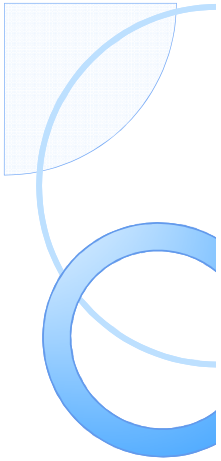
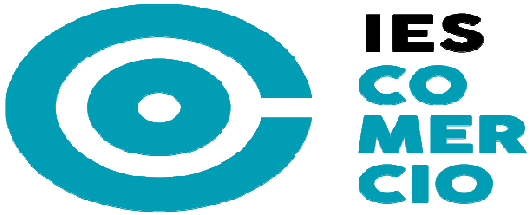




Unidad 8: Utilización de mecanismos de comunicación asíncrona. AJAX

Desarrollo Web en Entorno Cliente

Mª Luz Ancín

Curso: 2019-20

Objetivos

- Conocer las tecnologías asociadas con la técnica AJAX y su utilización en el desarrollo de aplicaciones interactivas.
- Identificar los formatos de envío y recepción de información asíncrona.
- Conocer en detalle la realización de llamadas asíncronas.
- Describir las librerías de actualización dinámicas actuales.

Contenidos

1. Mecanismos de comunicación asíncrona.
2. Ejemplo de comunicación asíncrona.
3. Recepción de información:

   XML y JSON.
4. Librerías de actualización dinámica.

1.- MECANISMOS DE COMUNICACIÓN SÍNCRONA

- Si el usuario entra 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.
- Mediante mecanismos de comunicación **asíncrona**, se recarga en **segundo plano** una **parte** de la página web, dejando **desbloqueado** el resto.

1.- MECANISMOS DE COMUNICACIÓN SÍNCRONA

- 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.
- AJAX (Asynchronous JavaScript And XML), es una técnica de programación que tiene como objetivo intercambiar pequeñas cantidades de datos entre el cliente y el servidor en segundo plano, recargando partes de la página web sin la necesidad de recargar todo el contenido de la misma.

1.1.- AJAX. Tecnologías

"Ajax no es una tecnología en sí mismo. En realidad, se trata de un conjunto de varias tecnologías independientes que se unen para configurar esta técnica de desarrollo de aplicaciones web"

Las tecnologías que forman AJAX son:

- **XHTML** y **CSS**, para crear una presentación basada en estándares.
- **DOM** y **JavaScript**, para la interacción y visualización dinámica de la presentación.
- **XML, XSLT** y **JSON**, para el intercambio y la manipulación de información.
- **XMLHttpRequest**, para el intercambio asíncrono de información.
- **PHP**, para recoger los datos.

1.1.- AJAX. Tecnologías

Ajax es utilizado en aplicaciones como Gmail, Google, Google Maps, Flickr, etc.

Razones para utilizar Ajax:

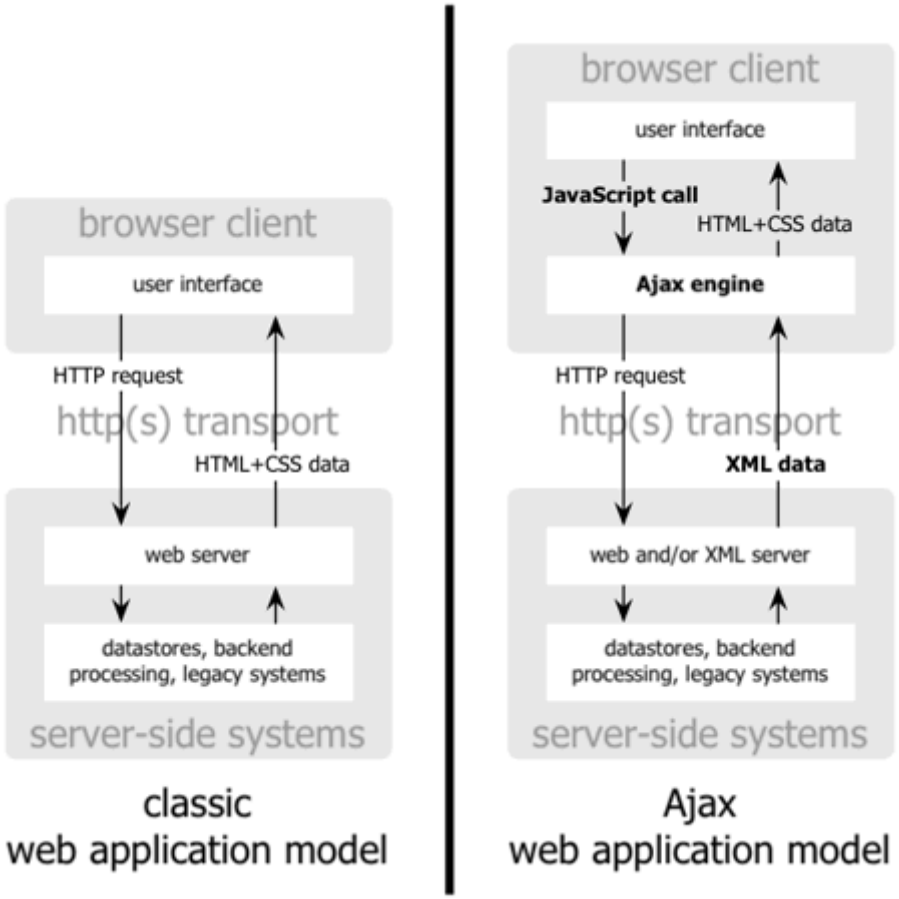
- Basado en estándares abiertos.
- Su usabilidad.
- Válido para cualquier plataforma y navegador.
- Beneficios que aporta a las aplicaciones web.
- Compatible con Flash..
- Es la base de la web 2,0
- Es independiente del tipo de tecnología de servidor utilizada.
- Mejora la estética de la web.

1.1.- Requerimientos previos

- Para la programación con Ajax vamos a necesitar de un servidor web, ya que las peticiones Ajax que hagamos, las haremos a un servidor. Los componentes que necesitamos son:
  - Servidor web (apache, ligHTTPd, IIS, …)
  - Servidor de base de datos (MySQL, Postgresql, …)
  - Lenguaje de servidor (PHP, ASP, …)
- Utilizaremos alguna aplicación que agrupa los servicios de servidor como.
  - Servidor LAMP (Linux, Apache, MySQL y PHP).
  - Servidor WAMP (Windows, Apache, MySQL y PHP).
  - Servidor XAMPP (tanto para Windows como para Linux).

1.2.- Como funciona Ajax

- Comparación gráfica del modelo tradicional de aplicaciones web y del nuevo modelo empleando Ajax:



1.3.- El objeto XMLHttpRequest

- Es la raíz de Ajax, ya que sin este objeto, no sería posible realizar las peticiones asíncronas al servidor.
- Se utiliza para realizar peticiones, http o https, directamente al servidor web, y para cargar las respuestas directamente en la página del cliente.
- Los datos que recibamos del servidor (texto plano, XML, JSON) podrán ser utilizados para modificar el DOM del documento actual sin tener que recargar la página.
- Una de las limitaciones es que por seguridad, solo nos deja realizar peticiones Ajax a las páginas que se encuentran hospedadas en el mismo dominio desde el que se esta realizando la petición Ajax.

