

Programación Web Avanzada: AJAX y Google Maps

**Universidad de Colima
México**



AJAX

Sergio Luján Mora



**Departamento de Lenguajes y
Sistemas Informáticos**



Índice (1)

- Dos ejemplos tradicionales
- Introducción
- Definición y conceptos
- Cómo funciona
- Ventajas y desventajas
- El objeto XMLHttpRequest
- Ciclo de vida de una transferencia con XMLHttpRequest



Índice (2)

- Ejemplos (1 y 2)
- Envío de datos GET/POST
- Transformación a AJAX
- Formato de datos XML
- Formato de datos JSON
- Bibliografía

Dos ejemplos tradicionales



- **EJERCICIO**
- Página web que muestra titulares de noticias que se actualizan cada 10 segundos

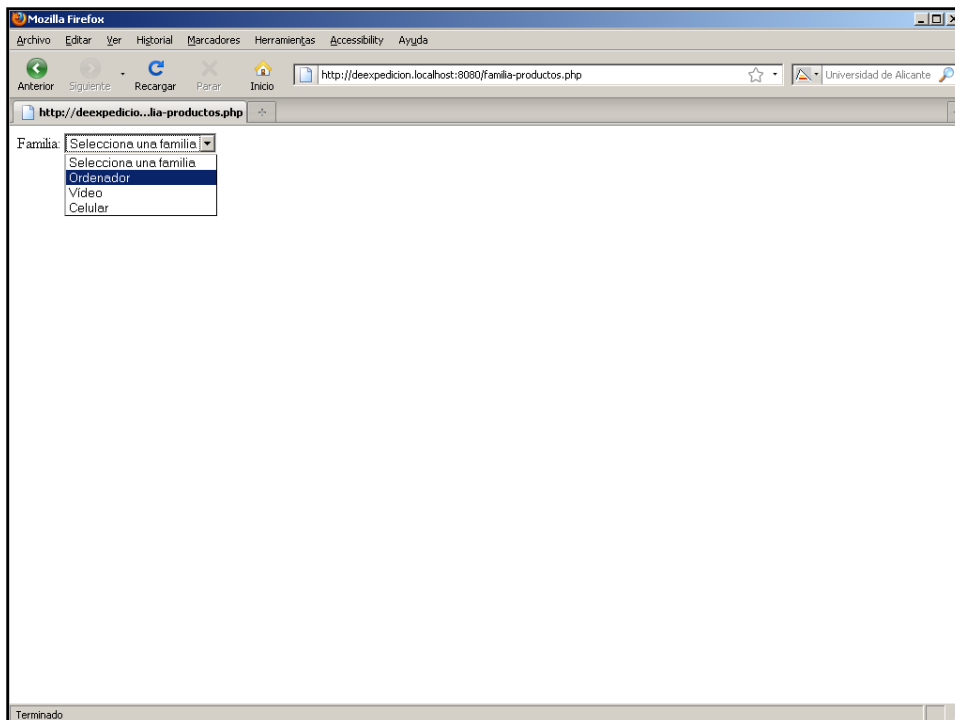
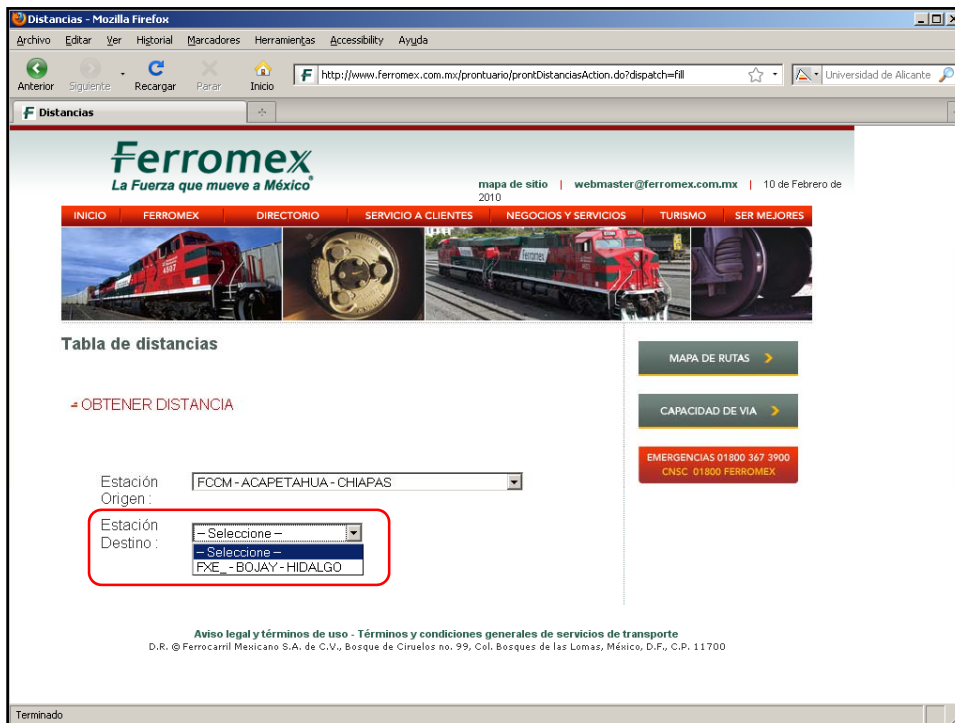
```
<html>
<head>
<title>Titulares del día</title>
<meta http-equiv="refresh" content="10" />
</head>
<body>
<h1>Titulares del día</h1>
<?php
    // Genera de forma aleatoria entre 1 y 10 noticias
    $n = rand(1, 10);
    for($i = 1; $i <= $n; $i++) {
        echo "<h2>Noticia $i</h2>\n";
        // Repite de forma aleatoria la cadena "Bla, bla, bla." entre 1
        y 5 veces
        echo "<p>" . str_repeat("Bla, bla, bla.", rand(1, 5)) .
            "</p>\n";
    }
?>
</body>
</html>
```

