TEMA 8. Ajax

Introducción

Ajax es el acrónimo de *Asynchronous JavaScript and XML* y hace referencia a un conjunto de técnicas que nos permiten:

- enviar información al servidor desde un documento HTML
- y recibir información del servidor sobre el documento HTML

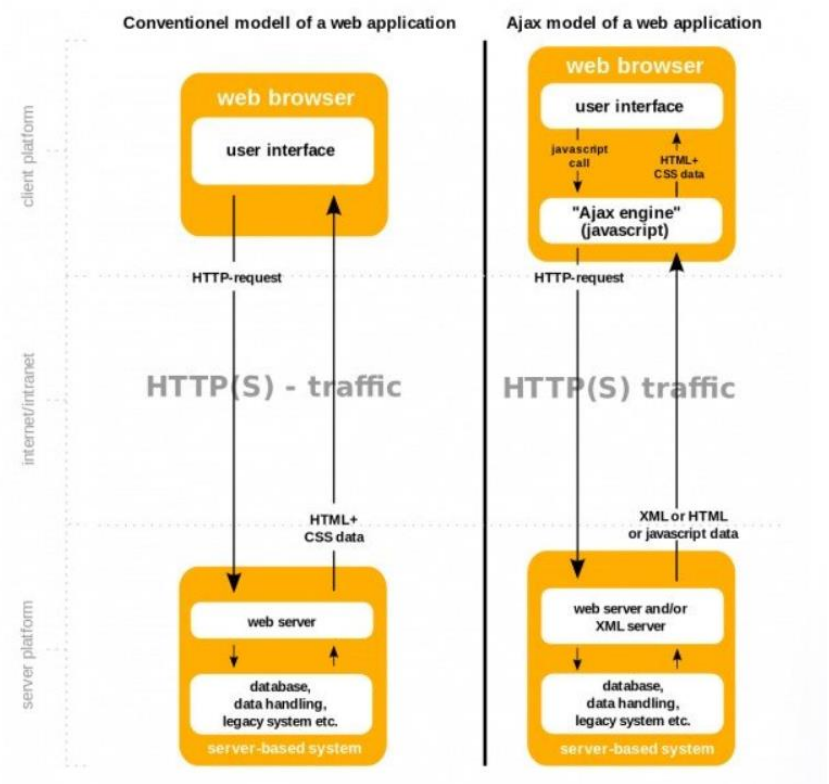
sin tener que volver a cargarlo.

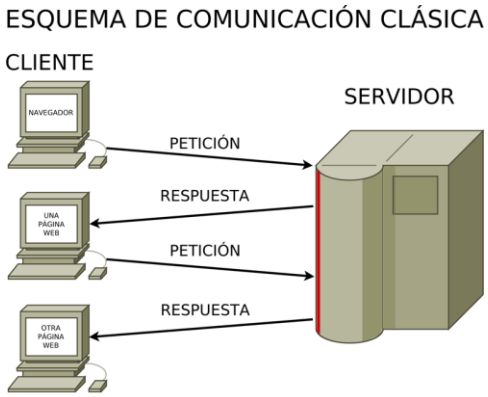
Obviamente esto supone una mejora de la experiencia de usuario sobresaliente, pues **el usuario tendrá la impresión de estar contemplando un documento "vivo" en el que los contenidos se van actualizando en tiempo real**; piense por ejemplo en la portada de un diario de economía en el que las cotizaciones se van actualizando automáticamente sin que el usuario tenga que recargar una y otra vez el documento.

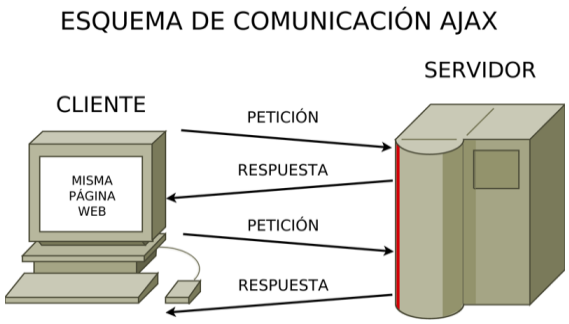
¿Cómo funciona AJAX?

