# AJAX

Desarrollo de Aplicaciones en Entornos Web

Curso 2016/2017

# Contenido

- Concepto básicos de una aplicación web
  - Protocolo HTTP
- AJAX
- Ajax – Petición: XMLHttpRequest
- Ajax – Respuesta : procesamiento de datos
- Ajax y JSON

# La Web


Universidad de Murcia - Microsoft Internet Explorer
UNIVERSIDAD DE MURCIA
Navegador
Petición
http://www.um.es/index.html
HTTP
Respuesta
Servidor Web
HTML
Documento /index.html
Sistema Ficheros

# Sitio Web vs. Aplicación Web

- Aplicación web
  - "Sitio web donde las entradas del usuario (navegación y entrada de datos) afectan al **estado de un sistema de información**"
  - "Una aplicación web usa un sitio web como fachada de una aplicación tradicional"

# Protocolo HTTP

- Protocolo de comunicación entre navegador y servidor web
- Modelo **Petición/Respuesta**
- Es un protocolo **sin estado**:
  - Cada ciclo petición/respuesta es independiente.
- **Tipos de peticiones**:
  - **GET**: petición de un recurso
  - **POST**: envío de datos a un recurso
  - HEAD: petición cabeceras de un recurso
  - ...

# Petición HTTP

**GET /index.html HTTP/1.1**

Accept: image/gif, image/jpeg,
application/msword, */*

Accept-Language: es

Accept-encoding: gzip

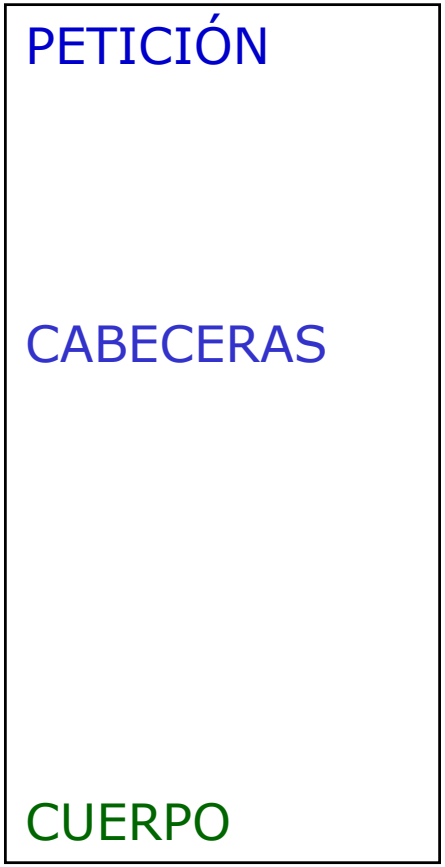
User-Agent: Mozilla/4.72

Host: dis.um.es

Connection: Keep-Alive

(línea en blanco)

CUERPO



# Respuesta HTTP

HTTP/1.1 200 OK

Date: Mon, 23 oct 2006 12:24:23 GMT

Connection: close

Server: Apache/1.3.24 (Linux)

Content-Length: 2032

Last-Modified: Sun, 22 oct 2006 10:03:11 GMT

Content-Type: text/html

Content-Language: es

Expires: Sun, 22 oct 2006 12:24:23 GMT

(línea en blanco)

<html>

<head> ...

Código de Estado

Cabeceras

Cabecera de Entidad

Cuerpo

# AJAX

- **Motivación**:
  - En una página de alta, comprobar si existe el identificador de usuario antes de enviar los datos al servidor.
  - En general, **obtener datos del servidor sin necesidad de recargar la página**.

- AJAX es la tecnología que permite el acceso al servidor de forma **asíncrona** en respuesta a eventos de la programación dinámica en HTML.

- Supone un **punto de inflexión en el desarrollo de aplicaciones web**.