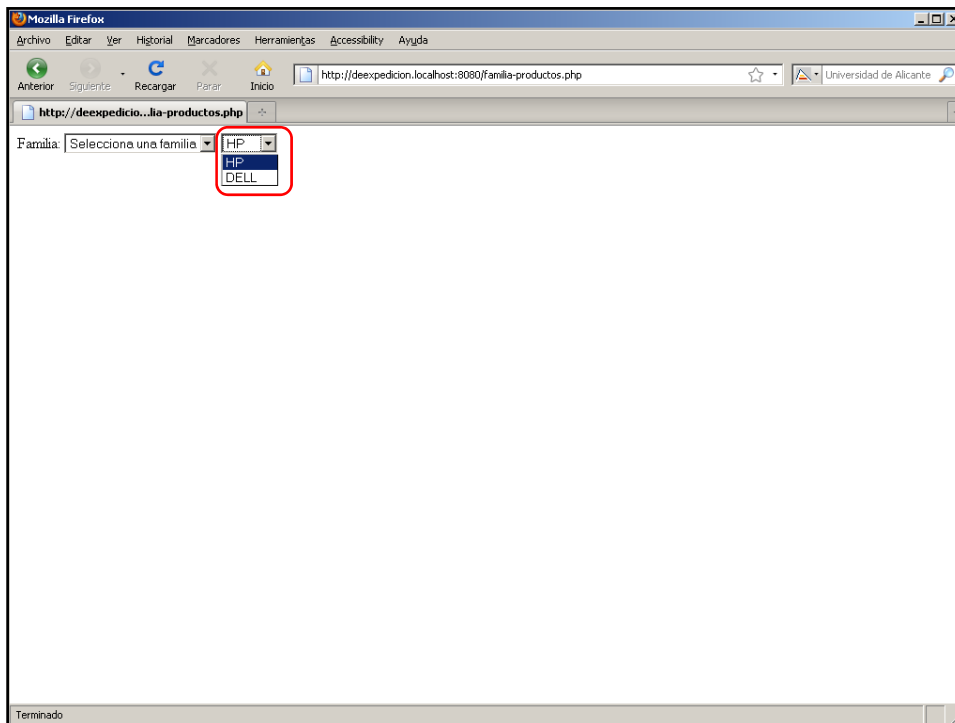
Dos ejemplos tradicionales




- EJERCICIO
- Página web que muestra dos listas desplegables (<select>) en cascada:
 - Inicialmente, la segunda lista está vacía o no aparece
 - Al seleccionar un valor en la primera lista desplegable, se recarga la página con datos en la segunda lista

The screenshot shows the Ferromex website in a Mozilla Firefox browser. The page title is "Distancias - Mozilla Firefox". The address bar shows the URL: <http://www.ferromex.com.mx/prontuario/prontDistanciasAction.do?dispatch=fill>. The page content includes the Ferromex logo and navigation links: INICIO, FERROME, SOCIOS Y SERVICIOS, TURISMO, and SER MEJORES. A dropdown menu is open, displaying a list of station pairs such as "FSRR - PENUELA - VERACRUZ", "FSRR - PIEDRAS NEGRAS - VERACRUZ", "FSRR - POTRERO - VERACRUZ", "FSRR - PRESIDENTE JUAREZ - OAXACA", "FSRR - PUEBLA - PUEBLA", "FSRR - RINCONADA - PUEBLA", "FSRR - RIO BLANCO - VERACRUZ", "FSRR - RODRIGUEZ CLARA - VERACRUZ", "FSRR - ROSENDO MARQUEZ - PUEBLA", "FSRR - RUBIN - VERACRUZ", "FSRR - SALINA CRUZ - OAXACA", "FSRR - SAN AGUSTIN - HIDALGO", "FSRR - SAN ANDRES - PUEBLA", "FSRR - SAN CRISTOBAL - VERACRUZ", "FSRR - SAN LORENZO - HIDALGO", "FSRR - SAN MARCOS - PUEBLA", "FSRR - SAN MARTIN - PUEBLA", "FSRR - SAN MIGUELITO - VERACRUZ", "FSRR - SANCHEZ - PUEBLA", and "FSRR - SANTA ANA - TLAXCALA". Below the dropdown menu, there are two "Estación" (Station) dropdown menus: "Estación Origen" and "Estación Destino", both currently showing "-- Seleccione --". The "Estación Destino" dropdown is highlighted with a red box. At the bottom of the page, there is a footer with the text: "Aviso legal y términos de uso - Términos y condiciones generales de servicios de transporte" and "D.R. © Ferrocarril Mexicano S.A. de C.V., Bosque de Ciruelos no. 99, Col. Bosques de las Lomas, México, D.F., C.P. 11700".





