprogress`	Evento que se dispara periódicamente con información de estado.

Perspectiva global de AJAX



Perspectiva global de AJAX

- La página recibida por el cliente contiene código JavaScript, el cuál puede obtener datos del servidor. El navegador, al recibir la página Web crea el DOM asociado el cuál puede ser accedido y modificado por el código JavaScript.

- Cuando la página necesita información del servidor (por ejemplo, cuando se ha elegido el origen de un viaje y es necesario el listado de destinos posibles), desde el lenguaje JavaScript se utiliza un elemento especial: el objeto **XMLHttpRequest**.

- A través de este elemento se envía una petición al servidor **sin provocar que la página sea recargada**. Esta es la clave, la ejecución de JavaScript en el cliente dispara una comunicación asíncrona, no bloquea la página hasta la recepción de los datos (en el caso del ejemplo, los destinos posibles para el origen elegido) y el usuario puede seguir trabajando en la página.

- Los datos obtenidos del servidor son, por lo general, texto plano u objetos XML que será leído por el código JavaScript.

Ejemplo de AJAX

- La página Web muestra un botón, cuando hacemos pulsamos sobre él, muestra un mensaje en un elemento "div" cambiando el texto que se encontraba anteriormente.

```
<form>
  <input type="button" value="Buscar información"
   onclick="obtenerDatosServidor('http://web/datos.txt',
     'elemento_destino')" />
</form>
<div id="elemento_destino">
  <p>La información aparecerá aquí</p>
</div>
```

Ejemplo de AJAX

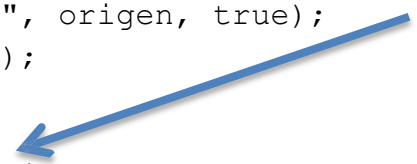
1. Función "obtenerDatosServidor" contiene dos parámetros.
2. Se elige el elemento HTML a ser modificado.
3. Se indica la función a ser llamada una vez el estado del objeto cambie.
4. Se configura una conexión asíncrona con una URL.
5. Se abre la conexión.

```
<script>

    var objetoXHR = new XMLHttpRequest();

1   function obtenerDatosServidor(origen, elemento)
    {
2    var objeto_destino = document.getElementById(elemento);
3    objetoXHR.onreadystatechange = respuesta();
4    objetoXHR.open("GET", origen, true);
5    objetoXHR.send(null);
    }

    function respuesta() {
     if (objetoXHR.readyState == 4 &&
      objetoXHR.status == 200) {
      objeto_destino.innerHTML = objetoXHR.responseText;
     }
    }
</script>
```



AJAX

- Ejemplo: Crea una aplicación que reciba datos de un archivo JSON haciendo uso de la tecnología AJAX.

```
//2. Llamada a JSON
let datosJson;
//Objeto xmlhttprequest
let xhr = new XMLHttpRequest();

//funcion open, orden Get, url del json, y peticion asícrona, la pag no se va a bloquear mientras
xhr.open('GET', "persona.json", true);
//ponemos qué tipo de respuesta se espera
xhr.responseType = 'json';

//onload se desencadena cuando la solicitud HTML http request ha sido completada exitosamente
//es decir que los datos ya están disponibles para ser procesados
xhr.onload= function (){
    //el estado http 200 significa que la solicitud se ha completado exitosamente
    if(xhr.status === 200){
        datosJson = xhr.response;
        let elementoTexto = document.getElementById("nombre");
        elementoTexto.textContent = datosJson.nombre;
    }else{
        //manejar el error
    }

}
//tras la configuración, hay que enviar el objeto
xhr.send();
```