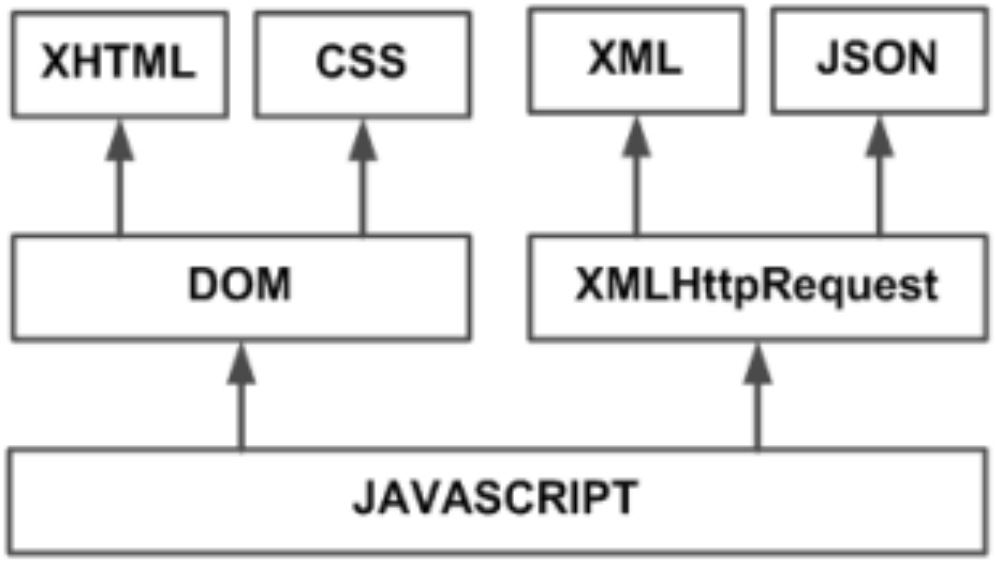
# AJAX

- **AJAX**: ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML*

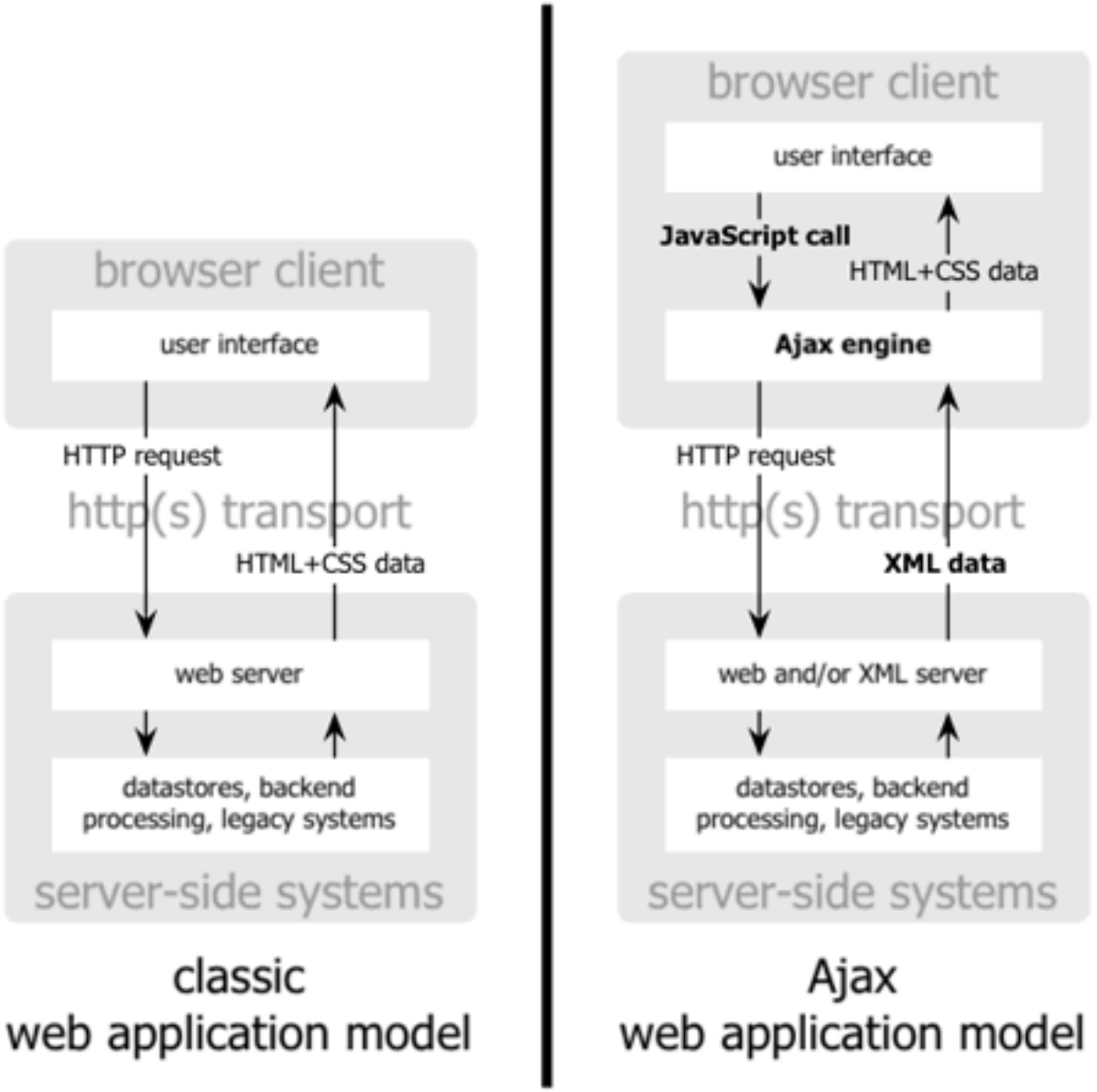
- **Inicialmente** AJAX representaba el soporte de los navegadores para acceder, de modo asíncrono, a datos en el servidor utilizando *JavaScript*.