Función cross-browser para crear objetos XMLHttpRequest

```
function objetoXHR() {
  if (window.XMLHTTPRequest){ // El navegador implementa la interfaz XHR de forma nativa
      return new XMLHTTPRequest();
  }else if (window.ActiveXObject){
     var versionesIE = new Array('MsXML2.XMLHTTP.5.0', 'MsXML2.XMLHTTP.4.0',
                'MsXML2.XMLHTTP.3.0', 'MsXML2.XMLHTTP', 'Microsoft.XMLHTTP');
     for (var i = 0; i < versionesIE.length; i++){
         try{
                  /* Se intenta crear el objeto en Internet Explorer comenzando
                  en la versión más moderna del objeto hasta la primera versión.
                  En el momento que se consiga crear el objeto, saldrá del bucle
                  devolviendo el nuevo objeto creado. */
                 return new ActiveXObject(versionesIE[i]);
             } catch (errorControlado) {} //Capturamos el error
          }
     }
      /* Si llegamos aquí es porque el navegador no posee ninguna forma de crear el objeto.
      Emitimos un mensaje de error usando el objeto Error.
      */
   throw new Error("No se pudo crear el objeto XMLHTTPRequest");
}

  // para crear un objeto XHR lo podremos hacer con la siguiente llamada.
var objetoAJAX = new objetoXHR();
```

1.3.- El objeto XMLHttpRequest

Propiedades	Descripción
`readyState`	Devuelve un número que indica el estado de la carga: **0** solicitud no iniciada. **1** conexión establecida con el servidor. **2** solicitud recibida. **3** procesando solicitud. 4 solicitud terminada y la respuesta disponible.
`responseBody`	Devuelve la respuesta como un array de bytes.
`responseText`	Devuelve la respuesta como una cadena de texto.
`responseXML`	Devuelve la respuesta como un archivo XML.

1.3.- El objeto XMLHttpRequest

Propiedades	Descripción
`status`	Devuelve el código de estado de la URL solicitada: **200** acceso correcto a la pagina **404** solicitud incorrecta, no se encuentra la página. **500** error interno que no permite ver la pagina. Wikipedia-códigos de error
`statusText`	Devuelve le código de estdo de la URL solicitado, como una cadena de texto: “OK”, Not Found” “Internal Server Error”, etc

1.3.- El objeto XMLHttpRequest

Métodos	Descripción
`abort()`	Cancela la petición en curso
`getAllResponseHeaders()`	Devuelve una cadena de texto con todas las cabeceras HTTP de la respuesta del servidor.
`getResponseHeader ("cabecera")`	Devuelve una cadena de texto con el contenido de la cabecera solicitada.
`open ( "método", "URL" [,"async" [,"usuario" [, "password"]]])`	Especifica el método, URL y otros atributos opcionales de la petición: El parámetro de **método** puede tomar los valores "GET", "POST". El parámetro **URL** puede ser una URL relativa o completa. El parámetro **asíncrono** especifica si la petición será gestionada asíncronamente o no (true/false).
`send([datos])`	Envía la petición al servidor HTTP.
`setRequestHeader(cabecera, valor)`	Permite establecer cabeceras personalizadas en la petición HTTP. Se debe invocar antes el método open()

1.3.- El objeto XMLHttpRequest

Eventos	Descripción
`onreadystatechange`	Se produce cada vez que hay un cambio de estado de carga del documento, se usa para saber si el documento ha sido cargado completamente.
`onabort`	Se produce cada vez que hay un cambio de estado de carga del documento, normalmente mediante el método `abort()`
`onload`	Se produce cuando el documento se ha cargado completamente.
`onloadstart`	Se produce cuando comienza la carga del documento.
`onprogress`	Se produce mientras el documento se está cargando.

1.4.- Crear un objeto XMLHttpRequest

- Todas las aplicaciones realizadas con técnicas de AJAX deben instanciar en primer lugar el objeto XMLHttpRequest.
- Cualquier aplicación AJAX se compone de cuatro grandes bloques:
  1. Instanciar el objeto XMLHttpRequest
  2. Preparar la función de respuesta
  3. Realizar la petición al servidor
  4. Ejecutar la función de respuesta.