AJAX

- Ejemplo: Crea una aplicación que reciba datos de un archivo XML haciendo uso de la tecnología AJAX.

```
$.ajax({
    type: "GET",
    url: "./demo.xml",
    dataType: "xml",

    error: function (e) {
        alert("Error al procesar el fichero XML");
        console.log("XML reading Failed: ", e);
    },

    success: function (response) {

        // eliminamos el contenido previo de la ul
        //así, si llamamos de nuevo, será el contenido del xml
        $("ul").children().remove();

        $(response).find("food").each(function () {
            var _name = 'Nombre: ' + $(this).find('name').text();
            console.log(_name);

            var _price = 'Precio: ' + $(this).find('price').text();
            var _calories = 'Calorias: ' + $(this).find('calories').text();
            var _description = 'Descripción: ' + $(this).find('description').text();

            // añadimos contenido al HTML
            $("ul").append('<li>' + _name + '</li>');
            $("ul").append('<li>' + _price + '</li>');
            $("ul").append('<li>' + _calories + '</li>');
            $("ul").append('<li>' + _description + '</li>');
            $("ul").append('<hr>');
        });
    }
```

Promesas en JavaScript

- Mismo ejemplo de JSON pero con fetch

```
let datosJson;

//fetch y la url de la dirección de archivo json
//Devolverá una promesa
fetch('persona.json')
  .then(res => res.json())
  //then, una vez que se haya resuelto la promesa, se ejecutará
  // ejecutamos una función anónima y en la variable res guardamos
  //res es la respuesta del servidor que proviene de la solicitud realizada por fetch

  //json() es un método que recibe como parámetro res y devuelve
  //con un retorno interno el resultado de parsear la respuesta del servidor
  //a un objeto JSON

  //dentro de fetch
  //vamos a cargar la salida una vez haya terminado lo anterior, de ahí then
  .then((salida)=>{
      datosJson = salida;

      let elementoTexto= document.getElementById("nombre");
      elementoTexto.textContent = datosJson.nombre;

  })

  //promesa, cumplida, rechazada o pendiente
  //recibes un objeto con ello

  //Peticiones http
  //GET - obtener info servidor
  //POST - enviar info servidor
  //PUT - actualizar info servidor
  //DELETE - borrar info servidor

  //Respuesta http . Codigos de estado:
  //200 - respuesta completada
  //201 - elemento creado
  //204 - respuesta vacía
  //400 - mal solicitado
  //401 - no autorizado
```

```
let datosJson;

//fetch y la url de la dirección de archivo json
//Devolverá una promesa
fetch('persona.json')
  .then(res => res.json())

  .then((salida)=>{
      datosJson = salida;

      let elementoTexto= document.getElementById("nombre");
      elementoTexto.textContent = datosJson.nombre;

  })
  .catch(function(error){
      //pasamos la información que devuele y la guardamos en error
      alert(error);
  })
```

Promesas en JavaScript

Explora estos mismo ejemplos empleando los distintos métodos de petición asíncrona de datos (XMLHttpRequest, jQuery, fetch)

Promesas en JavaScript

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

- Fetch

Promesas en JavaScript

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Using_promises

- Promesa: Una Promise es un objeto que representa la terminación o el fracaso de una operación asíncrona.

U.D. 6 (parte 2/2):
Librería jQuery

Objetivos

- Librerías de JavaScript.
- Librerías de actualización dinámica.
- jQuery.

Librerías útiles de JavaScript

- Las librerías o bibliotecas de JavaScript contienen funciones, métodos u objetos para realizar tareas específicas en una página o aplicación Web con el fin de ahorrar tiempo y esfuerzo.

- Para utilizar una librería en una aplicación, generalmente basta con añadir <script> al elemento <head> utilizando el atributo "src" que hace referencia a la ruta de origen o URL de la librería.

- Ejemplos: documento "Librerías de JavaScript"
  https://kinsta.com/es/blog/bibliotecas-javascript/#qu-son-las-bibliotecas-javascript

Librerías de actualización dinámica

- Junto con la tecnología AJAX, están las librerías que implementan una gran cantidad de funciones y controles:
  - Independiente de la tecnología del servidor (ej. PHP, etc.).
  - Manejar de manera transparente las incompatibilidades de los diferentes navegadores.
  - Manejar la comunicación asíncrona, sin necesidad de realizar la gestión de las operaciones de bajo nivel, como por ejemplo el manejo de estados y de tipos de errores.
  - Acceso sencillo al árbol DOM.
  - Información de errores para facilitar su utilización al desarrollador.
  - Proporcionar controles y objetos gráficos configurables, como por ejemplo botones, calendarios, campos de texto, etc.

