POLITÉCNICA "Ingeniamos el futuro"

CAMPUS DE EXCELENCIA INTERNACIONAL

MiW

AppStorming

Edición para

Taller HTML-CSS-JavaScript

Ajax: Asynchronous JavaScript and XML

Santiago Alonso Villaverde

FENW

Índice

Ajax: Asynchronous JavaScript and XML

- Conjunto de técnicas de desarrollo web que desde el cliente interactúan con el servidor en segundo plano o asíncrono para crear aplicaciones web interactivas
- Antecedentes:
  - Objetos iframe (1996) y layer (1997)
  - Remote Scripting (Microsoft - 1998)
  - Applets Java, Frames ocultos, plug-ins
  - XMLHttpRequest (2002)
  - Término AJAX (2005)

Ajax: Tecnologías

- HTML y CSS (hojas de estilo) para el diseño de la presentación
- DOM (Document Object Model) para mostrar e interactuar con los datos presentados
- JSON (ó XML) para el intercambio, manejo y presentación de los datos con el servidor
- XMLHttpRequest para el intercambio asíncrono con el servidor
- JavaScript para unir todas las tecnologías anteriores

Ajax: Usos

- Validación de formularios en tiempo real
- Códigos de autocompletado
- Carga de páginas en background
- Interfaces de usuario más complicados (menús, calendarios, editores,…)
- Validaciones parciales de página
- Mashups (aplicaciones de terceros: Google)
- Desarrollo de Páginas como Aplicaciones (Portlets)

Ajax: Desventajas

- Indexación desde los motores de búsqueda
- Depende de JavaScript (habilitado)
- Uso de Marcadores de Página
- Acceso al Historial de Navegación
- Complejidad, estandarización/dependencia de Javascript, dificultad en la trazabilidad,..

Ajax: Objeto XMLHttpRequest (I)

- La comunicación entre cliente y servidor se realiza a través del objeto XMLHttpRequest
- Propiedades del objeto XMLHttpRequest:
  - readyState Devuelve el estado del objeto (0 = sin inicializar, 1 = abierto, 2 = cabeceras recibidas, 3 = cargando y 4 = completado)
  - responseText Devuelve la respuesta como una cadena
  - responseXML Devuelve la respuesta como XML
  - status Devuelve el estado como un número (p. ej. 404 para "Not Found" y 200 para "OK").
  - statusText Devuelve el estado como una cadena (p. ej. "Not Found" o "OK")

Objeto XMLHttpRequest versión 2

- La nueva versión de XMLHttpRequest es más sencilla de manejar
- XMLHttpRequest v2 incluye:
  - onload - Se dispara cuando la propiedad readyState == 4, con lo que se puede evitar esa comprobación en el procesamiento de la respuesta
  - response - Contendrá la respuesta después de ser tratada dependiendo del valor de responseType. Es el equivalente a responseText o responseXML
  - responseType – Si se asigna el valor "json" el objeto realiza la llamada al método JSON.parse de manera automática, evitando que el programador lo haga.

Ajax: Objeto XMLHttpRequest (II)

- Métodos de XMLHttpRequest
  - open(método,URL[,asíncrono[,usuario[,clave]]]) Especifica las características de una petición:
    - método Toma los valores "GET", "POST", o "HEAD"
    - URL Destino de la petición
    - asíncrono Si toma valor true, el script continúa la ejecución después del método send(), y si es false el script se detiene hasta que termine la operación
    - usuario clave (optativo) para autenticar

Ajax: Objeto XMLHttpRequest (III)

- Métodos de XMLHttpRequest
  - abort() Cancela la petición en curso
  - getAllResponseHeaders() Devuelve una cadena con todas las cabeceras HTTP como una cadena.
  - getResponseHeader(identificadorCabecera) Devuelve una cadena con el valor de la cabecera HTTP especificada.
  - send([datos]) Envía al servidor la petición. Opcionalmente se pueden enviar datos (para POST).
  - setRequestHeader(identificador,valor) Añade un par identificador/valor a la cabecera HTTP a enviar.
- Eventos de XMLHttpRequest
  - onreadystatechange Se dispara cada vez que cambia de estado el objeto.

