Advanced Web Programming with JavaScript and Google Maps

Voronezh State University
Voronezh (Russia)

AJAX

Sergio Luján Mora
Table of contents

• Two classical examples
• Introduction
• How it works
• XMLHttpRequest object
• AJAX step by step
• Sending data with GET/POST
• XML data format
• JSON data format
• More information

Two classical examples

• EXERCISE
• A web page shows news headlines updated every 10 seconds
Two classical examples

• **EXERCISE**

• A web page shows two lists (<select>):
  
  – Initially, the second list is empty or it doesn’t appear
  
  – When selecting a value in the first list, the web page is reloaded and the second list shows new data
Two classical examples

```php
<?php
// Products for each family
$products[1] = "<option>HP</option><option>DELL</option>
$products[2] = "<option>Sony</option><option>Panasonic</option>
$products[3] = "<option>Motorola</option><option>Samsung</option>
// Detects if there is a family
if(isset($_POST["family"]))
{
    echo '<select name="product">';
    echo $products[$_POST["family"]] . ';
    echo '</select>';}
?>
```
Introduction

• Client-server architecture:
  – The client request a service and the server provides the service
  – Client part of the application only contains presentation logic and sometimes a small amount of business logic

CLIENT

Communication protocol

SERVER

Introduction

• Web applications:
  – Based on client-server architecture
  – Every user interaction generates a server request and reloads the web page
  – Sometimes, depending on the application and the user interface, too many requests can be generated
  – Problems:
    • Long waits of the client
    • Slow applications
    • Errors on the transmission
Introduction

• **Ajax**, shorthand for *Asynchronous JavaScript and XML*.
• Web development technique for creating interactive web applications.
• The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user makes a change.
• This is meant to increase the web page's interactivity, speed, and usability.

The first known use of the term in public was by Jesse James Garrett in his February 2005 article *Ajax: A New Approach to Web Applications*.

Introduction

- The Ajax technique uses a combination of:
  - XHTML (or HTML), CSS, for marking up and styling information.
  - The DOM accessed with a client-side scripting language, especially ECMAScript implementations such as JavaScript and JScript, to dynamically display and interact with the information presented.
  - The XMLHTTPRequest object to exchange data asynchronously with the web server. In some Ajax frameworks and in certain situations, an IFrame object is used instead of the XMLHttpRequest object to exchange data with the web server.
  - XML is sometimes used as the format for transferring data between the server and client, although any format will work, including preformatted HTML, plain text, JSON and other formats.
- Like DHTML, LAMP, or SPA, Ajax is not a technology in itself, but a term that refers to the use of a group of technologies together.

How it works

- A new layer, called “Ajax engine”, is inserted between the client and the server:
  - XMLHttpRequest object is used
  - This object communicates to the server
- The server response can be in different formats:
  - Plain text
  - HTML
  - XML
  - JSON
  - ...
Advantages

- Asynchronous transactions: users don’t have to wait after a request
- Less amount of sent data
- Technologies based on open standards
- Similar to traditional applications in terms of interactivity and speed of response

Disadvantages

- The number of requests against the server can increase
- There are some incompatibilities between browsers:
  - The XMLHttpRequest object is not invoked in the same way in different browsers
- Loss of control over navigation:
  - The back and forward buttons of the browser stops working
  - Bookmarks to specific web pages can’t be created, because it is always the same page ➔ There are some partial solutions
XMLHttpRequest

- XMLHttpRequest is an API that can be used by JavaScript, JScript, VBScript and other web browser scripting languages to transfer and manipulate XML data to and from a web server using HTTP, establishing an independent connection channel between a web page's Client-Side and Server-Side.

- The XMLHttpRequest concept was originally developed by Microsoft.

- The Microsoft implementation is called XMLHttpRequest and, as an ActiveX object, it differs from the published standard in a few small ways. It has been available since Internet Explorer 5.0 and is accessible via JScript, VBScript and other scripting languages supported by IE browsers.
XMLHttpRequest

- The Mozilla project incorporated the first compatible native implementation of XMLHttpRequest in Mozilla 1.0 in 2002.
- This implementation was later followed by Apple since Safari 1.2, Konqueror, Opera Software since Opera 8.0 and iCab since 3.0b352.

