Programación con AJAX

José Antonio Muñoz Jiménez

jamj2000 at gmail dot com

Marzo 2010

Introducción.

- ¿Qué es **AJAX?**
- ¿Dónde se utiliza?
- Ventajas y desventajas

El formato **XML**

Conceptos de **Javascript**

- Variables
  - Arrays
  - Diccionarios
- Funciones y Eventos

El objeto **XMLHttpRequest**

- Métodos del objeto XMLHttpRequest para obtener información
  - open
  - setRequestHeader
  - send
- Propiedades de método XMLHttpRequest
  - onreadystatechange
  - readyState
  - responseText
  - responseXML

El objeto **document**

- Métodos para gestionar la información
  - getElementsByTagName
  - getElementById
  - getAttribute
- La propiedad innerHTML

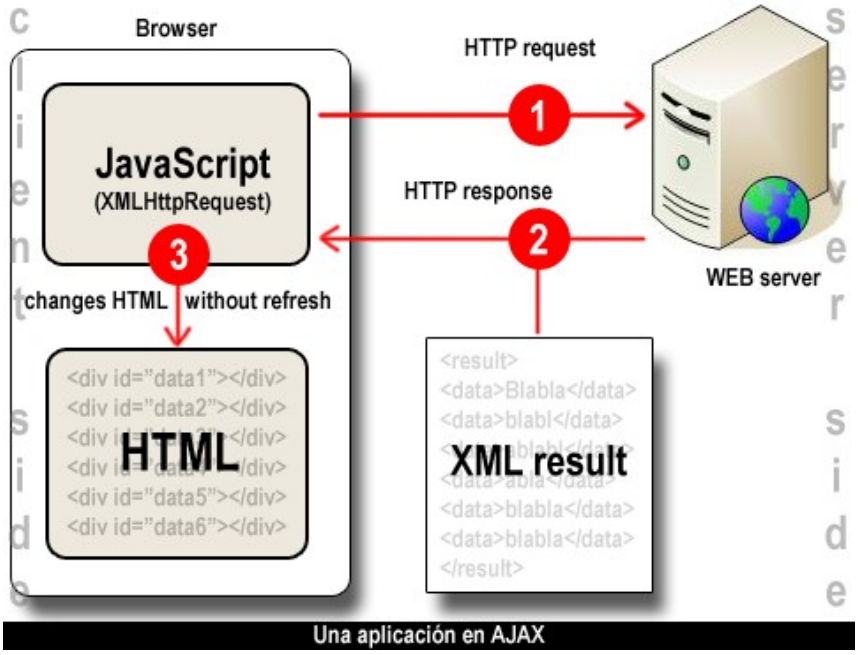
Compatibilidad con distintos navegadores

Introducción

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas que se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación **asíncrona** con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. Dicha técnica es ampliamente utilizada en la Web 2.0.

AJAX significa

- Asynchronous
- Javascript
- And
- XML



Una aplicación en AJAX

Pueden verse ejemplos de aplicaciones web que hacen uso de AJAX en:

- Google (Gmail, Google maps, Google docs, Google suggest, ...)
- Flickr
- Twitter
- Digg y Reddit
- Meebo

En este mini-tutorial encontraras diversos ejemplos que tratan distintos aspectos de AJAX.

Actividades propuestas

- Hacer un listado de al menos 10 sitios de internet que hacen uso de AJAX.
- Hacer un resumen de la historia de la tecnología AJAX.

El formato XML

La mayoría de la comunicación de datos en AJAX entre un servidor y un cliente web suele realizarse en formato XML. Por ello es aconsejable adquirir un conocimiento básico de este formato. Este conocimiento podría resumirse de forma muy escueta en que todo documento XML:

- tiene uno y sólo un nodo raíz.
- todas las etiquetas deben de tener su etiqueta de cierre correspondiente.

Por ejemplo, teniendo en cuenta esto, podemos crear un archivo **datos.xml**:

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/css" href="estilo.css" ?>
<alumnos>
   <alumno>José Antonio</alumno>
   <alumno>Susana</alumno>
   <alumno>Ana</alumno>
</alumnos>
```

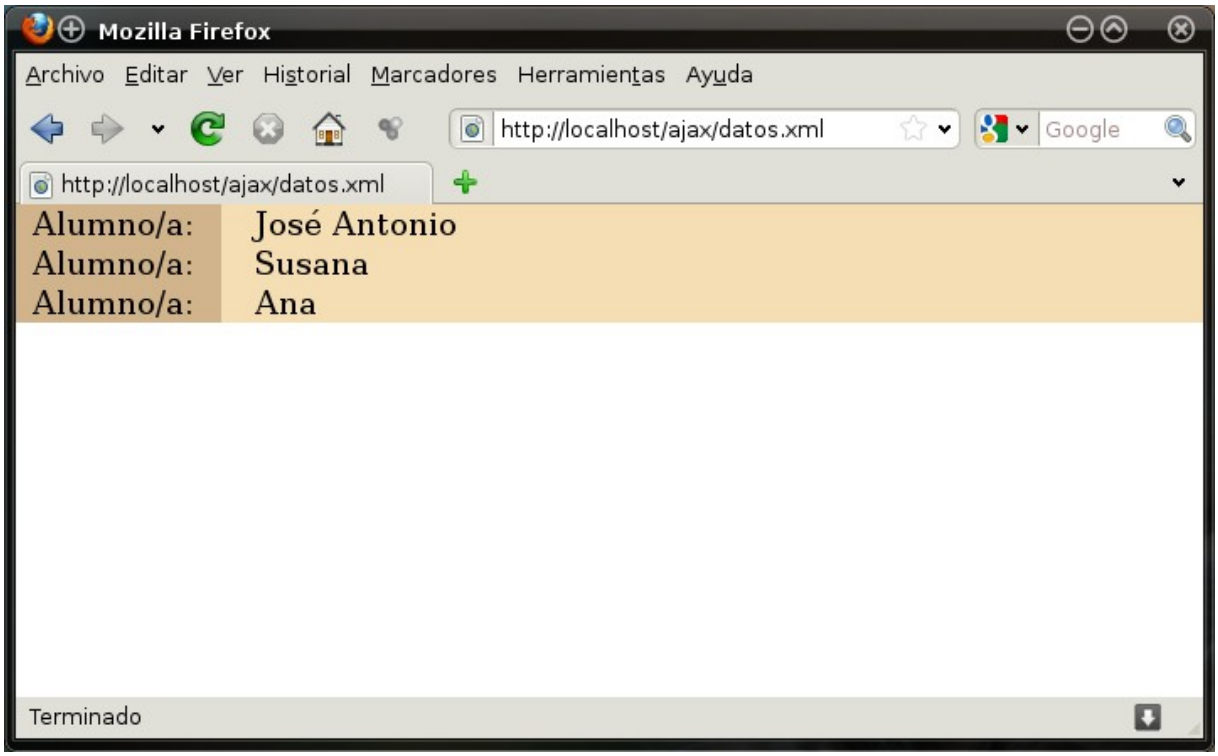
A un archivo XML puede asociarsele una hoja de estilo CSS, si lo indicamos en su cabecera mediante

```
<?xml-stylesheet type="text/css" href="estilo.css" ?>
```

Por ejemplo, la hoja **estilo.css:**

```
// Etiquetas XML
* { margin: 0px; padding: 0px; }

alumnos { font-size: 14pt; }
alumno  { display: block; background-color: Wheat; }

alumno:before {
        content: "Alumno/a: ";
        padding-left: 10px;
        padding-right: 10px;
        margin-right: 20px;
        background-color: Tan;
}



// Etiquetas HTML
* { margin: 0px; padding: 0px; }

#resultado {
        overflow: auto;
        width: 500px; height: 200px;
        background-color: LightGrey;
        border: 1px solid Black;
        padding: 10px;
}
```

Con lo cual se vería de la siguiente forma en el navegador:



Actividades propuestas

- Modificar el archivo datos.xml para que aparezcan al menos 10 alumnos.
- Añadir al archivo datos.xml al menos 3 profesores.
- Modificar estilo.css para que se muestren adecuadamente los profesores.
- Elaborar un nuevo archivo XML que muestre otro tipo de información, con su estilo CSS asociado.

Proporcionando datos desde el servidor

A menudo no se accede a los datos XML de forma directa sino que estos son generados de forma dinámica en el servidor, normalmente mediante un script PHP. Por ejemplo esa información podría obtenerse de una base datos cuya información cambia con el tiempo.

A modo de simplificación mostramos un sencillo script **datos.php**:

```
<?php
sleep (2);       // Simulamos un retardo de 2 seg.
header('Content-type: text/xml');
echo '<?xml version="1.0" encoding="UTF-8" ?>
<alumnos>
   <alumno>José Antonio</alumno>
   <alumno>Susana</alumno>
   <alumno>Ana</alumno>
</alumnos>
';
?>
```

Actividades propuestas

- Modificar el script datos.php para que aparezcan al menos 10 alumnos.
- Realizar una llamada a datos.php desde el navegador.

Obteniendo datos desde el cliente

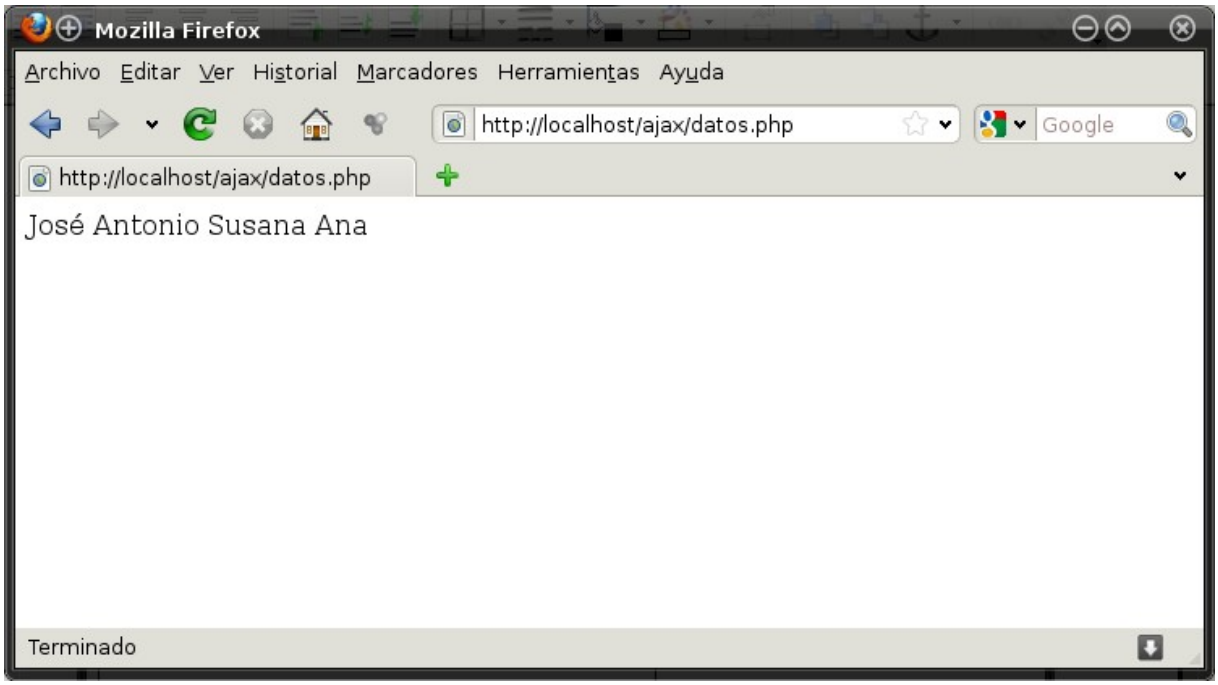
En el lado del cliente estos datos pueden obtenerse de dos formas:

- La forma tradicional
- Mediante AJAX

La forma tradicional

Creamos un archivo **ver.html** y hacemos la petición mediante un submit.

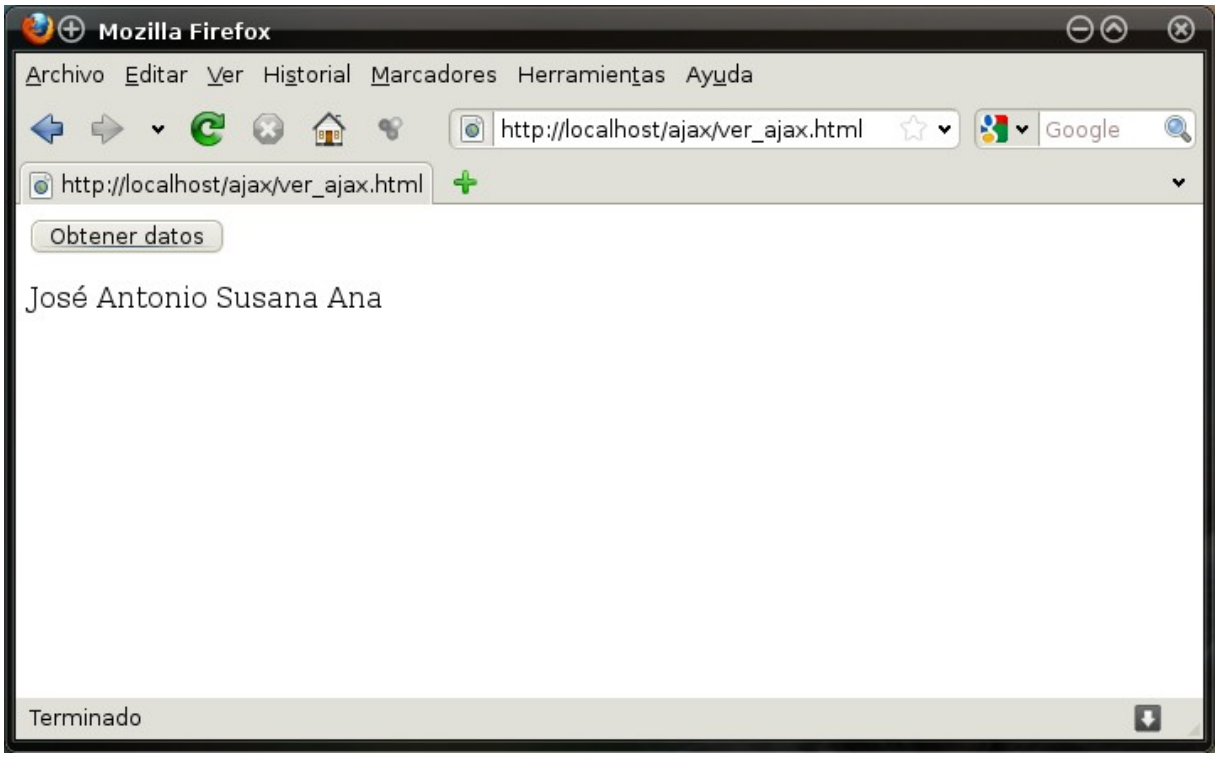
```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
</head>
<body>
  <form action="datos.php" method="get" enctype="text/plain">
      <input type="submit" value="Obtener datos">
  </form>
</body>
</html>
```



Mediante AJAX

Creamos un archivo **ver_ajax.html** y hacemos la petición en javascript.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<script type="text/javascript">

 var datos;

 function pedir_datos (url) {
   datos = new XMLHttpRequest();
   datos.onreadystatechange = mostrar_datos;
   datos.open("GET", url, true);
   datos.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
   datos.setRequestHeader("Connection", "close");
   datos.send(null);
 }


 function mostrar_datos () {
   if (datos.readyState == 4)
     document.getElementById('resultado').innerHTML = datos.responseText;
 }

</script>
</head>

<body>
  <form action="javascript:pedir_datos('datos.php');">
      <input type="submit" value="Obtener datos">
  </form>
  <div id='resultado'> </div>

</body>
</html>
```