Hay que tener en cuenta que AJAX no es una sola tecnología, ni es un lenguaje de programación. Como se ha introducido antes, AJAX es un conjunto de técnicas de desarrollo web. El sistema generalmente comprende:

1. **HTML/XHTML** para el lenguaje principal y CSS para la presentación.
2. El **Modelo de objetos del documento (DOM)** para datos de visualización dinámicos y su interacción.
3. **XML** para el intercambio de datos y **XSLT** para su manipulación. Muchos desarrolladores lo han reemplazado por **JSON** porque es más similar a JavaScript en su forma.
4. El objeto **XMLHttpRequest** para la ***comunicación asíncrona***. Actualmente API **Fetch**.
5. Finalmente, el lenguaje de programación **JavaScript** para unir todas estas tecnologías.







Comparación de los dos modelos:

Modelo convencional

1. Se envía una solicitud HTTP desde el navegador web al servidor.
2. El servidor recibe y, posteriormente, recupera los datos.
3. El servidor envía los datos solicitados al navegador web.
4. El navegador web recibe los datos y vuelve a cargar la página para que aparezcan los datos.

Durante este proceso, los usuarios no tienen más remedio que esperar hasta que se complete todo el proceso. No solo consume mucho tiempo, sino que también supone una carga innecesaria en el servidor.

Modelo AJAX

1. El navegador crea una llamada de JavaScript que luego activará XMLHttpRequest.
2. En segundo plano, el navegador web crea una solicitud HTTP al servidor.
3. El servidor recibe, recupera y envía los datos al navegador web.
4. El navegador web recibe los datos solicitados que aparecerán directamente en la página. No se necesita recargar.

El término Ajax fue acuñado por Jesse James Garret en 2005, pero en realidad las técnicas a las que se refiere son muy anteriores, casi tan antiguas como la propia Web. Dentro de estas técnicas destaca el objeto **XMLHttpRequest**, que fue introducido por Internet Explorer 5 en 1999. Este **objeto es el núcleo de la mayoría de aplicaciones Ajax** actuales y ha sido estandarizado por el W3C dentro del DOM. A pesar de su nombre, **XMLHttpRequest permite intercambiar con el servidor datos en cualquier formato**, ***no sólo en XML***.

Comunicación asíncrona basada en el objeto XMLHttpRequest

El procedimiento general para comunicarse mediante el objeto XMLHttpRequest consta de las siguientes fases:

1. Crear una instancia del objeto XMLHttpRequest como si de cualquier otro objeto intrínseco se tratara. Por ejemplo: `var xhr = new XMLHttpRequest();`.
2. Si se desea, añadir oyentes de evento al objeto **XMLHttpRequest**. Podemos configurar oyentes para distintos eventos que se producen durante la comunicación; el más importante de estos eventos (por su universal aceptación) es **`readystatechange`**, que se dispara cada vez que cambia el estado de la petición (y que se explica con más detalle en esta misma sección). No obstante, existen otros (que se explican en la sección siguiente), como **`progress`**, que sólo podrán ser debidamente "escuchados" si **asignamos el oyente antes de ejecutar el método `open`** (siguiente paso) del objeto XMLHttpRequest.

3. Llamar al método **`open`** del objeto XMLHttpRequest enviándole como argumentos el tipo de petición que deseamos realizar (generalmente 'GET' o 'POST'), el url al que queremos remitir la petición, y si queremos realizarla asíncronamente (true) o síncronamente (false); por ejemplo, `xhr.open('GET','ajax.php',true);`

    - Si **petición síncrona**: la ejecución de nuestro código JavaScript se detendrá hasta que se reciba la respuesta.
    - Si **petición asíncrona**: el código podrá seguir su flujo de ejecución.

    En cualquier caso (síncrono o asíncrono), al recibir la respuesta, se disparará un evento **`readystatechange`** sobre el objeto **XMLHttpRequest**.