1.4.- Crear un objeto XMLHttpRequest

Crear un objeto Ajax:

```
function ObjetoAjaX()
{   // Recogemos el objeto XMLHttpRequest en una variable
    var nuevoajax=crearObjetoAjax();
   // Devolvemos el objeto como una propiedad
    this.objeto=nuevoajax;
}

fucntion crearObjetoAjax()
{
    var obj; //variable que recogerá el objeto
    if(window.XMLHttpRequest) {
   // Navegadores que siguen W3C
       obj= new XMLHttpRequest();
       }
      else if(window.ActiveXObject) {
        // Navegadores obsoletos
        obj= new ActiveXObject("Microsoft.XMLHTTP");
        }
  return obj;  //devolvemos el objeto
}
```

Este código lo guardamos en un archivo aparte para poderlo reutilizar. Por ejemplo **objetoAjax.js**

1.4.- Crear un objeto XMLHttpRequest

1. Instanciar el objeto:

Ahora lo primero que haremos será insertar el archivo js

```
<script type="text/javascript" src="objetoAjax.js">
```

Y fuego en el código JavaScript instanciamos el objeto y llamamos a su propiedad .objeto, en la cual recogeremos el XMLHttpRequest.

```
pagina = new objwroAjax();
objetoReuest = pagina.objeto;
```

1.4.- Crear un objeto XMLHttpRequest

2. Preparar la función de respuesta:

```
peticion_http.onreadystatechange = respuesta;
```

3. Realizar la petición al servidor:

```
peticion_http.open('metodo','url',async,u,p);
```

- **metodo**: GET, POST
- **url**: La dirección del fichero al que le enviamos las peticiones en el servidor
- **async**: true (asíncrona) o false (síncrona).
- **u** y **p** : usuario y password si fuese necesaria la autenticación en él (parámetros opcionales).

1.4.- Crear un objeto XMLHttpRequest

3. Realizar la petición al servidor:
   - Para el método GET:

```
peticion_http.open('GET','prueba.txt',true);
peticion_http.send(null);
//en el caso de tener que enviar parámetros se haría junto con
la dirección url
```

   - Para el método POST:

```
peticion_http.open('POST','prueba.txt',true);
//creamos la cabecera para enviar los datos formando parte de
ésta.
peticion_http.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
peticion_http.send(parametros);
```

1.4.- Crear un objeto XMLHttpRequest

4. Ejecutar la función de respuesta.

```
function respuesta(){
 if(peticion_http.readyState == 4 {
 //si se han recibido todos los datos
  if (peticion_http.status == 200) {
  //si la respuesta del servidor es
   correcta (OK)
    alert(peticion_http.responseText);
  }
 }
}
```

1.- Primera aplicación AJAX

- La aplicación Ajax completa más sencilla consiste en una adaptación del clásico "Hola Mundo".
- En este caso la aplicación JavaScript descarga el contenido de un archivo llamado *holamundo.txt* que se encuentra en el servidor y muestra su contenido sin necesidad de recargar la página.