Observaciones

Aunque en principio pueda parecer que hay poca diferencia entre ambas formas de obtener y mostrar la información y que incluso el uso de AJAX es algo más complicado, existen diferencias importantes.

Estas son:

- En la forma tradicional existe un cambio de página (observa que en URL aparece http://localhost/ajax/datos.php). Esto elimina de la vista del usuario toda la información que tuviéramos previamente.
- Mediante AJAX, la información se muestra en la misma página original (observa que en URL sigue apareciendo http://localhost/ajax/ver_ajax.html). Esto nos permite trabajar en la misma página como si nada hubiera sucedido actualizándose ésta con nueva información cuando sea necesario. Normalmente el “cuando sea necesario” significa cuando ocurra algún evento que deseemos tener en cuenta.

Otras cuestiones a tener en cuenta:

- Cuando usamos AJAX debemos tener una división o capa (DIV) en la cual mostraremos los datos. Por ejemplo:

```
<div id='resultado'> </div>
```

- Podemos acceder a dicha capa y mostrar el resultado mediante:

```
document.getElementById('resultado').innerHTML = "<b>Aquí va el resultado</b>";
```

Resultado

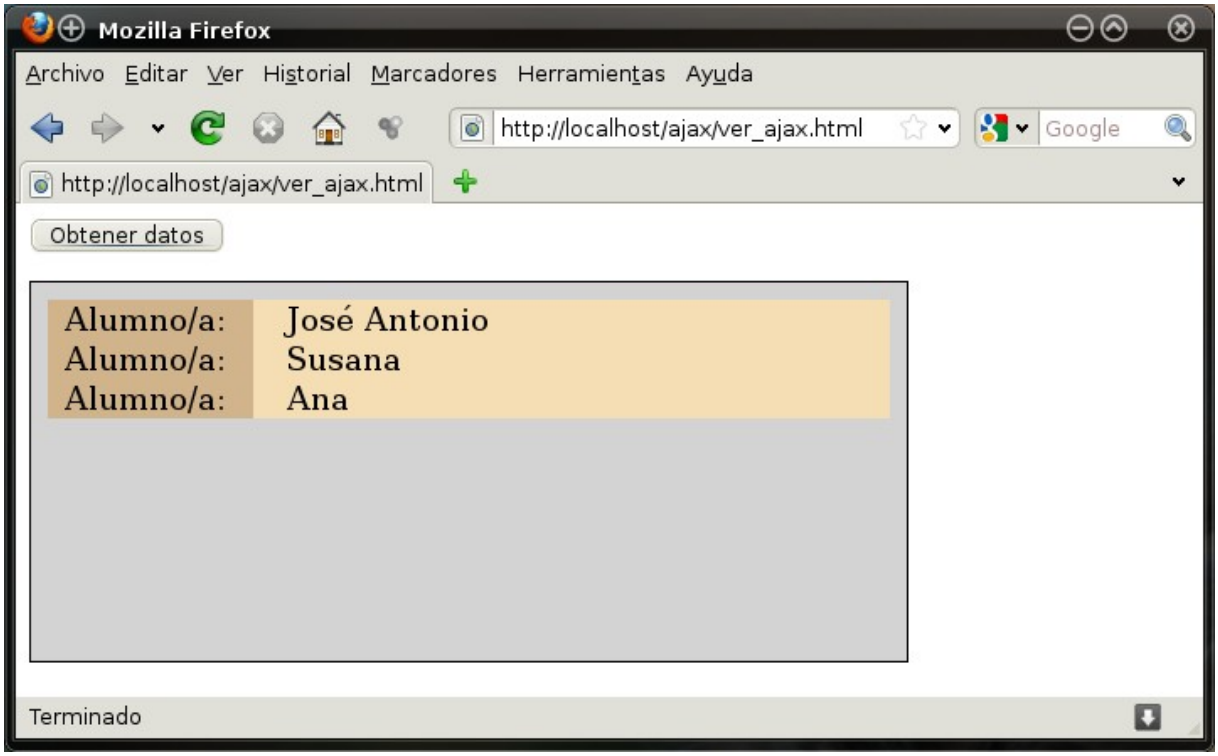
El uso de AJAX nos permitirá que el usuario tenga una experiencia más fluida en la interacción con nuestro servidor web y dará a nuestras páginas un diseño más profesional.