Programación Web Avanzada: AJAX y Google Maps

 **Universitat d'Alacant**
Universidad de Alicante

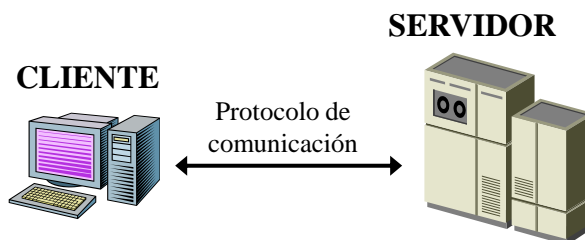
```
<html>
<head><title>Listas desplegables en cascada</title></head>
<body>
<form action="" method="post">
<p>
Familia:
<select name="familia" onchange="submit()">
<option value="">Selecciona una familia</option>
<option value="1">Ordenador</option>
<option value="2">VÍdeo</option>
<option value="3">Celular</option>
</select>
<?php
    // Código que genera la segunda lista con los productos
?>
</p>
</form>
</body>
</html>
```

Dos ejemplos tradicionales

```
<?php
// Productos para cada familia
$productos[1] =
  "<option>HP</option><option>DELL</option>";
$productos[2] =
  "<option>Sony</option><option>Panasonic</option>";
$productos[3] =
  "<option>Motorola</option><option>Samsung</option>";
// Detecta si ha habido envío de datos
if(isset($_POST["familia"]))
{
  echo '<select name="producto">';
  echo $productos[$_POST["familia"]];
  echo '</select>';
}
?>
```

Introducción

- Arquitectura cliente-servidor
 - El cliente solicita un servicio y el servidor lo provee.
 - La parte de cliente solo tiene la lógica de presentación y en algunos casos un mínimo de lógica de negocio.





Introducción

- Aplicaciones en entorno web:
 - Basada en la arquitectura cliente-servidor
 - Cada interacción del usuario genera una transacción contra el servidor.
 - En algunos casos, según las características de la aplicación, el interfaz (una página HTML) puede generar demasiadas peticiones.
 - Problemas:
 - Esperas del cliente
 - Errores en la transmisión



Introducción

- *Asynchronous Javascript And XML*
 - XML y JavaScript asíncronos
- No es una nueva tecnología,
 - sino un conjunto de tecnologías, el uso de las cuales permite una nueva filosofía de desarrollo de aplicaciones web.
- Combinación de HTML/XHTML, CSS, JavaScript, DOM, XML, XSLT, HTTP.
 - Todas ellas son estándares abiertos, tecnologías implantadas, difundidas y conocidas en Internet.



Introducción

- El primer uso “público” de este término fue hecho por Jesse James Garrett en su artículo de febrero 2005 *Ajax: A New Approach to Web Applications*.
 - <http://adaptivepath.com/publications/essays/archives/000385.php>



Introducción

- ¿Qué aporta Ajax?
 - Interacción diferente con el servidor:
 - Anticipo de transferencias con el servidor para disponer de los datos más rápidamente (interacción en segundo plano, *background*).
 - Permitir al usuario poder hacer otras cosas en el cliente mientras una transferencia está en curso (interacción asíncrona).
 - Nueva filosofía:
 - La aplicación web se apoya más en el cliente, al cual se le dota de más lógica de negocio.
 - Se consigue una interacción en el cliente similar a las aplicaciones tradicionales de escritorio.



Introducción

- ¿Y qué aporta XML?
 - Es el formato en el que se transmiten los datos el servidor y el cliente.
 - Pero no es el único formato, existe completa libertad.
- Usado en todos los sitios web modernos:
 - Google Docs
 - Google Maps
 - Gmail
 - ...



Definición y conceptos

- La filosofía que se presenta con Ajax es:
 - Cargar y mostrar una página,
 - para luego mantenerse en esa misma página
 - mientras código de script realiza peticiones al servidor en segundo plano (*background*),
 - para obtener los datos que son usados para actualizar la página
 - sólo mostrando u ocultando porciones de la misma.



Definición y conceptos

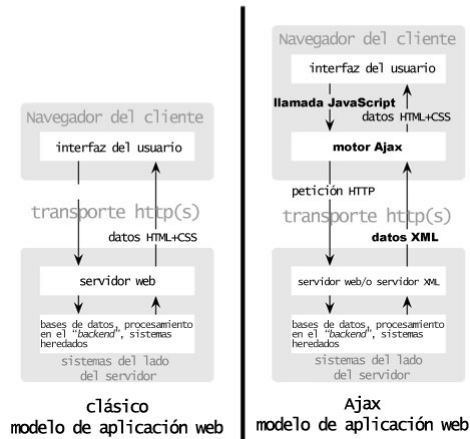
- Tecnologías usadas con Ajax:
 - **HTML/XHTML:** Presentación del contenido.
 - **DOM:** Interacción en el cliente.
 - **JavaScript:** Lenguaje empleado para programar el motor Ajax.
 - Recoger la interacción del usuario (eventos).
 - Para realizar la comunicación con el servidor.
 - Para cambiar el contenido de la pantalla del cliente.
- Se aporta un nuevo enfoque al desarrollo de aplicaciones web.
 - Funcionamiento similar a las aplicaciones de escritorio.
 - Cambio del concepto de programación clásico.
 - Reducción del tráfico que generan las aplicaciones web en cuanto al número y tamaño de las transacciones.



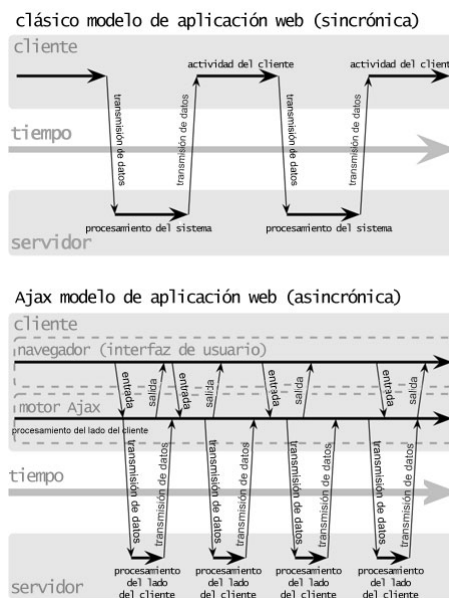
Cómo funciona

- Se introduce una nueva capa entre el cliente y el servidor, llamada "Motor Ajax"
 - Se usa un objeto del DOM llamado **XMLHttpRequest**
 - Por medio de este objeto se realiza la comunicación con el servidor
 - Permite obtener información del servidor y actualizar la interfaz de usuario
- La respuesta que se obtiene del servidor puede estar en varios formatos:
 - Texto plano
 - HTML
 - XML
 - JSON
 - ...