XMLHttpRequest

- While this is still a work in progress, its goal is "to document a minimum set of interoperable features based on existing implementations, allowing Web developers to use these features without platform-specific code".
- The draft specification is based upon existing popular implementations, to help improve and ensure interoperability of code across web platforms.
XMLHttpRequest

Methods:
- abort()
- getAllResponseHeaders()
- getResponseHeader(header)
- open(method, url, asynchronous, user, password):
- send(content)
- setRequestHeader(header, value)
XMLHttpRequest

- open(method, url, async, user, password):
  - Initializes an XMLHTTP request.
  - Specifies the method, URL, and authentication information for the request.
  - After calling this method, you must call send to send the request and data, if any, to the server.

XMLHttpRequest

- send(content):
  - Sends an HTTP request to the server and receives a response.
  - null for no data.
XMLHttpRequest

• Properties:
  – onreadystatechange
  – readyState
  – responseText
  – responseXML
  – status
  – statusText

• onreadystatechange:
  – Function that handles the different events.
**XMLHttpRequest**

• `readyState`:
  – The property is read-only.
  – It represents the state of the request as an integer.
  – The following values are defined:

<table>
<thead>
<tr>
<th>State</th>
<th>Numeric Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNSENT</td>
<td>0</td>
<td>The object has been constructed.</td>
</tr>
<tr>
<td>OPENED</td>
<td>1</td>
<td>The <code>open()</code> method has been successfully invoked. During this state request headers can be set using <code>setRequestHeader()</code> and the request can be made using the <code>send()</code> method.</td>
</tr>
<tr>
<td>HEADERS_RECEIVED</td>
<td>2</td>
<td>All HTTP headers have been received. Several response members of the object are now available.</td>
</tr>
<tr>
<td>LOADING</td>
<td>3</td>
<td>The response entity body is being received.</td>
</tr>
<tr>
<td>DONE</td>
<td>4</td>
<td>The data transfer has been completed or something went wrong during the transfer (e.g. infinite redirects).</td>
</tr>
</tbody>
</table>
XMLHttpRequest

- `responseText`:
  - The property is read-only.
  - This property represents only one of several forms in which the HTTP response can be returned.

- `responseXML`:
  - The property is read-only.
  - This property represents the parsed response entity body.
XMLHttpRequest

- **status:**
  - Return the HTTP status code.
  - RFC 2616 Hypertext Transfer Protocol -- HTTP/1.1:
    - [http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html](http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html)
    - 2xx Successful (200 OK)
    - 3xx Redirection (301 Moved Permanently)
    - 4xx Client error (404 Not Found)
    - 5xx Server error (500 Internal Server Error)

- **statusText:**
  - Return the HTTP status text.

---

**AJAX step by step**

1. Create XMLHttpRequest object
2. Assign a function to the state change event
3. Send a request to the server
4. On a state change, manage the response
5. On a correct response, process the result and show to the user
Create XMLHttpRequest object

- Depending on the browser:
  - Internet Explorer
    ```javascript
    request = new ActiveXObject("Microsoft.XMLHTTP");
    ```
  - Other browsers:
    ```javascript
    request = new XMLHttpRequest();
    ```

- Code adapted for different browsers:
  ```javascript
  if(window.XMLHttpRequest) {
    request = new XMLHttpRequest();
  }
  else if(window.ActiveXObject) {
    request = new ActiveXObject("Microsoft.XMLHTTP");
  }
  ```

- Another way, using exceptions:
  ```javascript
  try {
    request = new ActiveXObject("Msxml2.XMLHTTP");
  } catch (e) {
    try {
      request = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (ee) {
      try {
        request = new XMLHttpRequest();
      } catch (eee) {
        alert("Your browser doesn’t support Ajax");
        return;
      }
    }
  }
  ```
Assign a function to the state change event

- This function will be called automatically, every time the state of the XMLHttpRequest object changes:

```javascript
request.onreadystatechange = nameOfFunction;
```

**Important:** without “( )”, only the name

---

Send a request to the server

- Open the connection, define the method and the type of connection:
  - A synchronous connection (`false`) blocks the browser until the response is obtained
  - An asynchronous connection (`true` and default value) executes on the background
  - **Important:** the URL must belong to the same domain of the current page

```javascript
request.open('GET','http://www.ua.es/ajax.jsp', true);
```

- Send the additional data:

```javascript
request.send(data or null);
```
On a state change, manage the response

