

AJAX y PHP

Laboratorio de Aplicaciones Telemáticas

Jesús Arias Fisteus

jaf@it.uc3m.es

Curso 2007/2008

Universidad Carlos III de Madrid

AJAX

(Asynchronous JavaScript and XML)

AJAX (Asynchronous JavaScript and XML)

- Nombre que se aplica al uso combinado de:
 - JavaScript.
 - XMLHttpRequest.

Introducción a JavaScript

- JavaScript:
 - Lenguaje de programación interpretado.
 - Débilmente tipado.
 - Sintácticamente parecido a C, C++ y Java.
 - Utilizado habitualmente en navegadores Web (*client-side JavaScript*) para mejorar la interactividad de las páginas.
 - Estandarizado bajo el nombre de ECMAScript.

Interactividad en el navegador con JavaScript

- *Client-side* JavaScript hace interactivo el documento (X)HTML mediante, principalmente:
 - Manejadores de eventos:
 - Se puede ejecutar código específico (manejadores) cuando se cargue/cierre la página, el usuario interaccione con elementos de la misma o periódicamente.
 - Modificación dinámica del documento:
 - `document.write()` permite escribir directamente el contenido del documento.
 - API de DOM: acceso lectura/escritura a la estructura del documento (HTML/XML).

Inclusión de JavaScript en (X)HTML

```
<!-- directamente con el elemento script
      (en la cabecera o en el cuerpo del documento) -->
<script type="text/javascript">
var d = new Date();
document.write(d.toLocaleString());
</script>

<!-- desde un recurso externo -->
<script src="scripts/util.js" type="text/javascript" />

<!-- desde un manejador de eventos de (X)HTML -->
<input type="button" value="Change" onclick="changeName()" />
<p onmouseover="showHelp('p1')">...</p>
```

Ejemplo: API DOM (I)

```
var n = document.documentElement; // objeto Node
var children = n.childNodes; // objeto NodeList
var head = children[0];
var body = children[1];

// contar el número de tablas
var tables = document.getElementsByTagName("table");
alert("El documento contiene " + tables.length + " tablas.");

// acceso a un párrafo <p id="specialParagraph">...</p>
var paragraph = document.getElementById("specialParagraph");
```

Ejemplo: API DOM (II)

```
// modificar un atributo de un elemento
var headline = document.getElementById("headline");
// alternativa genérica:
headline.setAttribute("align", "center");
// alternativa para atributos estándar del elemento
headline.align = "center";

// añadir un elemento
var p = document.getElementById("headline");
var i = document.createElement("i");
i.class = "resaltado";
i.appendChild(document.createTextNode("Texto en cursiva"));
p.appendChild(i);
```

Programación de HTTP desde JavaScript

- En principio, es el navegador el que genera peticiones HTTP y procesa las respuestas:
- JavaScript puede forzar peticiones estableciendo el atributo *src* en *img*, *iframe* y *script*, pero tiene problemas de portabilidad entre navegadores.
- La API XMLHttpRequest permite de forma más sencilla a JavaScript realizar peticiones HTTP y procesar sus respuestas.

Uso de XMLHttpRequest

- Proceso de tres etapas:
 1. Creación del objeto XMLHttpRequest.
 2. Especificación y envío del mensaje HTTP al servidor.
 3. Recepción (síncrona o asíncrona) de la respuesta del servidor.
- A pesar del nombre, no es estrictamente necesario que los mensajes HTTP intercambiados codifiquen los datos con XML.

Peticiones síncronas:

- La función `send` retorna una vez se haya recibido la respuesta.
- La página queda bloqueada hasta que se recibe la respuesta.

Ejemplo de petición síncrona

```
// creación de un objeto XMLHttpRequest (no portable)
var request = new XMLHttpRequest();

// especificación de método, URL y petición síncrona
request.open("GET", url, false);

// envío (sin cuerpo de la petición por ser GET)
request.send(null);

// obtención de la respuesta síncrona
if (request.status == 200) {
    var response = request.responseText;

    // como alternativa, si es una respuesta XML
    var responseXML = request.responseXML;
} else {
    // manejar el error
    ...
}
```

Peticiones asíncronas:

- La función `send` retorna inmediatamente, sin esperar la respuesta.
- Se registra una función de *callback* que se invoca cada vez que cambia el estado de la petición (propiedad `readyState`):
 - `readyState == 0`: sin inicializar.
 - `readyState == 1`: conexión establecida.
 - `readyState == 2`: petición recibida.
 - `readyState == 3`: respuesta en proceso.
 - `readyState == 4`: respuesta recibida.

Ejemplo de petición asíncrona

```
// creación de un objeto XMLHttpRequest (no portable)
var request = new XMLHttpRequest();

// establecimiento de una función de callback
request.onreadystatechange = function()
{
    if(request.readyState == 4) {
        if(request.status == 200) {
            alert("Received: " + req.responseText);
        } else {
            alert("Error: returned status code " + request.status
                + " " + request.statusText);
        }
    }
};

// especificación de método, URL y petición asíncrona
request.open("GET", url, true);

// envío (sin cuerpo de la petición por ser GET)
request.send(null);
```

Creación de XMLHttpRequest portable

```
var request = null;
try {
    request = new ActiveXObject("Msxml2.XMLHTTP");
} catch (b) {
    try {
        request = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (c) {
        request = null;
    }
}
if(!request && typeof XMLHttpRequest!="undefined") {
    request = new XMLHttpRequest;
}
```

Referencias

- David Flanagan. “JavaScript: The Definitive Guide” (5th Ed.) O’Reilly.