Cómo funciona



- La mayor parte del tráfico se da al iniciar la comunicación.
- Durante la interacción del usuario se envía la mínima cantidad de información posible para agilizar la aplicación.
- La página no espera la respuesta del servidor.
- Se generan transacciones asíncronas mientras la aplicación continua funcionando.





Ventajas

- Transacciones asíncronas, el usuario no tiene que esperar después de una petición.
- Menor cantidad de información.
- Tecnologías basadas en estándares abiertos.
- Significa una aproximación a las aplicaciones tradicionales en cuanto a interactividad y velocidad de respuesta.



Desventajas

- Puede aumentar el número de peticiones contra el servidor.
- Existen incompatibilidades entre navegadores.
 - El objeto XMLHttpRequest no se invoca de la misma forma en los navegadores.
- Pérdida de control de navegación.
 - Los botones atrás y adelante del navegador dejan de funcionar.
 - No se puede crear un favorito (marcador) a una página, porque siempre es la misma página.



XMLHttpRequest

- Permite la comunicación asíncrona con el servidor.
 - Es un API que se puede utilizar desde JavaScript, Jscript, VBScript y otros lenguajes de script
 - Permite crear canales de comunicación independientes del navegador.
 - La comunicación se realiza mediante HTTP.
- En proceso de estandarización por el W3C:
 - The XMLHttpRequest Object W3C Working Draft 19 November 2009:
 - <http://www.w3.org/TR/XMLHttpRequest/>



XMLHttpRequest

- El objeto `XMLHttpRequest` fue originalmente desarrollado por Microsoft.
- La implementación de Microsoft se llama XMLHTTP y es un objeto ActiveX.
 - Difiere ligeramente de los estándares publicados.
 - Ha estado disponible desde Internet Explorer 5.0 via JScript y VBScript.



XMLHttpRequest

- El proyecto Mozilla incluyó la primera implementación nativa compatible de XMLHttpRequest en Mozilla 1.0 en 2002.
- Esta implementación fue seguida posteriormente por Apple desde Safari 1.2, Konqueror, Opera Software desde Opera 8.0 e iCab desde 3.0b352.



XMLHttpRequest

- Métodos:
 - **abort()**
 - **getAllResponseHeaders()**
 - **getResponseHeader(etiqueta)**
 - **open(metodo, url, asincrona, usuario, password):**
 - Métodos: GET, POST, HEAD, POST o DELETE.
 - **send(contenido)**
 - **setRequestHeader(etiqueta, valor)**



XMLHttpRequest

- Propiedades:
 - **onreadystatechange**: Asignación de manejador al evento de cambio de estado.
 - **readyState**: Estado de la conexión (sólo lectura).
 - 0: no inicializada
 - 1: cargando
 - 2: cargada
 - 3: interactiva
 - 4: completada
 - **responseText**: La respuesta en forma de texto plano.
 - **responseXML**: La respuesta en formato XML.
 - **status**: Código de respuesta del servidor.
 - Códigos HTTP. 200: Transacción completada con éxito.
 - **statusText**: Mensaje asociado a la respuesta.



Ciclo de vida de una transferencia con XMLHttpRequest

1. Se **crea el objeto** XMLHttpRequest
 2. Se le **asigna un manejador** del evento de cambio de estado
 3. Se **envía la petición** en función de la interacción del usuario
 4. Al **recibir un cambio de estado**, se gestiona la respuesta.
 5. Si la respuesta es correcta, se **obtienen los datos** y se muestran al cliente.
- Se debe vincular a algún objeto de la página, la invocación de la petición al servidor, ya sea mediante un click, un link o cualquier otro evento.

Creación del objeto XMLHttpRequest

- Según el navegador se crea de dos formas:

- Internet Explorer:

```
peticion = new ActiveXObject("Microsoft.XMLHTTP");
```

- Otros navegadores:

```
peticion = new XMLHttpRequest();
```

- Código fuente independiente del navegador

```
if (window.XMLHttpRequest)
    peticion = new XMLHttpRequest();
else if(window.ActiveXObject)
    peticion = new ActiveXObject("Microsoft.XMLHTTP");
else {
    alert("Su navegador no soporta AJAX");
    return;
}
```

Creación del objeto XMLHttpRequest

- Otra forma, mediante excepciones:

```
try {
    peticion = new ActiveXObject("Msxml2.XMLHTTP");
} catch (e) {
    try {
        peticion = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (ee) {
        try {
            peticion = new XMLHttpRequest();
        } catch (eee) {
            alert("Su navegador no soporta AJAX");
            return;
        }
    }
}
```

Asignar un manejador

- Se debe asignar una función que se invoque cuando cambie el estado del objeto XMLHttpRequest, es decir, cuando el servidor responda a nuestra petición:

```
peticion.onreadystatechange = nombreDeLaFuncion;
```

Esta función la invoca automáticamente el navegador.

Importante: Sin paréntesis (), solo el nombre.