Librería jQuery

- Es una librería de funciones clásica de JavaScript (creada en 2006) con licencia GPL que permite:
  - Selección y manipulación de elementos HTML y CSS.
  - Funciones de eventos en HTML.
  - Efectos y animaciones. Soporta efectos complejos como arrastrar y soltar.
  - Soporte de temas y de manejo directo con el teclado.
  - Funciones de búsqueda.
  - Edición en el mismo control.
  - Soporta la tecnología AJAX.
  - Compatibilidad con todos los navegadores.
  - "Write less, do more" (escribe menos, haz más).

Selector jQuery

- La forma básica de interactuar es mediante la función $() que recibe como parámetro el identificador de un elemento HTML o el nombre de una etiqueta HTML.
  - Para utilizar la librería:

```
<script src="jquery-3.6.1.min.js"></script>
```

  - Sintaxis: $(selector).action()
    - $: Símbolo para definir jQuery.
    - selector: Consulta sobre elementos HTML.
    - action: Acción que ejecuta sobre los elementos.
  - Ejemplo: Momento de ejecución de código JS cuando se carga una página.

```
window.onload = function() { /*Código JS*/ };
$(document).ready(function() { /*Código jQuery*/ });
```

jQuery

- Ejemplos de uso:

$("p") // Se seleccionan todos los elementos de tipo párrafo.

$("p.intro") // Todos los párrafos con class=intro.

$("p#demo") // Todos los párrafos con id=demo.

$("[href]") // Todos los elementos con atributo href.

$("[href='#']") // Todos los elementos con atributo href="#".

$("[href!='#']") // Todos los elementos con atributo href diferente de #.

$("[href$='.jpg']") // Todos los elementos con atributo href que acabe en .jpg.

$("p").css("background-color","yellow"); // Se modifica el color de fondo de todos los párrafos a amarillo.

$("p#intro:first") // El primer párrafo con id="intro".

$("ul li:first") // El primer elemento <li> de cada <ul>.

$("div#intro.head") // Todos los elementos con class="head" dentro de un con id="intro".

jQuery

- Ejemplos de uso:

$(selector).html(contenido) // Cambia el contenido de un elemento HTML.

$(#boton).fadeOut() // Se agrega un efecto visual al elemento con identificador "boton".

$(this).hide() // Oculta el elemento actual.

$(this).toggle() // Conmuta entre ocultar y mostrar el elemento.

$("input[name='nombre']").val() // Se obtiene el valor del elemento <input> de name nombre.

$(document).ready(function(){ }) // Se ejecuta la función cuando se cargue la página Web.

$(input:button).click(function(){ }) // Se ejecuta la función al pulsar en el elemento <input> de tipo button.

$.ajax({ }) // Método para trabajar con AJAX.

jQuery

- Ejemplo: Crea una página Web que muestre un párrafo y se oculte al pulsar sobre un botón haciendo uso del selector jQuery.

- Ejemplo: Continuando con el ejemplo anterior, al pulsar en el botón, el párrafo debe desaparecer. Pero si se vuelve a pulsar, debe aparecer de nuevo.

jQuery

- Ejemplo: Crea una aplicación que envíe y reciba datos de un archivo PHP haciendo uso de la tecnología AJAX y del selector jQuery.

- Ejemplo: Crea una página Web que muestre información de una API REST del Ayuntamiento de Zaragoza, a la que se conecte a través del método AJAX y haciendo uso del selector jQuery.

Ejercicios

- Realiza todos los ejercicios del bloque 'Ejercicios 6.1'.

- Realiza todos los ejercicios del bloque 'Ejercicios 6.6'.