Ahora sólo nos falta añadir una hoja de estilo a la página ver_ajax.html y tendremos nuestra primera aplicación AJAX medianamente decente.

```
<link rel="stylesheet" type="text/css" href="estilo.css" />
```

El resultado puede verse a continuación.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<link rel="stylesheet" type="text/css" href="estilo.css" />
<script type="text/javascript">

 var datos;

 function pedir_datos (url) {
   datos = new XMLHttpRequest();
   datos.onreadystatechange = mostrar_datos;
   datos.open("GET", url, true);
   datos.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
   datos.setRequestHeader("Connection", "close");
   datos.send(null);
 }


 function mostrar_datos () {
   if (datos.readyState == 4)
     document.getElementById('resultado').innerHTML = datos.responseText;
 }

</script>
</head>

<body>
  <form action="javascript:pedir_datos('datos.php');">
      <input type="submit" value="Obtener datos">
  </form>
  <div id='resultado'> </div>
</body>
</html>
```



Actividades propuestas

- Modificar ver_ajax.html para que los datos se muestren en 2 capas distintas a la vez.
- Modificar ver_ajax.html para que los datos se muestren en 4 capas distintas a la vez.

Conceptos de Javascript

Variables

En Javascript las variables no tienen tipo.

```
var dato;

dato = 123;             // ahora dato es un entero
dato = "Hola que tal!"; // ahora dato es una cadena de texto
dato = new Array ();    // ahora dato es un array
dato = {};              // ahora dato es un diccionario
```

Javascript realiza las conversiones implícitas necesarias.

Para dar valores a un **array:**

```
var direccion = new Array ();

direccion[0] = "Avda. ";
direccion[1] = "Estudiantes ";
direccion[2] = 99;

alert (direccion[0] + direccion[1] + direccion[2]);
```

Para dar valores a un **diccionario:**

```
var nota = {};

nota["Rocio"] = "Sobresaliente";
nota["Eva"]   = "Notable";
nota["Jairo"] = "Suficiente";

alert (nota["Rocio"] + "\n" + nota["Eva"] + "\n" + nota["Jairo"]);
```

Actividades propuestas

- Si tenemos la siguiente función mostrar_datos() y el siguiente body, cual será el resultado. ¿Qué tipo de dato es la variable d?

```
function mostrar_datos () {
    if (datos.readyState == 4) {
              var d = document.getElementsByTagName('div');

              d[0].innerHTML = datos.responseText;
              d[1].innerHTML = datos.responseText;
              d[2].innerHTML = datos.responseText;
              d[3].innerHTML = datos.responseText;
    }
}
```

```
<body>
  <form>
  <input type="button" value="Obtener datos" onclick='pedir_datos("datos.php")'>
  </form>
  <div>  </div><br/>
  <div>  </div><br/>
  <div>  </div><br/>
  <div>  </div><br/>
</body>
```

- Escribe una página con el siguiente contenido. Haz una captura de pantalla de cada mensaje de alerta que aparezca.

```
<html>
<head>
<script type="text/javascript">

  var direccion = new Array();
  var nota      = {};

  direccion[0]  = "Avda. ";
  direccion[1]  = "Estudiantes ";
  direccion[2]  = 99;

  nota["Rocio"] = "Sobresaliente";
  nota["Eva"]   = "Notable";
  nota["Jairo"] = "Suficiente";

  alert (direccion[0] + direccion[1] + direccion[2]);

  alert (nota["Rocio"] + " <--> Rocio \n"
       + nota["Eva"]   + " <--> Eva \n"
       + nota["Jairo"] + " <--> Jairo \n" );

</script>
</head>

<body></body>
</html>
```

- Modificar el ejercicio anterior para que aparezca la dirección del alumno/a y su nota en el mismo cuadro de alerta. Debe hacerse 3 veces (3 alumnos).

Funciones y Eventos

El código Javascript puede agruparse dentro de funciones. Por ejemplo mediante

```
function pedir_datos (url) {
  datos = new XMLHttpRequest();
  datos.onreadystatechange = mostrar_datos;
  datos.open("GET", url, true);
  datos.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  datos.setRequestHeader("Connection", "close");
  datos.send(null);
}
```

hemos agrupado varias sentencias dentro de la función *pedir_datos (url)*.

A menudo es útil asociar la ejecución del código de una función al disparo de un evento. Es decir cuando se produce determinado evento (pulsación del ratón, pulsación de una tecla, movimiento del ratón, carga de la página, …)

No todos los elementos de una página soportan todos los eventos. A continuación se muestra una breve lista:

Manejador de evento	**Objetos para los que está definido**
onAbort	Image
onBlur	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea, window
onChange	FileUpload, Select, Text, Textarea
onClick	Button, document, Checkbox, Link, Radio, Reset, Submit
onDblClick	document, Link
onDragDrop, onLoad, onUnload	window
onError	Image, window
onFocus	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea, window
onKeyDown	document, Image, Link, Textarea
onKeyPress	document, Image, Link, Textarea
onKeyUp	document, Image, Link, Textarea
onLoad	Image, Layer, window
onMouseDown	Button, document, Link
onMouseMove	Ninguno (debe asociarse a uno)
onMouseOut	Layer, Link
onMouseOver	Layer, Link
onMouseUp	Button, document, Link
onMove	window

onReset, onSubmit	Form
onResize	window
onSelect	Text, Textarea

Nosotros, en nuestro ejemplo, hemos asociado, de **forma implícita,** la función **pedir_datos(url)** al disparo del evento **onSubmit** en el formulario.