Enviar la petición

- Se debe especificar el tipo de petición, la URL que deseamos invocar y si la transacción es asíncrona o no.
 - Una conexión síncrona (`false`), bloquea el navegador hasta que se obtiene la respuesta
 - Una conexión asíncrona (`true`, el valor por defecto), se ejecuta en segundo plano
 - La URL tiene que ser del mismo dominio que la página actual

```
peticion.open('GET', 'http://www.ua.es/ajax.php', true);
```

- Después se envían los datos con la función send:

```
peticion.send(parámetros de la petición o null);
```

Gestión de la respuesta

- El manejador se invoca al recibir un cambio de estado de la transacción.
 - 0: no inicializada
 - 1: cargando
 - 2: cargada
 - 3: interactiva
 - 4: completada

- Fragmento de código del manejador:

```
if (peticion.readyState == 4) { //fin
  if (peticion.status==200) { // todo OK
    //procesaríamos la petición
  }
} else {
  // Todavía no está...
}
```

Tratamiento de los datos

- La respuesta puede estar en varios formatos: texto plano, HTML, XML, etc.
- La propiedad `responseText` nos da acceso al texto plano de la misma:
`peticion.responseText`
- Si es XML, con la propiedad `responseXML` obtenemos un objeto de tipo `XMLDocument`



Ejemplo

```
<script type="text/javascript">
function ajaxFunction() {
  var xmlhttp;
  if (window.XMLHttpRequest)
    xmlhttp = new XMLHttpRequest();
  else
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

  xmlhttp.onreadystatechange=function() {
    if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      document.myForm.time.value += xmlhttp.responseText + "\n";
    }
  }
  xmlhttp.open("GET","time.php",true);
  xmlhttp.send(null);
}
</script>
```



Ejemplo

```
<script type="text/javascript">
function ajaxFunction() {
  var xmlhttp;
  if (window.XMLHttpRequest)
    xmlhttp = new XMLHttpRequest();
  else
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

  xmlhttp.onreadystatechange=function() {
    if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      document.myForm.time.value += xmlhttp.responseText + "\n";
    }
  }
  xmlhttp.open("GET","time.php",true);
  xmlhttp.send(null);
}
</script>
```

Crear el objeto XMLHttpRequest

Ejemplo

```
<script type="text/javascript">
function ajaxFunction() {
  var xmlhttp;
  if (window.XMLHttpRequest)
    xmlhttp = new XMLHttpRequest();
  else
    xmlhttp = new A
    Asignar una función para el cambio de estado
  xmlhttp.onreadystatechange=function() {
    if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      document.myForm.time.value += xmlhttp.responseText + "\n";
    }
  }
  xmlhttp.open("GET","time.php",true);
  xmlhttp.send(null);
}
</script>
```

Ejemplo

```
<script type="text/javascript">
function ajaxFunction() {
  var xmlhttp;
  if (window.XMLHttpRequest)
    xmlhttp = new XMLHttpRequest();
  else
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  xmlhttp.onreadystatechange=function() {
    if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      document.myForm.time.value += xmlhttp.responseText + "\n";
    }
  }
  Enviar la petición al servidor
  xmlhttp.open("GET","time.php",true);
  xmlhttp.send(null);
}
</script>
```



Ejemplo

```
<script type="text/javascript">
function ajaxFunction() {
  var xmlhttp;
  if (window.XMLHttpRequest)
    xmlhttp = new XMLHttpRequest();
  else
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  xmlhttp.onreadystatechange=function() {
    if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      document.myForm.time.value += xmlhttp.responseText + "\n";
    }
  }
  xmlhttp.open("GET","time.php",true);
  xmlhttp.send(null);
}
</script>
```

Cuando se produzca un cambio, manejar la respuesta



Ejemplo

```
<script type="text/javascript">
function ajaxFunction() {
  var xmlhttp;
  if (window.XMLHttpRequest)
    xmlhttp = new XMLHttpRequest();
  else
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  xmlhttp.onreadystatechange=function() {
    if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      document.myForm.time.value += xmlhttp.responseText + "\n";
    }
  }
  xmlhttp.open("GET","time.php",true);
  xmlhttp.send(null);
}
</script>
```

Cuando sea una respuesta correcta, procesar el resultado y mostrar



Ejemplo (1)

- Una página web con un botón
- Al pulsar el botón, mediante AJAX se llama a un fichero PHP
- El fichero PHP devuelve la hora en el servidor
- En el cliente se muestra el resultado devuelto con una ventana de aviso



Ejemplo (1): código HTML

```
<html>
<head>
<title>"Hola mundo" en AJAX</title>
<script src="holamundo.js"
  type="text/javascript"></script>
</head>
<body>
<p>
<input type="button" onclick="holaMundo()"
  value="Hola mundo" />
</p>
</body>
</html>
```



Ejemplo (1): funciones JavaScript

```
function ajaxobj() {  
  try {  
    _ajaxobj = new ActiveXObject("Msxml2.XMLHTTP");  
  } catch (e) {  
    try {  
      _ajaxobj = new ActiveXObject("Microsoft.XMLHTTP");  
    } catch (ee) {  
      _ajaxobj = false;  
    }  
  }  
  if (!_ajaxobj && typeof XMLHttpRequest!='undefined')  
    _ajaxobj = new XMLHttpRequest();  
  
  return _ajaxobj;  
}
```



Ejemplo (1): funciones JavaScript

```
function holaMundo() {  
  ajax = ajaxobj();  
  ajax.onreadystatechange=function() {  
    if (ajax.readyState==4 && ajax.status == 200) {  
      alert(ajax.responseText);  
    }  
  }  
  
  ajax.open("GET", "holamundo.php", true);  
  ajax.send(null);  
}
```