- **Actualmente** *también* se entiende por AJAX las **bibliotecas de componentes gráficos** programados con HTML Dinámico.

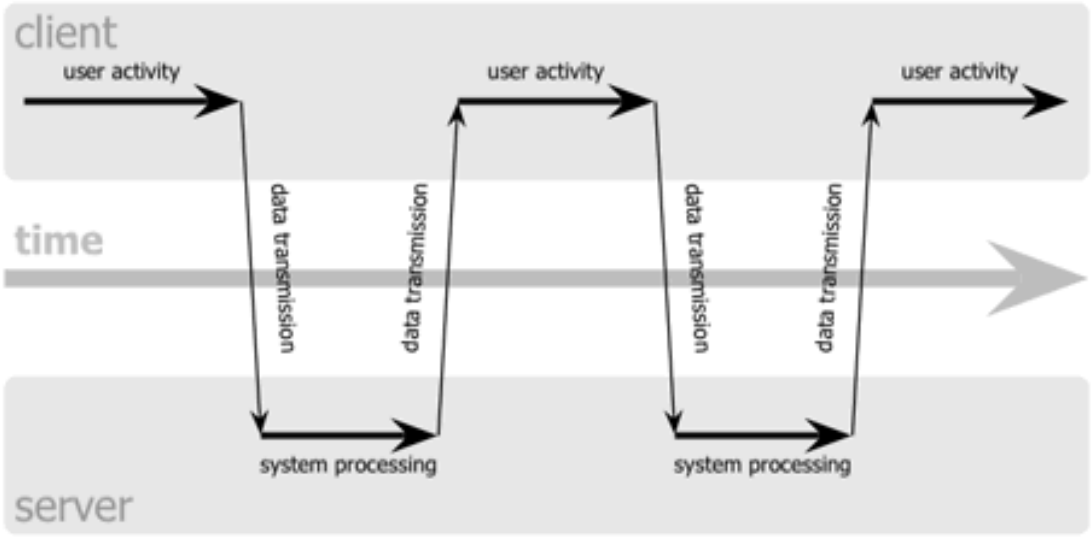
# AJAX



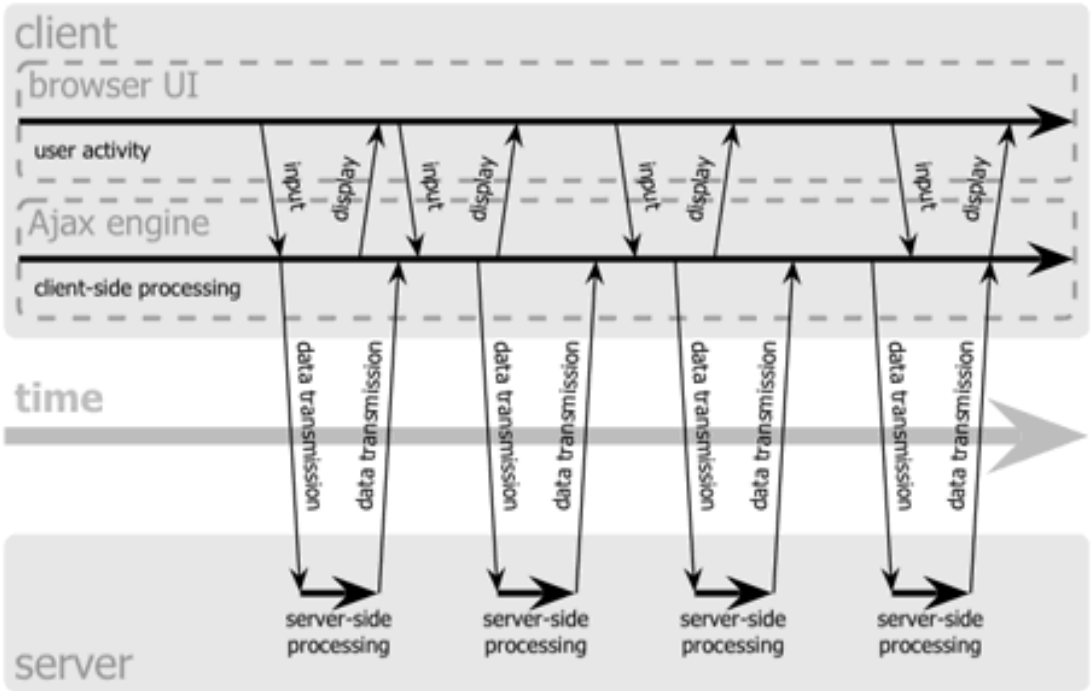
# AJAX


browser client
user interface
HTTP request
http(s) transport
HTML+CSS data
web server
datastores, backend
processing, legacy systems
server-side systems
classic
web application model
browser client
user interface
JavaScript call
HTML+CSS data
Ajax engine
HTTP request
http(s) transport
XML data
web and/or XML server
datastores, backend
processing, legacy systems
server-side systems
Ajax
web application model

classic web application model (synchronous)




Ajax web application model (asynchronous)



# AJAX

- Las peticiones AJAX utilizan un objeto **XMLHttpRequest**.
- Permite realizar con *JavaScript* una petición HTTP al servidor y esperar a que se reciban los datos.
- La clave de la programación con AJAX es la **petición asíncrona de datos al servidor**:
  - Se solicitan datos al servidor, pero el navegador no queda bloqueado a la espera.
  - Cuando llegan los datos, una función *JavaScript* se encarga de procesarlos.

# AJAX

- La **función** que recibe los datos del servidor se conoce como **“callback”.**

- La función callback es llamada cada vez que cambia el estado de la petición HTTP:
  - Cuando el **estado de la petición** es *completo* (4) y el **código** de respuesta de **HTTP** es *correcto* (200) se pueden obtener los datos de la petición en texto normal o estructurado en un árbol DOM.

# AJAX – Petición HTTP

- Los navegadores antiguos no soportan los objetos `XMLHttpRequest`.
- **Solución**:

```
if (typeof XMLHttpRequest != "undefined") {
  req = new XMLHttpRequest();
} else if (window.ActiveXObject) {
  req = new ActiveXObject("Microsoft.XMLHTTP");
}
```

- La **variable** "req" es **global**. Almacena el objeto de la petición y permite a la función callback obtener los datos.

# AJAX

- Ejemplo '***Hola Mundo***':

La petición HTTP y la descarga de los contenidos del archivo (mensaje hola mundo) se realizan sin necesidad de recargar la página.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Hola Mundo con AJAX</title>

<script type="text/javascript">
function descargaArchivo() {
          // Obtener la instancia del objeto XMLHttpRequest
          if(window.XMLHttpRequest) {
                    peticion_http = new XMLHttpRequest();
          }
          else if(window.ActiveXObject) {
                    peticion_http = new ActiveXObject("Microsoft.XMLHTTP");
          }
          // Preparar la funcion de respuesta
          peticion_http.onreadystatechange = muestraContenido;
          // Realizar peticion HTTP
          peticion_http.open('GET', 'http://localhost/holamundo.txt', true);
          peticion_http.send(null);
          function muestraContenido() {
                    if(peticion_http.readyState == 4) {
                              if(peticion_http.status == 200) {
                                        alert(peticion_http.responseText);
                              }
                    }
          }
}
window.onload = descargaArchivo;
</script>
</head>
<body></body>
</html>
```

# AJAX

- Descripción del ejemplo:
  - instanciar el objeto XMLHttpRequest

```
peticion_http = new XMLHttpRequest();
```

  - preparar la función de respuesta

```
peticion_http.onreadystatechange = muestraContenido;
```

    - onreadystatechange = cuando reciba la respuesta del servidor, se ejecuta manejador del evento.
  - realizar la petición al servidor