- The handler is called every time there is a change:
  - 0: UNSET
  - 1: OPENED
  - 2: HEADERS_RECEIVED
  - 3: LOADING
  - 4: DONE
- Example of handler:
  ```javascript
  if (request.readyState == 4) { // Finished
    if (request.status==200) { // OK
      // Process the result
    }
  } else { // Not finished
  }
  ```

On a correct response, process the result and show to the user

- The result can be in different formats: plain text, HTML, JSON, XML, etc.
- `responseText` when not structured result as XML:
  ```javascript
  alert(request.responseText);
  ```
- `responseXML` when structured result as XML:
  - Returns an XMLDocument object
  - Use DOM functions
Example

```javascript
function ajaxFunction() {
    var xmlHttp;
    if (window.XMLHttpRequest)
        xmlHttp = new XMLHttpRequest();
    else
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");

    xmlHttp.onreadystatechange=function() {
        if(xmlHttp.readyState == 4) {
            document.myForm.time.value += xmlHttp.responseText + "\n";
        }
    }

    xmlHttp.open("GET","time.php",true);
    xmlHttp.send(null);
}
</script>
```
Example

```javascript
<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) {
            document.myForm.time.value += xmlHttp.responseText + "\n";
        }
    };

    xmlHttp.open("GET","time.php",true);
    xmlHttp.send(null);
}
</script>
```

Assign a function to the state change event

Send a request to the server
Example

```javascript
function ajaxFunction() {
    var xmlHttp;
    if (window.XMLHttpRequest)
        xmlHttp = new XMLHttpRequest();
    else
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");

    xmlHttp.onreadystatechange=function() {
        On a state change, manage the response
        if(xmlHttp.readyState == 4) {
            document.myForm.time.value += xmlHttp.responseText + "\n";
        }
    }
    xmlHttp.open("GET","time.php",true);
    xmlHttp.send(null);
}
</script>
```

Example

```javascript
function ajaxFunction() {
    var xmlHttp;
    if (window.XMLHttpRequest)
        xmlHttp = new XMLHttpRequest();
    else
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");

    xmlHttp.onreadystatechange=function() {
        On a correct response, process the result and show to the user
        if(xmlHttp.readyState == 4) {
            document.myForm.time.value += xmlHttp.responseText + "\n";
        }
    }
    xmlHttp.open("GET","time.php",true);
    xmlHttp.send(null);
}
</script>
```
Example

<html>
<head>
<title>Ajax example</title>
<!--[script]-->
</head>
<body>
<form name="myForm">
Name: <input type="text" onkeyup="ajaxFunction();" name="username" />
<br />
Time: <textarea name="time" cols="40" rows="10"></textarea>
</form>
</body>
</html>

• PHP:

<?php
header("Expires: -1");
$str1 = date('h:i:s A');
sleep(2);
$str2 = date('h:i:s A');
echo "$str1 -- $str2";
?>
Sending data GET/POST

- **GET:**
  - Data is sent in the URL in the `open()` method
  - `encodeURIComponent()` codes data to be sent in the URL

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

- **POST:**
  - Data is sent in the `send()` method
  - `encodeURIComponent()` codes data to be sent
  - `setRequestHeader()` is used to define some HTTP headers
Sending data GET/POST

- POST:
  ```javascript
  var idFamily = document.getElementById("family").value;
  var parameters = "family=" + encodeURIComponent(idFamily);
  xmlhttp.open("POST","products.php", true);
  xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
  xmlhttp.setRequestHeader("Content-length", parameters.length);
  xmlhttp.setRequestHeader("Connection", "close");
  xmlhttp.send(parameters);
  ```

Transformation to AJAX

- Separate client and server logic
Transformation to AJAX

- **EXERCISE**
- *Classical example (1)*: A web page shows news headlines updated every 10 seconds

```php
 CLIENT SERVER
 New York Times
 Lastest news:

 Headline 1
 Bla, bla, bla.

 Headline 2
 Bla, bla, bla, bla, bla.

 <?php
 // News
 ?>
```
Advanced Web Programming

Transformation to AJAX

• **EXERCISE**

• **Classical example (2):** A web page shows two lists (<select>):
  
  – Initially, the second list is empty or it doesn’t appear
  
  – When selecting a value in the first list, the web page is reloaded and the second list shows new data