Ejemplo (1): fichero PHP

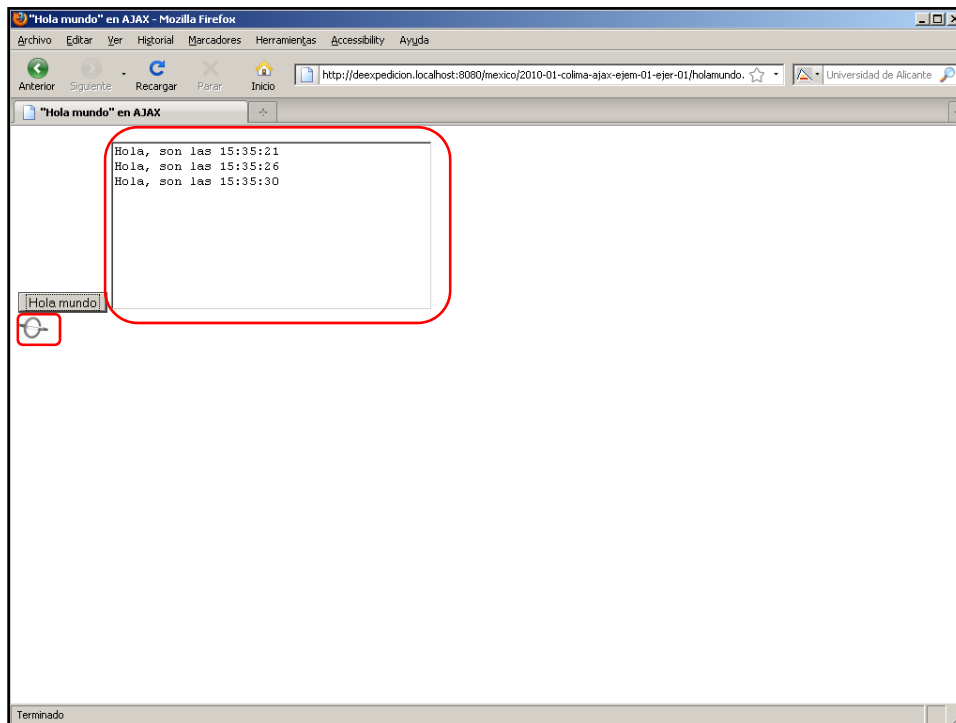
```
<?php
// Fichero holamundo.php
echo "Hola, son las " . date("H:i:s");
?>
```

Ejemplo (1)



• EJERCICIO

- Mostrar un mensaje (o una imagen en movimiento) mientras se espera la respuesta del servidor
- Mostrar un área de texto donde se vayan almacenando las respuestas recibidas



Programación Web Avanzada: AJAX y Google Maps

Universitat d'Alacant
Universidad de Alicante

Ejemplo (1)

- Añadir a la página un área de texto:

```
<textarea id="mensajes" cols="40"  
rows="10"></textarea>
```
- Añadir a la página web una imagen (GIF animado) inicialmente oculto:

```

```

Ejemplo (1)

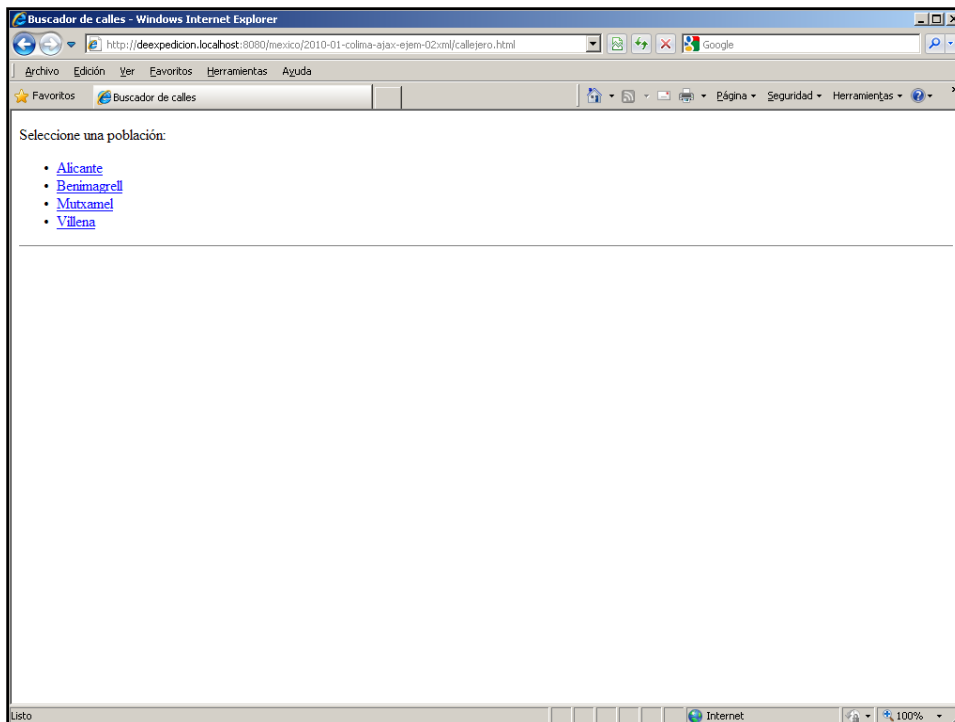
```
function holaMundo() {  
    ajax = ajaxobj();  
    ajax.onreadystatechange=function() {  
        if (ajax.readyState==4 && ajax.status == 200) {  
            document.getElementById("mensajes").value +=  
            ajax.responseText + "\n";  
            document.getElementById("imgLoading").style.display =  
            "none";  
        }  
        else  
            document.getElementById("imgLoading").style.display =  
            "block";  
    }  
    ajax.open("GET", "holamundo.php", true);  
    ajax.send(null);  
}
```