Nota: **¿GET o POST?**

las peticiones GET:

- se utilizan cuando queremos solicitar información al servidor enviándole unos datos como criterio de selección de esa información; por el contrario,
- tienen restringido su tamaño.
- las GET NO pueden enviar archivos.

las peticiones POST:

- se utilizan habitualmente para enviar información al servidor que queremos que almacene (por ejemplo, en una base de datos, o en un archivo),
- las POST NO tienen restringido su tamaño.
- Las peticiones POST pueden enviar archivos.

❖ Las peticiones GET son más rápidas que las POST.

Existen otros métodos como PUT o DELETE, pero su uso aún no está ampliamente aceptado (incluso algunos firewalls bloquean este tipo de peticiones).

4. Definir los encabezados de la petición a través del método `setRequestHeader('nombreEncabezado','valorEncabezado')` del objeto XMLHttpRequest. **Una petición HTTP** está compuesta por el contenido en sí y los encabezados. **En los encabezados** el navegador envía información adicional al servidor, como cookies, formato del contenido de la petición (XML, JSON, …), …

5. Enviar la petición al servidor a través del argumento del método **`send`** del objeto XMLHttpRequest; por compatibilidad con versiones antiguas de navegadores, si no queremos enviar ningún dato en la petición conviene enviar a este método el argumento `null`.

    - Si la **petición** era **síncrona**: se **detendrá la ejecución del código JavaScript hasta que se reciba la respuesta del servidor** y **después** podremos saber cuál ha sido el tipo de respuesta a través de la propiedad **`status`** del objeto XMLHttpRequest (esta **propiedad está disponible tanto en las peticiones síncronas como en las asíncronas**, y **contiene el código de estado devuelto por el protocolo HTTP)**;

      por ejemplo:

        - valor 200 indica que la transacción se ha tramitado correctamente.
        - valor 304 indica que el recurso solicitado no ha sido modificado desde la última vez que fue solicitado, por lo que puede obtenerse de la caché del navegador.
        - valor 403 indica que no estamos autorizados a acceder al recurso solicitado.
        - valor 404 indica que el recurso solicitado no existe.
        - valor 500 indica que se ha producido un error interno en el servidor al procesar el recurso solicitado.

(puede consultar el listado completo de códigos de estado HTTP en http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html).

¿Y si el servidor no contesta? El navegador se quedará eternamente bloqueado. Obviamente esto resultaría nefasto, y ése es el principal motivo por el que **las peticiones Ajax síncronas (conocidas como Sjax) apenas se utilizan**.

- Si **peticiones** son **asíncronas**: seguirá el flujo de ejecución normal y, si lo deseamos, podremos incluso controlar el progreso de la transacción a través de distintos eventos; el evento más importante es `readystatuschange`, que se dispara cada vez que se actualiza el valor de la propiedad **`readyState`** del **objeto XMLHttpRequest** (no confundir con la propiedad readyState del objeto document).

  En la siguiente tabla se describen los valores que puede adoptar esta propiedad de sólo lectura.

Valor	Descripción
0	Sin inicializar. Aún no se ha llamado al método `open` del objeto XMLHttpRequest.
1	Abierto. Se ha llamado al método `open` del objeto XMLHttpRequest, pero no al método send.
2	Enviado. Se ha llamado al método `send` del objeto XMLHttpRequest, pero aún no se ha recibido la respuesta del servidor.
3	Recibiendo. Se ha empezado a recibir la respuesta del servidor.
4	Completado. Se ha recibido la respuesta completa del servidor. Tenga en cuenta que esto no quiere decir que la petición se haya tramitado correctamente; por ejemplo, puede ser que el servidor haya contestado con un código 404. **Para tener la certeza de que una petición se ha tramitado correctamente** tendremos que tener el valor 4 en `readyState` y el valor 200 en `status`.

Nota: Además de la propiedad `status`, el objeto XMLHttpRequest también posee la propiedad `statusText`, que en lugar del código de estado contiene el mensaje completo de estado; por ejemplo: '200 OK'.