```
peticion_http.open('GET', 'http://localhost/prueba.txt', true);
peticion_http.send(null);
```

  - ejecutar la función de respuesta
    - muestraContenido() comprueba que se ha recibido respuesta válida del servidor (propiedad *readyState*)

# AJAX - XMLHttpRequest

- Propiedades de ***XMLHttpRequest***

| Propiedad | Descripción |
|---|---|
| `readyState` | Valor numérico (entero) que almacena el estado de la petición |
| `responseText` | El contenido de la respuesta del servidor en forma de cadena de texto |
| `responseXML` | El contenido de la respuesta del servidor en formato XML. El objeto devuelto se puede procesar como un objeto DOM |
| `status` | El código de estado HTTP devuelto por el servidor (200 para una respuesta correcta, 404 para "No encontrado", 500 para un error de servidor, etc.) |
| `statusText` | El código de estado HTTP devuelto por el servidor en forma de cadena de texto: "OK", "Not Found", "Internal Server Error", etc. |

# AJAX

- Propiedad ***readyState***

| Valor | Descripción |
|---|---|
| 0 | No inicializado (objeto creado, pero no se ha invocado el método open) |
| 1 | Cargando (objeto creado, pero no se ha invocado el método send) |
| 2 | Cargado (se ha invocado el método send, pero el servidor aún no ha respondido) |
| 3 | Interactivo (se han recibido algunos datos, aunque no se puede emplear la propiedad responseText) |
| 4 | Completo (se han recibido todos los datos de la respuesta del servidor) |

# AJAX

- Métodos de ***XMLHttpRequest***

| Método | Descripción |
|---|---|
| `abort()` | Detiene la petición actual |
| `getAllResponseHeaders()` | Devuelve una cadena de texto con todas las cabeceras de la respuesta del servidor |
| `getResponseHeader("cabecera")` | Devuelve una cadena de texto con el contenido de la cabecera solicitada |
| `onreadystatechange` | Responsable de manejar los eventos que se producen. Se invoca cada vez que se produce un cambio en el estado de la petición HTTP. Normalmente es una referencia a una función JavaScript |
| `open("metodo", "url")` | Establece los parámetros de la petición que se realiza al servidor. Los parámetros necesarios son el método HTTP empleado y la URL destino (puede indicarse de forma absoluta o relativa) |
| `send(contenido)` | Realiza la petición HTTP al servidor |
| `setRequestHeader("cabecera", "valor")` | Permite establecer cabeceras personalizadas en la petición HTTP. Se debe invocar el método `open()` antes que `setRequestHeader()` |

# AJAX – Petición HTTP

- El segundo paso es **componer la URL del recurso** que procesa la petición. En nuestro contexto, una **página PHP**.
- Habitualmente se pasan **parámetros** a la página PHP en la URL.
- Sin embargo, hay que tener en cuenta que una URL no admite cualquier carácter, por lo que hay que **codificar los parámetros** con la función **encodeURIComponent**

```
var url = "validaUsuario.php?id="
          + encodeURIComponent("García");
```

# AJAX – Petición HTTP

- El siguiente paso es registrar la URL en el objeto `XMLHttpRequest`:

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

- Los **parámetros** son:
  - El **tipo de petición HTTP**: habitualmente GET.
  - La **URL del recurso** que procesa la petición.
  - El **modo de petición**, **asíncrono** (true) o *síncrono* (false).
- Si el modo es asíncrono, se registra la **función** de ***callback*** que es llamada cuando cambie el estado de la petición:

```
req.onreadystatechange = callback;
```

# AJAX – Petición HTTP

- Por último, se **realiza la petición**:

  ```
  req.send(null);
  ```

  - El parámetro es el *contenido* de la petición.
  - *contenido*: información que se va a enviar al servidor junto con la petición HTTP (una cadena de texto, un array de bytes o un objeto XML DOM)
  - Habitualmente es nulo debido a que los datos se establecen en la URL.
- **Importante**: si la llamada es asíncrona, al invocar `send` **el código no espera**. En general, esto es fundamental para que el navegador no se quede bloqueado por la latencia de la comunicación de red.

# AJAX – Respuesta

- El objeto `XMLHttpRequest` controla todo el proceso de petición.
- Conforme **cambia el estado**, notifica a la función de ***callback***.
- La **petición** es **completa** cuando se llega al estado 4.
- Una vez completa, el estado de la **respuesta HTTP** debe ser 200 (**OK**) para que el proceso haya finalizado correctamente.
- En general, el patrón de la **función *callback*** es:

```
if (req.readyState == 4 &&
      req.status == 200) {
         … → procesar la repuesta
```

# AJAX – Respuesta

- El objeto `XMLHttpRequest` mantiene el **resultado** de la respuesta de dos modos:
  - Como **texto**, habitualmente formateado en XHTML.

```
req.responseText;
```

  - Como **árbol DOM**:

```
req.responseXML;
```

- El **procesamiento como texto** es el más cómodo para asignar el contenido de la respuesta a un nodo de la página utilizando la propiedad **innerHTML**.

# AJAX – Envío de datos

- Ejemplo de interacción con el servidor

```
<form>
        <label for="fecha_nacimiento">Fecha de nacimiento:</label>
        <input type="text" id="fecha_nacimiento" name="fecha_nacimiento" />
        <br/>
        <label for="codigo_postal">Codigo postal:</label>
        <input type="text" id="codigo_postal" name="codigo_postal" />
        <br/>
        <label for="telefono">Telefono:</label>
        <input type="text" id="telefono" name="telefono" />
        <br/>
        <input type="button" value="Validar datos" />
</form>
<div id="respuesta"></div>
```