```
<form action="javascript:pedir_datos('datos.php');">
    <input type="submit" value="Obtener datos">
</form>
```

De **forma explícita** sería:

```
<form action="#" onsubmit="pedir_datos('datos.php'); return false;">
    <input type="submit" value="Obtener datos">
</form>
```

A continuación se muestran otros posibles eventos de disparo.

- Al pulsar un botón normal

```
<body>
  <input type="button" value="Obtener datos" onclick="pedir_datos('datos.php')">
  <div id='resultado'> </div>
</body>
```

- Al cargar la página

```
<body onload="pedir_datos('datos.php')">
  <div id='resultado'> </div>
</body>
```

- Al pasar por encima de la capa (DIV) del resultado

```
<body>
   <div id='resultado' onmouseover="pedir_datos('datos.php')"> </div>
</body>
```

- Al cambiar el valor de un cuadro de selección

```
<body>
   <select onchange="pedir_datos('datos.php')">
        <option> Pedir datos    </option>
        <option> Pedir otra vez </option>
        <option> Volver a pedir </option>
   </select>
   <div id='resultado'> </div>
</body>
```

Actividades propuestas

- Modifica ver_ajax.html de modo que pruebes el funcionamiento de los 4 casos anteriores.

El objeto XMLHttpRequest

Obteniendo los datos desde el servidor

Para obtener los datos desde el servidor utilizamos la **función *pedir_datos()***.

```
var datos;

function pedir_datos (url) {
   datos = new XMLHttpRequest();
   datos.onreadystatechange = mostrar_datos;
   datos.open("GET", url, true);
   datos.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
   datos.setRequestHeader("Connection", "close");
   datos.send(null);
}
```

La función pedir_datos () la asociaremos a un evento, de modo que se ejecute cuando a nosotros nos interese. Este evento podría ser la carga de la página, la pulsación de un botón del ratón, el cambio en un select, …

La función pedir_datos () hace lo siguiente:

- Crea un objeto XMLHttpRequest

```
   datos = new XMLHttpRequest();
```

- Asocia la función mostrar_datos () para se ejecute cuando se produzca un cambio de estado en dicho objeto.

```
   datos.onreadystatechange = mostrar_datos;
```

- Abre la petición de URL.

```
   datos.open("GET", url, true);
```

- Envía las cabeceras.

```
   datos.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
   datos.setRequestHeader("Connection", "close");
```

- Envía los parámetros.

```
   datos.send(null);
```

Observaciones

Observa que necesitamos una **variable global *datos*** donde se almacenará la información que recibamos. Dicha variable será utilizada después por la función mostrar_datos(). Podría ser necesaria asimismo para otras funciones.

Pasando datos al servidor

Suele ser habitual cuando se hace una petición, que en dicha petición se envíe información al servidor desde el cliente. Esto suele hacerse en la forma *variable1=valor1&variable2=valor2...*

La forma de pasar parámetros depende de si utilizamos el método GET o el método POST.

Ambos métodos son más o menos equivalentes, aunque el método POST es especialmente útil cuando se trabaja con formularios.

- **Método GET**
  Con el método GET los parámetros se pasan en la misma URL.

```
datos.open("GET", "datos.php?curso=SMR2&tutor=José+Antonio", true);
```

```
datos.send(null);
```

- **Método POST**
  Con el método POST los parámetros se pasan con la función datos.send().

```
datos.open("POST", "datos.php", true);
```

```
datos.send("curso=SMR2&tutor=José+Antonio");
```

Ejemplos:

(1) GET tradicional

get.html

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
</head>
<body>
   <form method="get" action="get.php">
      <label>Curso: </label>
      <select name="curso">
            <option value="SMR1"         > SMR1 </option>
            <option value="SMR2" selected> SMR2 </option>
      </select>
      <label>Tutor: </label>
      <input type="text"   name="tutor"  value="José Antonio" />
      <input type="submit" name="submit" value="OK" />
   </form>
</html>
```

get.php

```
<?php
header('Content-type: text/xml');
echo '<?xml version="1.0" encoding="UTF-8" ?>
<mensaje>El tutor del curso '.$_GET[curso].' es '.$_GET[tutor].'</mensaje>';
?>
```

Fíjate en la **URL** resultante:
http://localhost/ajax/**get.php?curso=SMR2&tutor=José+Antonio&submit=OK**

(2) POST tradicional

post.html

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
</head>
<body>
   <form method="post" action="post.php">
      <label>Curso: </label>
      <select name="curso">
            <option value="SMR1"         > SMR1 </option>
            <option value="SMR2" selected> SMR2 </option>
      </select>
      <label>Tutor: </label>
      <input type="text"   name="tutor"  value="José Antonio" />
      <input type="submit" name="submit" value="OK" />
   </form>
</html>
```

post.php