6. Gestionar la respuesta del servidor. El contenido de la respuesta estará disponible en:
   - la propiedad `responseText` del objeto XMLHttpRequest, y
   - en los encabezados de la respuesta, pudiéndolos conseguir a través de los métodos `getResponseHeader(`*`nombreEncabezado`*`)` y `getAllResponseHeaders()`.

Por ejemplo, copie los siguientes dos archivos en la carpeta de proyectos de su servidor web (si está usando WAMP Server, cópielos en la carpeta wamp/www/ajax), acceda al primero con su navegador web y pulse el botón **ENVIAR PETICIÓN** para comprobar cómo se reciben datos nuevos del servidor sin volver a recargar la página. El archivo ajax.php está escrito en PHP y simplemente escribe en la respuesta el mensaje 'La fecha del servidor es:', deja pasar 10 segundos, y escribe la fecha/hora actual del sistema, que es lo que recibirá nuestro objeto xhr en su propiedad `responseText`.

Nota: En los ejemplos de este tema vamos a necesitar una aplicación del lado del servidor a la que dirigir nuestras peticiones Ajax. Se ha elegido como lenguaje para estas aplicaciones PHP por ser uno de los más populares, pero existen otras alternativas entre las que cabe destacar el **Node.js**, que utiliza exactamente la misma sintaxis que JavaScript, por lo que resulta muy cómodo.

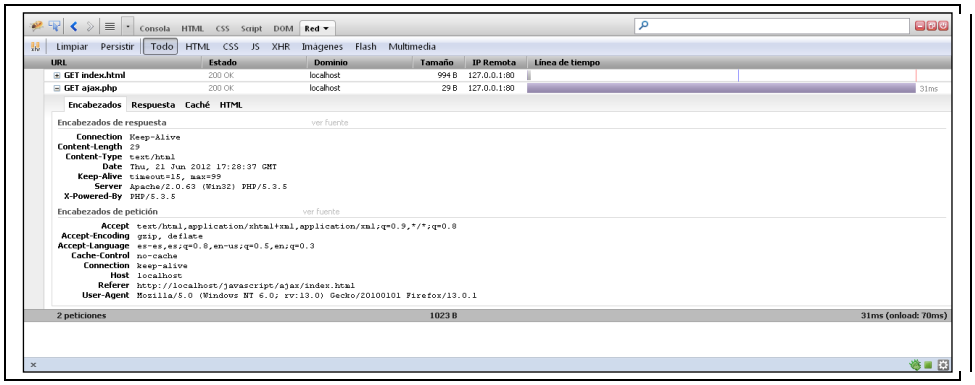
index.html

```
001  <!DOCTYPE html>
002  <html>
003          <head>
004                  <title>AJAX</title>
005                  <style>
006                  </style>
007                  <script>
008                          var xhr;
009                          function enviarPeticionAJAX(evento){
010                                  evento.target.disabled=true;
011                                  xhr = new XMLHttpRequest();
012                                  xhr.addEventListener('readystatechange',gestionarRespuesta);
013                                  xhr.open('GET','ajax.php',true);
014                                  xhr.send(null);
015                          }
016                          function gestionarRespuesta(evento){
017                          if(evento.target.readyState == 4 && evento.target.status == 200){
018                          document.getElementById('encabezados').innerHTML =
     evento.target.getAllResponseHeaders();
019          document.getElementById('contenido').innerHTML = evento.target.responseText;
020          document.getElementById('boton').disabled = false;
021                                  }
022                          }
023                  </script>
024          </head>
025          <body>
026                  <button id='boton' type='button'
     onclick='enviarPeticionAJAX(event);'>ENVIAR PETICI&Oacute;N</button>
027                  <h1>Encabezados respuesta</h1>
028                  <div id='encabezados'></div>
029                  <h1>Contenido respuesta</h1>
030                  <div id='contenido'></div>
031          </body>
032  </html>
```

ajax.php

```
001  <?php
002          echo 'La fecha del servidor es:
     <br />';
003          sleep(10);
004          echo date(DATE_RFC822);
005  ?>
```

Nota: Para depurar las peticiones y respuestas que se producen en AJAX es muy útil recurrir al panel **Red** de Firebug. Por ejemplo, en la siguiente figura se muestra la información que nos aporta este panel al ejecutar el ejemplo anterior.