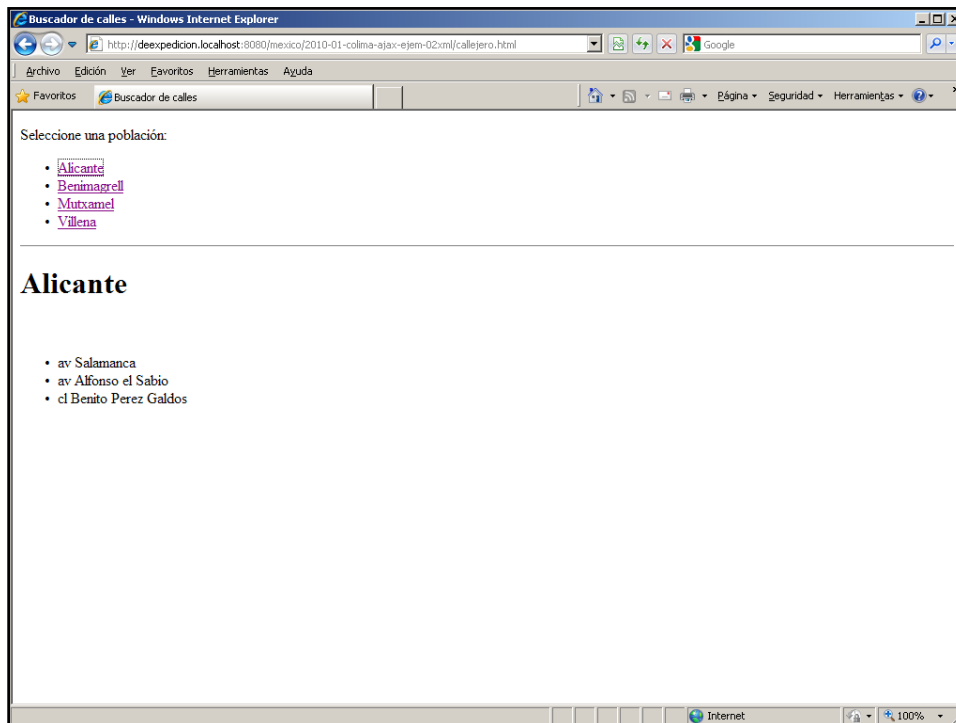
Ejemplo (1)

- Preloaders: Free AJAX animated loading gif's
– <http://preloaders.net/>

Ejemplo (2)

- Diseño de una página web con un listado de poblaciones.
- En función de la población que se seleccione, se mostrará un listado de calles de la misma.
- Las calles se solicitarán al servidor por medio de una petición asíncrona (AJAX), accediendo a un fichero XML con las calles de la población (un fichero por población).





Programación Web Avanzada: AJAX y Google Maps

Universitat d'Alacant
Universidad de Alicante

Ejemplo (2): fichero de datos

```
<?xml version="1.0"?>
<callejero>
  <poblacion>Benimagrell</poblacion>
  <calle tipo="avenida">del pais valencia</calle>
  <calle tipo="calle">alacant</calle>
  <calle tipo="calle">sant vicent</calle>
  <calle tipo="calle">del tordo</calle>
  <calle tipo="calle">del pulpo</calle>
  <calle tipo="plaza">del ayuntamiento</calle>
  <calle tipo="plaza">nova del ravalet</calle>
</callejero>
```

Ejemplo (2): funciones en JavaScript

```
function fInvoca(pURL) {
    if (window.XMLHttpRequest) {
        oPetición = new XMLHttpRequest();
    }
    else if (window.ActiveXObject) {
        oPetición = new
        ActiveXObject("Microsoft.XMLHTTP");
    }
    if (oPetición) {
        oPetición.onreadystatechange =
        fGestionPetición;
        oPetición.open('GET', pURL, true);
        oPetición.send(null);
    }
}
```

Ejemplo (2): funciones en JavaScript

```
function fGestionPetición() {
    if (oPetición.readyState==4) {
        if (oPetición.status==200) {
            //alert(oPetición.responseText);
            fProcesaDatos(oPetición.responseXML);
        } else {
            alert("Error en la petición de datos
            al servidor!\nError: "+oPetición.status);
        }
    }
}
```

Ejemplo (2): funciones en JavaScript

```
function fProcesaDatos(pDoc) {
    oCapa = document.getElementById("calles");
    //Nombre de la población:
    oPoblacion =
    pDoc.getElementsByTagName("poblacion").item(0);
    sHTML = "<h1>" + oPoblacion.text + "</h1>";
    // Lista de calles:
    // [continúa el código]
```

Ejemplo (2): funciones en JavaScript

```
// Lista de calles:
sHTML = sHTML + "<br><ul>";
oCalles = pDoc.getElementsByTagName("calle");
for (var i = 0; i < oCalles.length; i++) {
    // Ponemos el tipo de vía y el nombre:
    sHTML = sHTML + "<li>" +
    oCalles.item(i).getAttribute("tipo") + " " +
    oCalles.item(i).text + "</li>";
}
sHTML = sHTML + "</ul>";
oCapa.innerHTML = sHTML;
}
```



Ejemplo (2): código HTML

```
<body>
<div id="poblaciones">
<ul>
<li><a
  href="javascript:fInvoca('benimagrell.xml');">Ben
  imagrell</a></li>
<li><a
  href="javascript:fInvoca('mutxamel.xml');">Mutxam
  el</a></li>
<li><a
  href="javascript:fInvoca('villena.xml');">Villena
  </a></li>
</div>
<hr>
<div id="calles">
</div>
</body>
```



Envío de datos GET/POST

- GET:
 - En la URL en el método `open()`
 - `encodeURIComponent()` codifica los datos para que se puedan enviar en la URL mediante `get`

```
var idFamilia =
  document.getElementById("familia").value;
xmlHttp.open("GET", "productos.php?familia=" +
  encodeURIComponent(idFamilia), true);
```