PHP

PHP

- Lenguaje interpretado.
- Desarrollo rápido.
- API muy extensa.
- Código libre

Usos de PHP

- Aplicaciones Web ejecutadas en el servidor Web (código incrustado en documentos HTML).
- Aplicaciones Web como scripts CGI.
- Scripts de línea de comandos.
- Aplicaciones gráficas.

PHP en HTML

- Permite programar aplicaciones Web que se ejecutan en el servidor.
- Normalmente se ejecuta en un servidor Apache.
- Admite cualquiera de los principales gestores de bases de datos (aunque MySQL es el preferido).

Ejemplo: hola mundo

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php echo '<p>Hello World</p>'; ?>
  </body>
</html>
```

Resultado del ejemplo

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <p>Hello World</p>
  </body>
</html>
```

Dos tipos de comillas

- Las cadenas entre comillas simples se vuelcan literalmente.
- En las cadenas entre comillas dobles se evalúan variables, caracteres especiales tipo `\n`, etc.

```
$txt="Juan";
```

```
/* La siguiente línea vuelca "Yo me llamo Juan :)" */  
echo "Yo me llamo $txt :)";
```

```
/* Pero esta línea vuelca "Yo me llamo $txt :)" */  
echo 'Yo me llamo $txt :)';
```

Ejemplo: variables

```
<html>
  <body>
    <?php
      /* asigna valores a dos variables */
      $txt1="Hello World";
      $txt2="1234";

      /* escribe la concatenación de las variables */
      echo $txt1 . " " . $txt2 ;

      /* equivalente a: */
      echo "$txt1 $txt2";
    ?>
  </body>
</html>
```

Ejemplo: sentencias condicionales

```
<html>
  <body>
    <?php
      $d=date("D");
      if ($d=="Fri") {
        echo "Hello!<br />";
        echo "Have a nice weekend!";
        echo "See you on Monday!";
      } else
        echo "Have a nice day!";
    ?>
  </body>
</html>
```

Ejemplo: arrays numéricos

```
<?php
/* una forma de inicializar el array */
$names = array('Peter','Quagmire','Joe');

/* otra forma equivalente de hacerlo */
$names[0] = 'Peter';
$names[1] = 'Quagmire';
$names[2] = 'Joe';

/* acceso a posiciones del array */
echo $names[1] . " and " . $names[2] . " are
      ". $names[0] . "'s neighbors";
?>
```

Ejemplo: arrays asociativos

```
<?php
/* una forma de inicializar el array */
$ages = array('Peter'=>32, 'Quagmire'=>30, 'Joe'=>34);

/* otra forma equivalente de hacerlo */
$ages['Peter'] = 32;
$ages['Quagmire'] = 30;
$ages['Joe'] = 34;

/* acceso a una posición del array */
echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

Ejemplo: bucles

- Proporciona bucles *for*, *while* y *do/while* similares a C, C++ y Java.
- Adicionalmente, proporciona un bucle *foreach* para recorrer datos de tipo *array*:

```
<?php
$arr=array("one", "two", "three");
foreach ($arr as $value)
{
    echo "Value: " . $value . "<br />";
}
?>
```

Ejemplo: funciones

```
<?php
function writeMyName($fname,$punctuation)
{
    echo $fname . " Refsnes" . $punctuation . "<br />";
}
echo "My name is ";
writeMyName("Kai Jim", ".");
echo "My name is ";
writeMyName("Hege", "!");
echo "My name is ";
writeMyName("Ståle", "...");
?>
```

■ Resultado:

```
My name is Kai Jim Refsnes.
My name is Hege Refsnes!
My name is Ståle Refsnes...
```

Ejemplo: funciones con valor de retorno

```
<?php
function add($x,$y)
{
    $total = $x + $y;
    return $total;
}
echo "1 + 16 = " . add(1,16)
?>
```

Contenedores de variables útiles (I)

- `$_GET['varname']`: variables de la petición GET.
- `$_POST['varname']`: variables de la petición POST.
- `$_REQUEST['varname']`: variables de la petición (independiente del método).
- `$_SESSION['varname']`: variables de sesión.
- `$_COOKIE['varname']`: variables almacenadas en cookies.

Contenedores de variables útiles (I)

- `$_SERVER['varname']`: variables almacenadas desde el servidor.
- `$_FILE['varname']`: variables almacenadas a partir de un fichero externo.

Ejemplo: formularios

```
<!-- HTML del formulario -->
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

```
<!-- welcome.php -->
<html>
<body>
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
```

Ejemplo: sesiones

```
<?php  
  
session_start();  
if(isset($_SESSION['views']))  
    $_SESSION['views']=$_SESSION['views']+1;  
else  
    $_SESSION['views']=1;  
echo "Views=". $_SESSION['views'];  
?>
```

Ejemplo: acceso a MySQL (I)

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
$sql="INSERT INTO person (FirstName, LastName, Age)
VALUES
('$_POST[firstname]','$_POST[lastname]','$_POST[age]')";
if (!mysql_query($sql,$con))
{
    die('Error: ' . mysql_error());
}
echo "1 record added";
mysql_close($con);
?>
```

Ejemplo: acceso a MySQL (II)

```
<?php
$con = mysql_connect("localhost", "peter", "abc123");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM person");

/* continúa en la siguiente transparencia */
```

Ejemplo: acceso a MySQL (III)

```
/* continuación de la transparencia anterior */

echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>";
while($row = mysql_fetch_array($result))
{
echo "<tr>";
echo "<td>" . $row['FirstName'] . "</td>";
echo "<td>" . $row['LastName'] . "</td>";
echo "</tr>";
}
echo "</table>";
mysql_close($con);
?>
```

Referencias

- <http://www.php.net/>
- <http://www.php.net/manual/en/>
- <http://www.w3schools.com/php/>