Advanced Web Programming

XML data format

• **Property** `responseXML` -> `XMLDocument` object

• **DOM** properties and methods are used to access and traverse the XML document
XML data format

• Each node has a set of properties that relates to its “relatives”:
  - childNodes
  - firstChild
  - lastChild
  - parentNode
  - nextSibling
  - prevSibling

• getElementById("elementID")
  - Useful to get a specific element
    var element = document.getElementById("myTable")

• getElementsByTagNames("tagName"):
  - Useful to get a set of elements of the same type
    var images = document.getElementsByTagName("img")
  - The special value "*" all the tags (elements)

• getAttribute("attrName"):
  - var element = document.getElementById("myTable").
    getAttribute("width");
XML data format

```javascript
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 + ")
    );
}
```

- In the server, appropriate MIME type has to be used:

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

- **JSON (JavaScript Object Notation)** is a light exchange data format based on plain text
- **JSON Specification RFC 4627:**
- **JSON Validator:**

It’s used to serialized structured data:

- **Arrays:**
  - The list of values between square brackets ([ ]) and separated by commas (,)
- **Objects:**
  - Pairs property/value closed between curly braces ({ }) and separated by commas
  - Property and value are separated by colon
  - In both cases, properties and values of string type must be between double quotes
JSON data format

• Array example:

[1, 2, 3, 4, 5]

["Voronezh", "Moscow", "Saratov"]

• Object example:

{"name": "Sergio", "surnames": "Luján Mora"}

{"position": {"x": 10, "y": 20}}
Advanced Web Programming

Universidad de Alicante

**JSON data format**

- **Example:**

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

- **Example:**

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

- Data sent in JSON format is available in `responseText`
- For “rebuilding” the original array or object, use `eval()`:

```javascript
var json_data = xmlHttp.responseText;
myObject = eval('(' + json_data + ')');
```

### More information

- AJAX (Wikipedia):
- Ajax: A New Approach to Web Applications:
- AJAX un nuevo acercamiento a aplicaciones web (traducción del anterior):
- “Hola mundo” en Ajax:
- Introducción a Ajax:
  - [http://www.librosweb.es/ajax/index.html](http://www.librosweb.es/ajax/index.html)
More information

- **W3C:**
  - The XMLHttpRequest Object (W3C Working Draft 05 April 2006)
  - [http://www.w3.org/TR/XMLHttpRequest/](http://www.w3.org/TR/XMLHttpRequest/)
- **Microsoft:**
  - MSDN: IXMLHTTPRequest
  - MSXML 4.0 SDK
**IXMLHTTPRequest**

**See Also**

- Language Filter, A

Provides client-side protocol support for communication with HTTP servers.

**Example**

The following Microsoft Visual Studio example creates an `AXMLObject` object and asks a server for an XML document. The server sends back an XML document, which is then displayed in a message box.

**XML**

```
<c:copyCode><![CDATA[
<AXMLObject> axl = new AXMLObject();
axl.AXMLHttpRequest("http://localhost/books.xml");
axl.showMessage("XML");
Message, hideXMLHttpRequestRequest تستعن
]]>[/copyCode]
```

**Example**

The following Microsoft Visual Basic example creates an `AXXMLObject` object and asks a server for an XML document. The server sends back an XML document, which is then displayed in a message box.

**XML**

```
<c:copyCode><![CDATA[
<AXMLObject> Obj = New AXMLObject()
Obj.AXMLHttpRequest "http://localhost/books.xml"
MsgBox "XML:"
]]>[/copyCode]
```

**IXMLHTTPRequest Members**

**Properties**

- `onreadystatechange` Specifies the event handler to be called when the readyState property changes. `readyState`.
- `responseText` Represents the text of the request, read-only.
- `responseXML` Represents only one of several forms in which the HTTP response can be selected, read-only.
- `status` Represents the status code returned by a request. Read-only.
- `statusText` Represents the HTTP status text, read-only.

**Methods**

- `abort()` Closes the current `XMLHttpRequest`.
- `setRequestHeader()` Specifies the name of an HTTP request header.
- `send()` Sends an HTTP request to the server and receives a response.

**Events**

- `readystatechange` To view reference information for Visual Basic, C# or C++, or Script only, click the Language Filter button in the upper left corner of the page.