```
var peticion_http = null;
function inicializa_xhr() {
          return new XMLHttpRequest();
}
function crea_query_string() {
          var fecha = document.getElementById("fecha_nacimiento");
          var cp = document.getElementById("codigo_postal");
          var telefono = document.getElementById("telefono");
          return "fecha_nacimiento=" + encodeURIComponent(fecha.value) +
          "&codigo_postal=" + encodeURIComponent(cp.value) +
          "&telefono=" + encodeURIComponent(telefono.value) +
          "&nocache=" + Math.random();
}
```

```
function valida() {
peticion_http = inicializa_xhr();
   if(peticion_http) {
          peticion_http.onreadystatechange = procesaRespuesta;
          peticion_http.open("POST", "http://localhost/validaDatos.jsp", true);
          peticion_http.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
          var query_string = crea_query_string();
          peticion_http.send(query_string);
   }
}
function procesaRespuesta() {
          if(peticion_http.readyState == 4) {
            if(peticion_http.status == 200) {
                    document.getElementById("respuesta").innerHTML = peticion_http.responseText;
            }
          }
}
```

# AJAX

- peticion_http.**setRequestHeader**("Content-Type", "application/x-www-form-urlencoded");
  - si no se establece la cabecera Content-Type correcta, el servidor descarta todos los datos enviados mediante el método POST.
  - para enviar parámetros mediante el método **POST**, es obligatorio incluir la cabecera ***Content-Type***

- peticion_http.**send**(query_string);
  - se encarga de enviar los parámetros al servidor (cadena de texto, documento XML,…)
  - Los parámetros se envían en forma de cadena:
    - variables = valores, concatenados mediante ***&***
  - *encodeURIComponent() :* imprescindible para evitar problemas con caracteres especiales.

# AJAX – Procesamiento de datos

- XMLHttpRequest también permite la recepción de respuestas de servidor en formato XML.
  - *petición_http.**responseXML***
  - Procesamiento mediante ***métodos DOM*** de manejo de documentos XML/HTML

```
<respuesta>
        <mensaje>...</mensaje>
        <parametros>
           <telefono>...</telefono>
           <codigo_postal>...</codigo_postal>
           <fecha_nacimiento>...</fecha_nacimiento>
        </parametros>
</respuesta>
```

# AJAX

- Procesamiento de la respuesta

```
function procesaRespuesta() {
   if(peticion_http.readyState == READY_STATE_COMPLETE) {
      if(peticion_http.status == 200) {
          var documento_xml = peticion_http.responseXML;
          var root = documento_xml.getElementsByTagName("respuesta")[0];
          var mensajes = root.getElementsByTagName("mensaje")[0];
          var mensaje = mensajes.firstChild.nodeValue;
          var parametros = root.getElementsByTagName("parametros")[0];
          var telefono = parametros.getElementsByTagName("telefono")[0].firstChild.nodeValue;
          var fecha_nacimiento = parametros.getElementsByTagName("fecha_nacimiento")[0]
                                                                  .firstChild.nodeValue;
          var codigo_postal = parametros.getElementsByTagName("codigo_postal")
[0].firstChild.nodeValue;

          document.getElementById("respuesta").innerHTML = mensaje + "<br/>" + "Fecha
                    nacimiento = " + fecha_nacimiento + "<br/>" + "Codigo postal = " +
                              codigo_postal + "<br/>" + "Telefono = " + telefono;
      }
   }
}
```

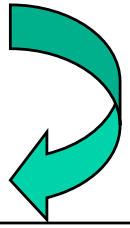
# JSON

- **JSON** (*JavaScript Object Notation*)
  - Formato mucho más compacto y ligero que XML, usado para el intercambio de información
  - Fácil de procesar por el navegador
  - Tipo MIME oficial es *application/json*

- Representa estructuras de datos (arrays) y objetos (arrays asociativos) en forma de texto

# JSON

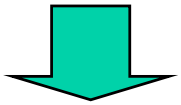
```
var modulos = new Array();
modulos[0] = "Lector RSS";
modulos[1] = "Gestor email";
modulos[2] = "Agenda";
modulos[3] = "Buscador";
modulos[4] = "Enlaces";
```



```
var modulos = ["Lector RSS", "Gestor email", "Agenda", "Buscador", "Enlaces"];
```

```
var modulos = new Array();
modulos.titulos = new Array();
modulos.titulos['rss'] = "Lector RSS";
modulos.titulos['email'] ="Gestor de email";
modulos.titulos['agenda'] = "Agenda";
```

```
var modulos = new Array();
modulos.titulos = new Array();
modulos.titulos.rss = "Lector RSS";
modulos.titulos.email ="Gestor de email";
modulos.titulos.agenda = "Agenda";
```