Ejemplo de AJAX

- La página Web muestra un botón que al hacer click 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 configura una conexión asíncrona con una URL.
4. Se indica la función a ser llamada una vez el estado del objeto cambie.
5. Se abre la conexión.

```
<script language = "javascript">

    var objetoXHR = new XMLHttpRequest();

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

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

Actividad 1

- Crea un script que compruebe con AJAX y el servidor si el nombre escogido por el usuario está libre o no.
- El script de servidor se llamará `compruebaDisponibilidad.php` y el parámetro que contiene el nombre se llama `login`.
- La respuesta del servidor es "`si`" o "`no`", en función de si el nombre de usuario está libre o ya ha sido ocupado por otro usuario (será una respuesta aleatoria)
- A partir de la respuesta del servidor, se mostrará un mensaje al usuario indicando la disponibilidad de ese nombre de usuario.

Actividad 1

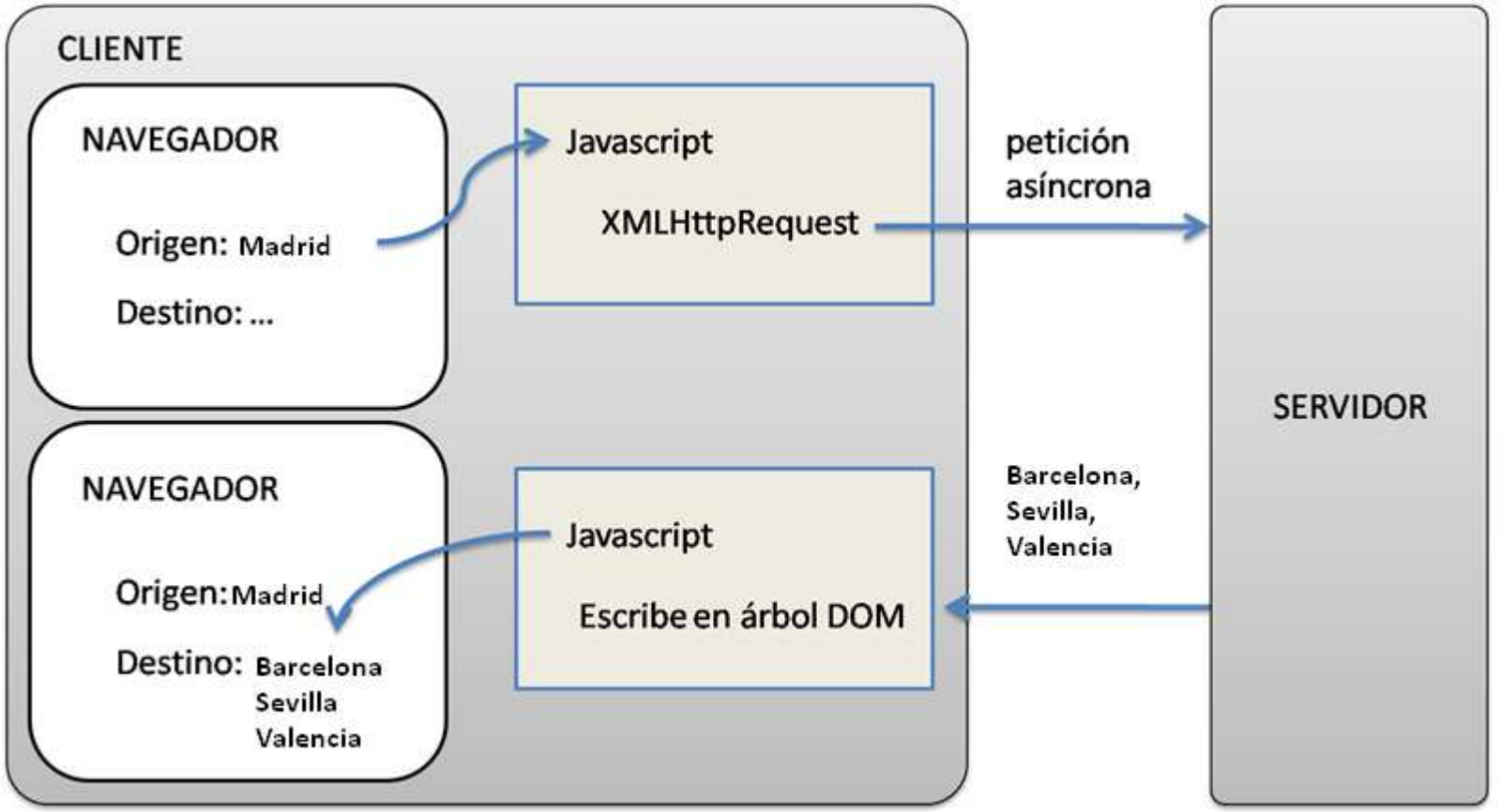
Comprobar disponibilidad del login

Nombre de usuario: RafaPerez

Comprobar disponibilidad...

El nombre elegido [RafaPerez] está disponible

1.- Perspectiva Global con AJAX



2.- Recepción de información: XML y JSON.

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