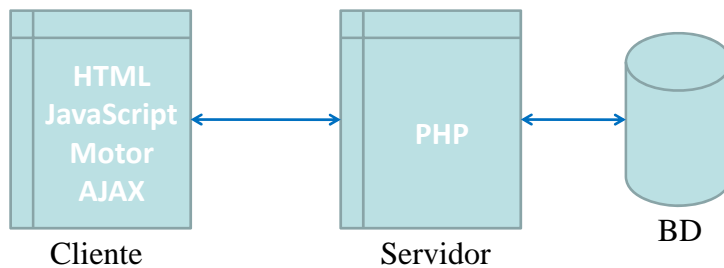
Envío de datos GET/POST

- POST:

```
var idFamilia = document.getElementById("familia").value;  
var parameters = "familia=" +  
    encodeURIComponent(idFamilia);  
xmlHttp.open("POST","productos.php", true);  
  
xmlHttp.setRequestHeader("Content-Type", "application/x-  
    www-form-urlencoded");  
xmlHttp.setRequestHeader("Content-length",  
    parameters.length);  
xmlHttp.setRequestHeader("Connection", "close");  
  
xmlHttp.send(parameters);
```

Transformación a AJAX

- Separar la lógica del cliente y del servidor



Transformación a AJAX



- **EJERCICIO**
- Página web que muestra titulares de noticias que se actualizan cada 10 segundos

Transformación a AJAX



- **EJERCICIO**
- Página web que muestra dos listas desplegables (<select>) en cascada:
 - Inicialmente, la segunda lista está vacía o no aparece
 - Al seleccionar un valor en la primera lista desplegable, se recarga la página con datos en la segunda lista

Formato de datos XML

- Con la propiedad `responseXML` obtenemos un objeto de tipo `XMLDocument`
- Emplear funciones y propiedades del DOM para recorrer el documento

Formato de datos XML

- Cada nodo posee un conjunto de propiedades que lo relacionan con sus “familiares”:
 - `childNodes`
 - `firstChild`
 - `lastChild`
 - `parentNode`
 - `nextSibling`
 - `prevSibling`



Formato de datos XML

- `getElementById("elementID")`
 - Útil para recuperar un elemento concreto
`var element = document.getElementById("myTable")`
- `getElementsByTagName("tagName")`:
 - Útil para recuperar un conjunto de elementos de un mismo tipo
`var images = document.getElementsByTagName("img")`
 - El valor especial "*" representa todas las etiquetas
- `getAttribute("attrName")`:
 - `var element = document.getElementById("myTable").getAttribute("width");`



Formato de datos XML

```
xmlDoc = xmlhttp.responseXML;  
  
x = xmlDoc.documentElement.childNodes;  
  
for (i=0 ; i < x.length; i++)  
{  
    document.write("Nodename: " + x[i].nodeName);  
    document.write(" (nodetype: " + x[i].nodeType +  
    ")<br />");  
}
```

Formato de datos XML

- En el servidor, al devolver un documento XML se tiene que indicar el tipo MIME adecuado:

```
<?php
  header("Content-Type: text/xml");
  // O también
  header("Content-Type: application/xml");
?>
```

Formato de datos JSON

- JSON (*JavaScript Object Notation*) es un formato de intercambio de datos ligero basado en texto
- JSON Specification RFC 4627:
 - <http://tools.ietf.org/html/rfc4627>
- JSON Validator:
 - <http://www.jsonlint.com/>

Formato de datos JSON

- Se emplea para la serialización de datos estructurados:
 - Arrays:
 - La lista de valores encerrados entre corchetes y separados por comas
 - Objetos:
 - Los pares propiedad/valor encerrados entre llaves y separados por comas
 - La propiedad se separa del valor por dos puntos
 - En ambos casos, las propiedades y valores de tipo **cadena encerrados entre comillas dobles**

Formato de datos JSON

- Ejemplo de arrays:

```
[1, 2, 3, 4, 5]
```

```
["Alicante", "Castellón", "Valencia"]
```

Formato de datos JSON

- Ejemplo de objetos:

```
{"nombre": "Sergio", "apellidos": "Luján  
Mora"}
```

```
{"posicion": {"x": 10, "y": 20}}
```

Formato de datos JSON

- Ejemplo:

```
{"menu": {  
  "id": "file",  
  "value": "File",  
  "popup": {  
    "menuitem": [  
      {"value": "New", "onclick": "CreateDoc()"},  
      {"value": "Open", "onclick": "OpenDoc()"},  
      {"value": "Close", "onclick": "CloseDoc()"}  
    ]  
  }  
}}
```

Formato de datos JSON

- Ejemplo:

```
{  
  "Image": {  
    "Width": 800,  
    "Height": 600,  
    "Title": "View from 15th Floor",  
    "Thumbnail": {  
      "Url": "http://www.ex.com/image/481989943",  
      "Height": 125,  
      "Width": "100"  
    }  
  }  
}
```

Formato de datos JSON

- Los datos enviados en formato JSON se reciben a través de la propiedad `responseText`
- Para reconstruir el array u objeto, se emplea `eval()`:

```
var json_datos = xmlhttp.responseText;  
miObjeto = eval('(' + json_datos + ')');
```



Bibliografía

- AJAX (Wikipedia):
 - <http://es.wikipedia.org/wiki/AJAX>
- Ajax: A New Approach to Web Applications:
 - <http://adaptivepath.com/publications/essays/archives/000385.php>
- AJAX un nuevo acercamiento a aplicaciones web (traducción del anterior):
 - <http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php>
- “Hola mundo” en Ajax:
 - <http://www.maestrosdelweb.com/editorial/ajaxpaso/>
- Introducción a Ajax:
 - <http://www.librosweb.es/ajax/index.html>