EJERCICIO

Cambie en la línea 13 el url, ajax.php por otro que no exista, y modifique el código del oyente gestionarRespuesta para que indique que se ha producido un error en caso de que el estado (`status`) de la respuesta no sea 200 ó 304 (recuerde que el código 304 indica que el recurso no ha cambiado y se está sirviendo desde la caché del navegador en lugar de solicitarlo completamente al servidor).

A continuación, se muestra una posible solución.

```
001  function gestionarRespuesta(evento){
002          if(evento.target.readyState == 4){
003                  if (evento.target.status >= 200 && evento.target.status <=304){
004                          document.getElementById('encabezados').innerHTML =
     evento.target.getAllResponseHeaders();
005                          document.getElementById('contenido').innerHTML =
     evento.target.responseText;
006                          document.getElementById('boton').disabled = false;
007                  }else{
008                          alert('Error de comunicaci\u00f3n\n' +
     evento.target.statusText);
009                          document.getElementById('boton').disabled = false;
010                  }
011          }
012  }
```

Controlar el progreso de las peticiones Ajax

En las transacciones Ajax asíncronas podemos controlar su progreso a través de los **eventos asociados al objeto** XMLHttpRequest. El más importante de estos eventos es `readystatechange`, que ya explicamos en la sección anterior.
Además, disponemos de los siguientes eventos asociados a XMLHttpRequest en procesos de transacciones. (aunque su implementación no está completamente homogeneizada en los navegadores):

- `abort`. Se produce al cancelar la transacción con el método `abort()` del objeto XMLHttpRequest. Es excluyente de los eventos `error` y `load`.
- `error`. Se produce cuando se detecta un error en la transacción. Es excluyente de los eventos `abort` y `load`.
- `load`. Se produce cuando la transacción se ha realizado correctamente. **No debemos confundir que la transacción sea correcta con que la respuesta sea correcta**. Por ejemplo, podemos tener una transacción correcta con una respuesta de código 404 (documento no encontrado). Por este motivo siempre hay que consultar el valor de la propiedad `status` del objeto XMLHttpRequest. Este evento es excluyente de los eventos `abort` y `error`.
- `loadstart`. Se produce al recibir el primer byte de la respuesta.
- `progress`. Es, junto a readystatechange, el **único evento que puede producirse varias veces** durante la transacción. Indica que se ha producido algún progreso en la recepción de la respuesta.
- `timeout`. Se produce si la **transacción ha superado su plazo de ejecución** y va a cancelarse. El plazo de ejecución de un objeto XMLHttpRequest se establece a través de su **propiedad** `timeout`, y su valor se expresa en milisegundos.

Todos estos eventos, excepto `readystatechange`, poseen las siguientes propiedades:

- `lengthComputable`. Es un **valor booleano** que nos indica si se conoce el tamaño en bytes total de la respuesta.
- `loaded`. Indica cuántos bytes de la respuesta se han cargado ya.
- `total`. Si `lengthComputable` es true, indica el tamaño total en bytes de la respuesta.