```
<?php
header('Content-type: text/xml');
echo '<?xml version="1.0" encoding="UTF-8" ?>
<mensaje>El tutor del curso '.$_POST[curso].' es '.$_POST[tutor].'</mensaje>';
?>
```

(3) GET con AJAX

get_ajax.html

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <link rel="stylesheet" type="text/css" href="estilo.css" />
  <script type="text/javascript">

  var datos;

  function pedir_datos (url, params) {
    datos = new XMLHttpRequest();
    datos.onreadystatechange = mostrar_datos;
    datos.open("GET", url+'?'+params, true);
    datos.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    datos.setRequestHeader("Connection", "close");
    datos.send(null);
  }


  function mostrar_datos () {
    if (datos.readyState == 4)
      document.getElementById('resultado').innerHTML = datos.responseText;
  }


  function iniciar() {
    var url    = 'get.php';
    var params = f.curso.name +'='+ f.curso.value +'&'+
                 f.tutor.name +'='+ f.tutor.value;
    pedir_datos(url, params);
  }

</script>
</head>
<body>
   <form name="f" method="get" action="javascript: iniciar ()">
      <label>Curso: </label>
      <select name="curso">
            <option value="SMR1"         > SMR1 </option>
            <option value="SMR2" selected> SMR2 </option>
      </select>
      <label>Tutor: </label>
      <input type="text" name="tutor" value="José Antonio" />
      <input type="submit" name="submit" value="OK" />
   </form>
   <div id="resultado"> </div>
</html>
```

(4) POST con AJAX

post_ajax.html

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <link rel="stylesheet" type="text/css" href="estilo.css" />
  <script type="text/javascript">

  var datos;

  function pedir_datos (url, params) {
    datos = new XMLHttpRequest();
    datos.onreadystatechange = mostrar_datos;
    datos.open("POST", url, true);
    datos.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    //datos.setRequestHeader("Content-length", params.length);
    datos.setRequestHeader("Connection", "close");
    datos.send(params);
  }


  function mostrar_datos () {
    if (datos.readyState == 4)
      document.getElementById('resultado').innerHTML = datos.responseText;
  }


  function iniciar() {
    var url    = 'post.php';
    var params = f.curso.name +'='+ f.curso.value +'&'+
                 f.tutor.name +'='+ f.tutor.value;
    pedir_datos(url, params);
  }


</script>
</head>
<body>
   <form name="f" method="post" action="javascript: iniciar()">
      <label>Curso: </label>
      <select name="curso">
            <option value="SMR1"          > SMR1 </option>
            <option value="SMR2" selected> SMR2 </option>
      </select>
      <label>Tutor: </label>
      <input type="text" name="tutor" value="José Antonio" />
      <input type="submit" name="submit" value="OK" />
   </form>
   <div id="resultado"> </div>
</html>
```

Actividades propuestas

- Editar los 4 ejemplos anteriores y comprobar su correcto funcionamiento.
- Modifica el último ejemplo para que devuelva una lista de alumnos distinta dependiendo del curso.

El objeto document

Visualizando los datos dentro del cliente

Para mostrar los datos en el cliente utilizamos la **función *mostrar_datos()***.

```
function mostrar_datos () {
   if (datos.readyState == 4)
       document.getElementById('resultado').innerHTML = datos.responseText;
}
```

Otra forma algo más detallada es

```
function mostrar_datos () {
   if (datos.readyState == 4 && datos.status == 200)
       document.getElementById('resultado').innerHTML = datos.responseText;
}
```

La condición

```
   if (datos.readyState == 4 && datos.status == 200)
```

comprueba que existe una respuesta completa del servidor (datos.readyState == 4) y que dicha respuesta es satisfactoria (datos.status == 200).

Los valores para **datos.readyState** son:

- **0**. Indica que el objeto existe pero no ha sido aún inicializado.
- **1**. El objeto ha sido inicializado tras la llamada al método open().
- **2**. La petición ha sido enviada.
- **3**. Se ha recibido una respuesta parcial a la petición.
- **4**. Se ha recibido una respuesta completa y la conexión ha sido cerrada.

Los valores para **datos.status** más frecuentes suelen ser:

- **200**. Indica que la petición HTTP se llevó a cabo satisfactoriamente.
- **403.** Indica que el servidor entendió la petición HTTP, pero no puede llevarla a cabo. Por ejemplo, debido a un problema de permisos.
- **404.** Indica que el recurso solicitado no existe.
- **500.** Indica que se ha producido un error interno en el servidor al intentar responder a la petición.

La sentencia

```
       document.getElementById('resultado').innerHTML = datos.responseText;
```

coge los datos en formato texto (datos.responseText) y vuelca (innerHTML) su contenido en el elemento identificado como 'resultado'.

Tratando los datos en el cliente

Imaginemos que deseamos mostrar los datos en orden inverso a como los obtenemos, o mostrar sólo las líneas pares, o mostrar sólo las que cumplan cierta condición. Para hacer esto necesitamos obtener los datos en formato XML y no en formato Texto como hasta ahora.

Mediante **datos.responseXML** obtendremos un objeto XML listo para poder ser interpretado.

La siguiente función nos muestra los datos en el orden recibido.