```
var modulos = new Array();
modulos.titulos = {rss: "Lector RSS", email: "Gestor de email", agenda: "Agenda"};
```

# JSON

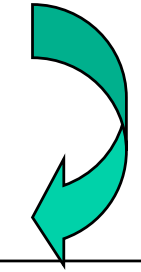
- La notación JSON para los arrays asociativos se compone de tres partes:
  - 1. Los contenidos del array asociativo se encierran entre llaves ({ y })
  - 2. Los elementos del array se separan mediante una coma (,)
  - 3. La clave y el valor de cada elemento se separan mediante dos puntos (:)
- Si la clave no contiene espacios en blanco, es posible prescindir de las comillas

# JSON

```
var modulo = new Object();
modulo.titulo = "Lector RSS";
modulo.objetoInicial = new Object();
modulo.objetoInicial.estado = 1;
modulo.objetoInicial.publico = 0;
modulo.objetoInicial.nombre = "Modulo_RSS";
modulo.objetoInicial.datos = new Object();
```

```
var modulo = {
titulo : "Lector RSS",
objetoInicial : { estado : 1, publico : 0, nombre : "Modulo RSS", datos : {} }
};
```

```
<respuesta>
          <mensaje>...</mensaje>
          <parametros>
             <telefono>...</telefono>
             <codigo_postal>...</codigo_postal>
             <fecha_nacimiento>...</fecha_nacimiento>
          </parametros>
</respuesta>
```



```
{
mensaje: "...",
parametros: {telefono: "...", codigo_postal: "...", fecha_nacimiento: "..." }
}
```