Por ejemplo, vamos a completar el código anterior añadiendo algunos oyentes (el código adicional se ha resaltado en negrita). Pruebe este código varias veces y en distintos navegadores (Firefox, Opera y Chrome, por ejemplo) alternando el valor de la propiedad `timeout` de 5000 a 15000, para comprobar que la implementación de los eventos distintos a `readystatechange` es ligeramente desigual.

```
001    <!DOCTYPE html>
002    <html>
003            <head>
004                    <title>AJAX</title>
005                    <style>
006                    </style>
007                    <script>
008                            var xhr;
009                            function enviarPeticionAJAX(evento){
010                                    evento.target.disabled=true;
011                                    xhr = new XMLHttpRequest();
012
               xhr.addEventListener('readystatechange',gestionarRespuesta,false);
013
               xhr.addEventListener('abort',function(){console.log('abort');evento.target.disable
       d = false;});
014
               xhr.addEventListener('load',function(){console.log('load');evento.target.disabled
       = false;});
015
               xhr.addEventListener('error',function(){console.log('error');evento.target.disable
       d = false;});
016                                    xhr.timeout = '15000';
017
               xhr.addEventListener('timeout',function(){console.log('timeout');evento.target.dis
       abled = false;});
018                                    xhr.addEventListener('progress',gestionarProgreso,false);
019                                    xhr.open('GET','ajax.php',true);
020                                    xhr.send(null);
021                            }
022                            function gestionarRespuesta(evento){
023                                    console.log('readystatechange: ' +
       evento.target.readyState);
024                                    if(evento.target.readyState == 4 && evento.target.status ==
       200){
025                                            document.getElementById('encabezados').innerHTML =
       evento.target.getAllResponseHeaders();
026                                            document.getElementById('contenido').innerHTML =
       evento.target.responseText;
027                                            document.getElementById('boton').disabled = false;
028                                    }
029                            }
030                            function gestionarProgreso(evento){
031                                    console.log('progress: ' + (evento.lengthComputable ?
       evento.loaded/evento.total * 100 + '%':'desconocido'));
032                            }
033                    </script>
034            </head>
035            <body>
036                    <button id='boton' type='button'
       onclick='enviarPeticionAJAX(event);'>ENVIAR PETICI&Oacute;N</button>
037                    <h1>Encabezados respuesta</h1>
038                    <div id='encabezados'></div>
039                    <h1>Contenido respuesta</h1>
040                    <div id='contenido'></div>
041            </body>
042    </html>
```

Envío de datos al servidor

Existen dos técnicas para configurar los datos que queremos enviar al servidor en las peticiones GET y POST de Ajax:

- El **método clásico**: que consiste en adjuntar los datos como cadena de búsqueda, bien
  - en el propio **url** de la petición en las peticiones **GET**.
  - como argumento del método **`send`** del objeto **XMLHttpRequest** en las peticiones **POST**. En este caso, *si queremos que **el servidor reciba los datos** como si procediesen de un formulario* (el método POST se usa generalmente para enviar datos de formulario) tendremos que **configurar en encabezado** **`Content-Type`** con el valor `application/x-www-form-urlencoded`.

  El mayor *inconveniente* del ***método clásico*** es que ***nos obliga a recurrir al DOM para ir recopilando todos los datos de los controles que queremos enviar***, y esto puede resultar bastante tedioso en algunos casos; por el contrario, su **mayor ventaja** es que es compatible con todos los navegadores.

  Por ejemplo, en este primer código se realiza una petición GET, y en el siguiente se resaltan en negrita las modificaciones que habría que introducir para realizar la misma petición con el método POST; tras ellos, el tercer código corresponde a la aplicación del lado del servidor, que simplemente responde con la calificación del alumno y materia elegidos (es válida para los métodos GET y POST porque utiliza la superglobal $_REQUEST).

index.html para petición GET

```
001  <!DOCTYPE html>
002  <html>
003          <head>
004                  <title>AJAX</title>
005                  <style>
006                  </style>
007                  <script>
008                          var xhr;
009                          var alumno;
010                          var materia;
011                          var calificacion;
012                          function enviarPeticionAJAX(evento) {
013                                  if (alumno.value !='' && materia.value != ''){
014                                          alumno.disabled = true;
015                                          materia.disabled = true;
016                                          xhr = new XMLHttpRequest();
017                                          xhr.addEventListener('readystatechange',
     gestionarRespuesta, false);
018                                          xhr.open('GET', 'ajax.php?alumno=' + alumno.value +
     '&materia=' + materia.value, true);
019                                          xhr.send(null);
020                                  }
021                          }

022                          function gestionarRespuesta(evento) {
023                                  if (evento.target.readyState == 4 && evento.target.status ==
     200) {
024                                          alumno.disabled = false;
025                                          materia.disabled = false;
026                                          calificacion.value = evento.target.responseText;
027                                  }
028                          }
029                          document.addEventListener('readystatechange', inicializar, false);
```

```
030                         function inicializar() {
031                                 if (document.readyState == 'complete') {
032                                         alumno = document.getElementById('alumno');
033                                         materia = document.getElementById('materia');
034                                         calificacion =
    document.getElementById('calificacion');
035                                         alumno.addEventListener('change', enviarPeticionAJAX,
    false);
036                                         materia.addEventListener('change',
    enviarPeticionAJAX, false);
037                                 }
038                         }
039                 </script>
040         </head>
041         <body>
042                 <label for='alumno'>Alumno: </label>
043                 <select id='alumno'>
044                         <option value='' selected='selected'>--Elija un alumno--</option>
045                         <option>Juan F&eacute;lix Mateos</option>
046                         <option>Ana Irene Palma</option>
047                 </select>
048                 <label for='materia'>Materia: </label>
049                 <select id='materia'>
050                         <option value='' selected='selected'>--Elija una materia--</option>
051                         <option>Lenguaje</option>
052                         <option>Matem&aacute;ticas</option>
053                 </select>
054                 <label for='calificacion'>Calificaci&oacute;n: </label>
055                 <input type='text' readonly='readonly' id='calificacion'/>
056         </body>
057 </html>
```