```
{
 'nombre':'pepe',
 'edad':34,
 'domiclio':'calle alcalá 1',
 'estudios':['primario','secundario','universitario']
}
```

2.- Recepción de información: XML

- El objeto XMLHttpRequest permite la recepción de respuestas del servidor en formato XML.
- Una vez obtenida la respuesta del servidor mediante la propiedad `peticion_http.responseXML`, es posible procesarla empleando los métodos DOM de manejo de documentos XML/HTML.

2.- Recepción de información: XML

```
function respuesta(){
if(peticion_http.readyState == 4 { //datos recibidos
 if (peticion_http.status == 200) { //respuesta OK
  //almacenamos el fichero XML en la vble datos
  var datosXML = peticion_http.responseXML;
  //recorremos el fichero XML mediante DOM
  var c1 = datosXML.getElementByTagName("ciudadano")[0];
  var nom = c1.getElementByTagName("nombre")[0].firstChild.nodeValue;
  var edad = c1.getElementByTagName("edad")[0].firstChild.nodeValue;
  var es = c1.getElementByTagName("estudios")[0];
  var esU= es.getElementByTagName("estudio")[2].firstChild.nodeValue;

  //mostramos los valores obtenidos
  document.getElementById("respuesta").innerHTML = "Nombre: " + nom +
   "<br/>" + "Edad: " + edad + <br/> + "Estudios Universitarios: " +
   esU;
  }
 }
}
```

Actividad 2

- Crea un script en el que se realice una petición AJAX mediante el método GET para cargar el fichero `catalogo.xml` (o `datosXML.php`)
- Esta aplicación PHP, nos devolverá un fichero XML, con una lista de CD's de música con el título, artista, año, etc.
- Haremos un bucle para recorrer todos los cd's del catálogo e imprimir en una tabla la información del título, artista, año.
- Dentro de cada CD, accederemos al elemento que nos interese e imprimiremos su contenido. Para imprimir el contenido de cada nodo, tendremos que hacerlo con el comando `try { } catch` {}, ya que si intentamos acceder a un nodo que no tenga contenido, nos dará un error de JavaScript, puesto que el elemento hijo no existe, y entonces se detendrá la ejecución de JavaScript y no imprimirá nuestro listado.

2.- Recepción de información: JSON

- Algunas ocasiones es más útil intercambiar información con el servidor en formato JSON, ya que es un formato mucho más compacto y ligero que XML.
- Para obtener la respuesta JSON del servidor utilizamos la propiedad `peticion_http.responseText`
- Como la respuesta se devuelve en forma de cadena de texto, se debe transformar esa cadena de texto en un objeto JSON. La forma más sencilla para realizar esa conversión es mediante la función `eval()`.

2.- Recepción de información: JSON

```
function respuesta(){
if(peticion_http.readyState == 4 { //datos recibidos
 if (peticion_http.status == 200) { //respuesta OK
  //almacenamos el fichero de texto recibido en la vble resJSON
  var resJSON = peticion_http.responseText;
  //convertimos la respuesta a un objeto JSON para acceder a él
  var datosJSON = eval("("+resJSON+"));
  //se debe añadir paréntesis al principio y final para la conversión
  //utilizamos la notación de puntos para acceder a la información
  var nom = datosJSON.nombre;
  var edad = datosJSON.edad;
  var esU = datosJSON.etudios[2];

//mostramos los valores obtenidos
  document.getElementById("respuesta").innerHTML = "Nombre: " + nom +
   "<br/>" + "Edad: " + edad + <br/> + "Estudios Universitarios: " +
   esU;
  }
 }
}
```

Actividad 3

- Crea un script en el que se realice una petición AJAX mediante el método GET para cargar el fichero `datosJSON.php`
- La página PHP, consultará a una tabla sql y nos devolverá los resultados en formato JSON. Será una lista de centros con su información (nombre, localidad, provincia, etc.)
- Deberemos evaluar la expresión recibida con la función `eval()` de JavaScript para convertir la cadena de texto recibida en un objeto JSON.

```
var resultados=eval('('+this.responseText+')');
```

- Después recorreremos mediante un bucle todos los objetos literales recibidos en el array resultados y mostraremos su contenido.