```
function procesaRespuesta() {
   if(http_request.readyState == READY_STATE_COMPLETE) {
      if(http_request.status == 200) {
         var respuesta_json = http_request.responseText;
         var objeto_json = eval("("+respuesta_json+")");
         var mensaje = objeto_json.mensaje;
         var telefono = objeto_json.parametros.telefono;
         var fecha_nacimiento = objeto_json.parametros.fecha_nacimiento;
         var codigo_postal = objeto_json.parametros.codigo_postal;
         document.getElementById("respuesta").innerHTML = mensaje +
                  "<br>" + "Fecha nacimiento = " + fecha_nacimiento +
                  "<br>" + "Codigo postal = " + codigo_postal +
                  "<br>" + "Telefono = " + telefono;
      }
   }}
```

# AJAX y JSON

- Se debe transformar la cadena en un objeto JSON.
  - Función **eval():** se añaden paréntesis al principio y al final para realizar la evaluación de forma correcta

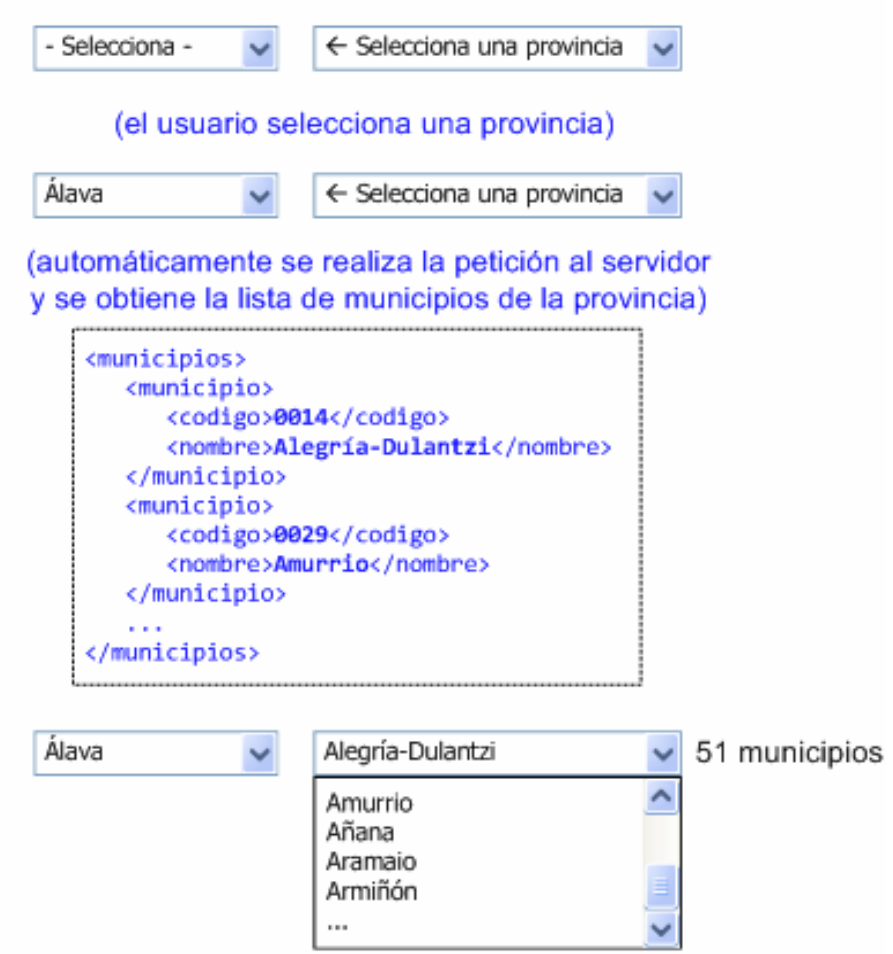
    *var objeto_json = **eval**("("+respuesta_json+")");*