index.html para petición POST

```
001 <!DOCTYPE html>
002 <html>
003         <head>
004                 <title>AJAX</title>
005                 <style>
006                 </style>
007                 <script>
008                         var xhr;
009                         var alumno;
010                         var materia;
011                         var calificacion;

012                         function enviarPeticionAJAX(evento) {
013                                 if (alumno.value !='' && materia.value != ''){
014                                         alumno.disabled = true;
015                                         materia.disabled = true;
016                                         xhr = new XMLHttpRequest();
017                                         xhr.addEventListener('readystatechange',
    gestionarRespuesta, false);
018                                         xhr.open('POST', 'ajax.php', true);
019                                         xhr.setRequestHeader('Content-Type',
    'application/x-www-form-urlencoded');
020                                         xhr.send('alumno=' + alumno.value +
    '&materia=' + materia.value);
021                                 }
022                         }

023                         function gestionarRespuesta(evento) {
024                                 if (evento.target.readyState == 4 &&
    evento.target.status == 200) {
025                                         alumno.disabled = false;
026                                         materia.disabled = false;
027                                         calificacion.value =
    evento.target.responseText;
028                                 }
029                         }
030                         document.addEventListener('readystatechange',
    inicializar, false);
```

```
031                         function inicializar() {
032                                 if (document.readyState == 'complete') {
033                                         alumno =
    document.getElementById('alumno');
034                                         materia =
    document.getElementById('materia');
035                                         calificacion =
    document.getElementById('calificacion');
036                                         alumno.addEventListener('change',
    enviarPeticionAJAX, false);
037                                         materia.addEventListener('change',
    enviarPeticionAJAX, false);
038                                 }
039                         }
040                 </script>
041         </head>
042         <body>
043                 <label for='alumno'>Alumno: </label>
044                 <select id='alumno'>
045                         <option value='' selected='selected'>--Elija un alumno--
    </option>
046                         <option>Juan F&eacute;lix Mateos</option>
047                         <option>Ana Irene Palma</option>
048                 </select>
049                 <label for='materia'>Materia: </label>
050                 <select id='materia'>
051                         <option value='' selected='selected'>--Elija una materia-
    -</option>
052                         <option>Lenguaje</option>
053                         <option>Matem&aacute;ticas</option>
054                 </select>
055                 <label for='calificacion'>Calificaci&oacute;n: </label>
056                 <input type='text' readonly='readonly' id='calificacion'/>
057         </body>
058 </html>
```

ajax.php

```
001 <?php
002         $alumno = $_REQUEST['alumno'];
003         $materia = $_REQUEST['materia'];
004         switch ($alumno){
005                 case 'Juan Félix Mateos':
006                         switch ($materia){
007                                 case 'Matemáticas':
008                                         echo '7.5';
009                                         break;
010                                 case 'Lenguaje':
011                                         echo '9.5';
012                                         break;
013                         }
014                         break;
015                 case 'Ana Irene Palma':
016                         switch ($materia){
017                                 case 'Matemáticas':
018                                         echo '8.5';
019                                         break;
020                                 case 'Lenguaje':
021                                         echo '7.5';
022                                         break;
023                         }
024                         break;
025         }
026 ?>
```

- **Utilizar** `FormData` **(objeto intrínseco)**.