Ajax: Procedimiento General (I)

- Instanciación del objeto XMLHttpRequest

```
var peticion;

if (window.XMLHttpRequest)
  {// codigo para IE9+, Firefox, Chrome, Opera, Safari
  peticion = new XMLHttpRequest();
  }
else
  {// codigo para IE6, IE5
  peticion = new ActiveXObject("Microsoft.XMLHTTP");
  }
```


Ajax: Procedimiento General (II)

2) Configuración del tratamiento de la respuesta

```
peticion.onload = TrataRespuesta;

function TrataRespuesta() {
  var Respuesta;
  if (peticion.status == 200){
    respuesta = peticion.responseText;
    //Tratamiento de la respuesta
   } //del if 200
  else
    alert("Ocurrio un problema con la URL.");
  } // de la función TrataRespuesta
```

Ajax: Procedimiento General (III)

3) Configuracion de la petición de XMLHttpRequest

```
peticion.open('GET', url, true)
```

4) Envio de la peticion XMLHttpRequest

```
peticion.send(null);
```


Ajax: Procedimiento General (IV): Paso de datos

a) Si la petición es por **"GET"**:

```
peticion.open('GET', url, true);
peticion.send(null);
```

Los datos deben incluirse en la url (urlencoded):

http://servidor/aplicacion?param1=valor1&param2=valor2&...

b) Si la petición es por **"POST"**:

```
peticion.open('POST', url, true)
```

Se debe indicar mediante la cabecera, después de abrirlo:

```
peticion.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
```

Los datos se incluyen en la petición "send" (urlencoded)

```
peticion.send(querystring);
```

JSON: *JavaScript Object Notation*

- Es un formato ligero de intercambio de datos
  {objeto: {"propiedad": "valor", "prop2": valor2" …}}
- Se puede definir cualquier estructura para almacenar datos (es muy flexible)
- Es mucho más fácil el tratamiento y recuperación de arrays y objetos
- Los lenguajes de servidor permiten el envío de datos codificados en JSON. En PHP:

```
$miarray = array('dato1','"dato2"','"dato3"','&dato4&');
echo  json_encode($miarray), "\n";
```

Llega el array codificado al cliente, que lo tiene que decodificar para obtener el objeto correspondiente

JSON: *JavaScript Object Notation*

- La estructura puede ser:
  - Objeto (colección de pares nombre/valor)
  - Lista de valores (arrays en la mayoría de lenguajes)
- Nombre/valor se separan por ":" y por comas entre ellos
  - "nombre1": "valor1", "nombre2": "valor2"
- Los objetos se encierran entre llaves "{ …. }"
  - {"nombre": "Santiago Alonso"}
- Los elementos del array se separan por comas
  - ["Santiago Alonso", "Javier Gil", "Jorge Tejedor"]
- Pueden crearse estructuras complejas:

```
{"profesores": [
        {"apellido":"Alonso", "nombre": "Santiago"},
        {"apellido":"Gil", "nombre": "Javier"},
        {"apellido":"Tejedor", "nombre": "Jorge"}
]}
```

JSON: *JavaScript Object Notation*

- Tipos de valores:
  - Números: pueden ser negativos y con punto decimal
  - Cadenas: se escriben entre comillas dobles
  - Booleanos: *true* o *false*
  - null: es un valor nulo
  - Vector (array): Lista ordenada de valores de cualquier tipo separados por comas. El vector va encerrado entre corchetes

```
{"clientes":[
  {"apellido":"Alonso", "gasto": 100, "es_habitual": true,
    "productos": ["P001", "P032", "P099"]},
  {"apellido":"Villaverde", "gasto": 10, "es_habitual": false,
    "productos": ["P008", "P012" ]},
  {"apellido":"Anguiano", "gasto": 10, "es_habitual": true,
    "productos": ["P100"]}
 ]}
```


PARSER JSON

- No usar eval() ya que es inseguro
- Solo admite cadenas JSON => más seguro
- Se incluirá en el próximo estándar ECMAScript el objeto JSON

```
dato_final = JSON.parse(dato_cadena);
dato_final = JSON.parse(dato_cadena, funcion);
```

función será una función que se aplica a cada clave/valor de dato_cadena y el valor se sustituye por su resultado

- JSON.stringify(objeto) devuelve una cadena con el formato JSON del objeto especificado

Json: Tratamiento de datos recibidos

```
var http_request = new XMLHttpRequest(),
      url = "http://laurlquedevuelvedatosenjson";
http_request.open("GET", url, true);
http_request.responseType = 'json';
http_request.onload = TrataRespuesta;
http_request.send();

function TrataRespuesta() {
 var respuesta;
  if (http_request.status == 200){
   respuesta = http_request.response;
   document.getElementById("dato").value = respuesta;
   }
  else
   alert("Ocurrio un problema con la URL.");
}
```


Objeto XMLHttpRequest

- ATENCIÓN:
  - Este objeto trabaja siempre con codificación de caracteres UTF-8

```
$datos = array('V.G.' => utf8_encode('Visión general'),
               'D.G.' => utf8_encode('Diseño Gráfico'),
               'P.D.' => utf8_encode('Patrones de Diseño')
               );

echo json_encode($datos), "\n";
```