  - Permite acceder a métodos y propiedades mediante la **notación de puntos** tradicional

    *var fecha_nacimiento = objeto_json**.parametros.fecha_nacimiento**;*
  - Es posible el envío de los parámetros en formato JSON
    - Uso de librerías JSON

# AJAX

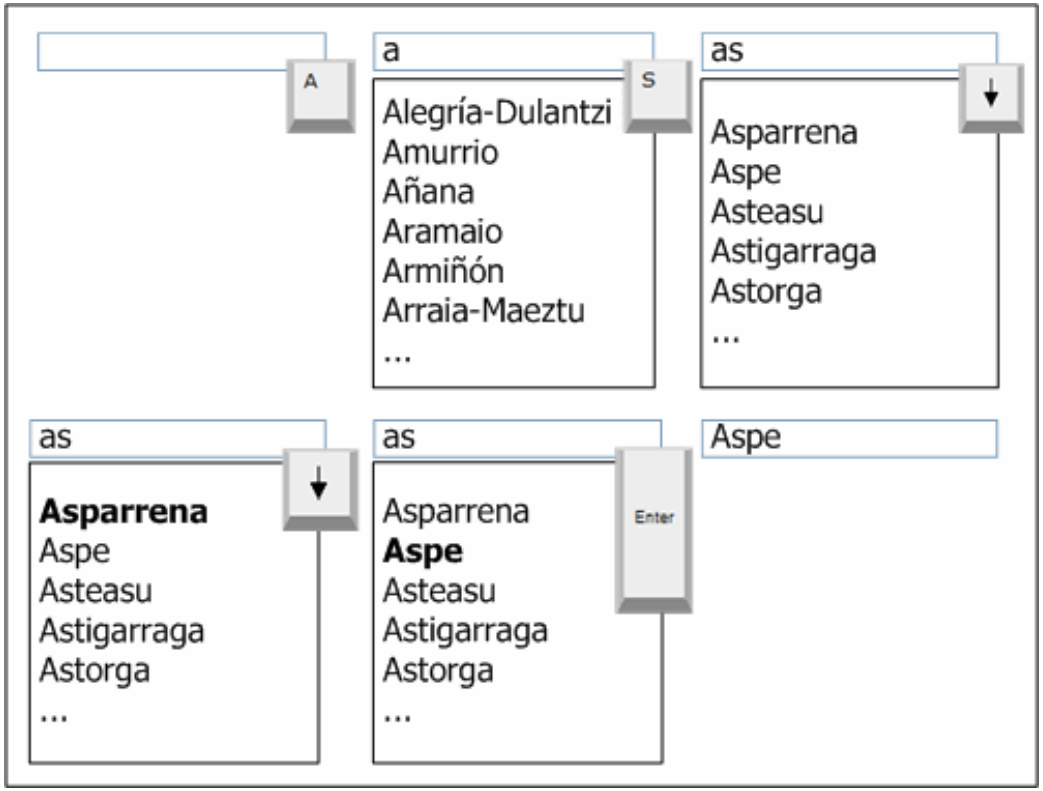
## Listas desplegables encadenadas

- Los valores de la primera lista se incluyen en la página y cuando se selecciona un valor, se realiza una consulta al servidor para obtener los valores de la otra lista.



# AJAX

- Autocompletar
  - Combinar un cuadro de texto y una lista desplegable en AJAX. Cuando el usuario escribe en el cuadro de texto, la aplicación solicita al servidor los términos relacionados con lo escrito. Cuando la aplicación recibe la respuesta, la muestra a modo de ayuda para autocompletar.



# Referencias

- Documentos de referencia:
  - Introducción a AJAX, Javier Eguíluz Pérez (www.librosweb.es) *<fuente transparencias>*
  - AJAX, Manual imprescindible, 2008, Javier Mellado Domínguez, ANAYA
  - Profesional AJAX, Nicholas C. Zakas, Jeremy McPeak y Joe Fawcett, 2006, ANAYA