```
function mostrar_datos () {
    var a = new Array ();

    if (datos.readyState == 4) {
       a = datos.responseXML.getElementsByTagName ("alumno");
       for (i = 0; i < a.length; i++)
          document.getElementById('resultado').innerHTML
               += a[i].firstChild.data + "<br/>";
    }
}
```

La siguiente función nos muestra los datos en orden inverso.

```
function mostrar_datos () {
    var a = new Array ();

    if (datos.readyState == 4) {
       a = datos.responseXML.getElementsByTagName ("alumno");
       for (i = a.length-1; i >= 0; i--)
          document.getElementById('resultado').innerHTML
              += a[i].firstChild.data + "<br/>";
    }
}
```

Los datos de cada alumno se guardan en la variable local ***a[i]***. Para acceder a los datos entre la etiqueta de inicio y la de cierre se utiliza

```
a[i].firstChild.data
```

En el caso de disponer de atributos, también es posible acceder a ellos. Supongamos que tenemos el siguiente contenido XML:

```
<alumnos>
   <alumno edad="18" sexo="V">José Antonio</alumno>
   <alumno edad="17" sexo="M">Susana</alumno>
   <alumno edad="19" sexo="M">Ana</alumno>
</alumnos>
```

La siguiente función nos mostrará los datos entre las etiquetas de inicio y cierre y el valor de los atributos *edad* y *sexo.*

```
function mostrar_datos () {
   var a = new Array ();

   if (datos.readyState == 4) {
      a = datos.responseXML.getElementsByTagName ("alumno");
      for (i = 0; i < a.length; i++)
         document.getElementById('resultado').innerHTML
               += a[i].firstChild.data
               +  " edad = " + a[i].getAttribute ('edad')
               +  " sexo = " + a[i].getAttribute ('sexo')
               +  " <br/>";
   }
}
```

Para acceder al valor del atributo edad se utiliza

```
a[i].getAttribute ('edad')
```

Para acceder a cualquier otro atributo se procede de idéntica forma.

Actividades propuestas

- Modificar el archivo datos.php para que aparezcan al menos 10 alumnos/as con los atributos de edad y sexo.
- Modificar el archivo ver_ajax.html para que se muestren sólo las líneas pares.
- Modificar el archivo ver_ajax.html para que se muestren sólo los alumnos/as mayores de 18 años.
- Modificar el archivo ver_ajax.html para que se muestren sólo las alumnas.

Compatibilidad con distintos navegadores

El objeto **XMLHtttpRequest** no se utiliza por todos los navegadores, de hecho Internet Explorer, el navegador más utilizado en la actualidad, utiliza el objeto **ActiveXObject**.

Por lo tanto la función **pedir_datos (url)** debe ser modificada de la forma que se muestra a continuación. Esta función también realiza el tratamiento de las posibles excepciones en tiempo de ejecución.

```
var datos;

function pedir_datos(url) {
  datos = false;

  // Navegadores Firefox, Safari, ... / XMLHttpRequest
  if(window.XMLHttpRequest) {
    try { datos = new XMLHttpRequest(); }
    catch(e) { datos = false; }
  }

  // Navegador IE de Windows / ActiveXObject
  else if(window.ActiveXObject){
    try { datos = new ActiveXObject("Msxml2.XMLHTTP"); }
    catch(e) {
      try { datos = new ActiveXObject("Microsoft.XMLHTTP"); }
      catch(e) { datos = false; }
    }
  }

  if(datos) {
      datos.onreadystatechange = mostrar_datos;
      datos.open("GET", url, true);
      datos.setRequestHeader("Content-type",
                             "application/x-www-form-urlencoded");
      datos.setRequestHeader("Connection", "close");
      datos.send("");
  }
}
```

Actividades propuestas

- Elaborar una aplicación AJAX que sea compatible con todos los navegadores y que emplee los conocimientos vistos en este tema.

Referencias

Javascript

- **http://en.wikipedia.org/wiki/JavaScript**
- **http://www.webdesigntoolslist.com/2009/04/webmastertools/javascript-cheat-sheets-quick-reference-guides-for-javascript-webmasters-coders-web-developers-designers/**
- **http://es.wikipedia.org/wiki/JavaScript**
- **http://www.elcodigo.net/cgi-bin/DBread.cgi?tabla=herramientas&campo=0&clave=49&info=1**

AJAX

- **http://en.wikipedia.org/wiki/Ajax_(programming)**
- **http://www.noupe.com/ajax/how-ajax-works.html**
- **http://es.wikipedia.org/wiki/AJAX**
- **www.huibert-aalbers.com/IT_Insight/Spanish/PDF/ITI006Sp-AJAX.pdf**
- **http://www.elrincondeajax.com/manual-ajax/**
- **http://www.ajaxhispano.com/quien-usa-ajax.html**
- **http://www.ajaxhispano.com/ajax-nuevo-acercamiento-aplicaciones-web.html**