  Este objeto es capaz de generar automáticamente la cadena de búsqueda de un formulario, sin que tengamos que ir recabando el valor de cada control recurriendo al DOM, y además ***se encarga de configurar implícitamente los encabezados para hacer creer al servidor que esta cadena de búsqueda procede de un formulario*** que se ha enviado utilizando la ***codificación*** **`multipart/form-data`** (esto puede tener ciertas implicaciones que se resaltarán en la sección posterior "Cross-Origin Resource Sharing").

  Además, el objeto FormData nos ofrece el ***método*** ***`append`***`(nombre,valor)` por si queremos añadir un par que no procede del formulario a la petición.

  Por ejemplo, en el siguiente código se han englobado los controles de lista desplegables en un formulario para poder serializar directamente sus datos con el objeto FormData y también se utiliza el método **`append`** para ilustrar cómo se añadiría un par nombre/dato adicional a una petición (el archivo ajax.php es el mismo del ejemplo anterior, y se han resaltado en negrita las principales diferencias):

```
001  <!DOCTYPE html>
002  <html>
003          <head>
004                  <title>AJAX</title>
005                  <style>
006                  </style>
007                  <script>
008                          var xhr;
009                          var alumno;
010                          var materia;
011                          var calificacion;
012                          var datos;
013                          function enviarPeticionAJAX(evento) {
014                                  if (alumno.value !='' && materia.value != ''){

015                                          datos = new FormData(document.forms[0]);
016                                          alumno.disabled = true;
017                                          materia.disabled = true;

018                                          datos.append('curso','11/12');
019                                          xhr = new XMLHttpRequest();
020                                          xhr.addEventListener('readystatechange',
     gestionarRespuesta, false);
021                                          xhr.open('POST', 'ajax.php', true);
022                                          xhr.send(datos);
023                                  }
024                          }
025                          function gestionarRespuesta(evento) {
026                                  if (evento.target.readyState == 4 && evento.target.status ==
     200) {
027                                          alumno.disabled = false;
028                                          materia.disabled = false;
029                                          calificacion.value = evento.target.responseText;
030                                  }
031                          }
032                          document.addEventListener('readystatechange', inicializar, false);
033                          function inicializar() {
034                                  if (document.readyState == 'complete') {
035                                          alumno = document.getElementById('alumno');
036                                          materia = document.getElementById('materia');
037                                          calificacion =
     document.getElementById('calificacion');
038                                          alumno.addEventListener('change', enviarPeticionAJAX,
     false);
039                                          materia.addEventListener('change',
     enviarPeticionAJAX, false);
040                                  }
041                          }
042                  </script>
043          </head>
```

```
044         <body>
045                 <form id='formulario'>
046                         <label for='alumno'>Alumno: </label>
047                         <select id='alumno' name='alumno'>
048                                 <option value='' selected='selected'>--Elija un alumno--
    </option>
049                                 <option>Juan F&eacute;lix Mateos</option>
050                                 <option>Ana Irene Palma</option>
051                         </select>
052                         <label for='materia'>Materia: </label>
053                         <select id='materia' name='materia'>
054                                 <option value='' selected='selected'>--Elija una materia--
    </option>
055                                 <option>Lenguaje</option>
056                                 <option>Matem&aacute;ticas</option>
057                         </select>
058                         <label for='calificacion'>Calificaci&oacute;n: </label>
059                         <input type='text' readonly='readonly' id='calificacion'/>
060                 </form>
061         </body>
062 </html>
```

Nota: Tenga en cuenta que los controles de los formularios que están deshabilitados (atributo disabled) no se envían al servidor. ¿Qué hubiera ocurrido en el código anterior si las líneas 16 y 17 se hubieran colocado delante de la 15? Que como los campos estarían deshabilitados al crear la instancia de FormData no se incluirían en él.