Internet Software Technologies
AJAX

IMCNE

A.A. 2008/09

Gabriele Cecchetti

What is Ajax ?

- AJAX stands for **A**synchronous **J**avaScript **A**nd **X**ML.
- AJAX is a type of programming made popular in 2005 by Google (with Google Suggest).
- AJAX is not a new programming language, but a new way to use existing standards.
- With AJAX you can create better, faster, and more user-friendly web applications.
- AJAX is based on JavaScript and HTTP requests.

History

- Before Ajax only Adobe-Macromedia Flash or Java Applet were used to create rich internet application. But they were not always supported by users' browsers or their use was limited only to special effects presentation.
- Since 1996 **`iframe`** tag supported by Internet Explorer 3, was exploited to refresh an embedded page, resembling an asynchronous interaction.
- In 1998 Microsoft began to develop **Remote Scripting** technology to create a technique to interact with remote contents. It was Ajax but with a different name! Then it evolved and it became a true object named XMLHttpRequest.
- Only Google was able to exploit Remote Scripting, when it was massively used for its applications.
- Nowadays, this technology has shown its strong and it is widely used offering also accessibility and better usability than Flash and Java Applets.

AJAX = Asynchronous JavaScript and XML

- AJAX is not a new programming language, but a technique for creating better, faster, and more interactive web applications.
- With AJAX, your JavaScript can communicate directly with the server, using the JavaScript **XMLHttpRequest** object. With this object, your JavaScript can trade data with a web server, without reloading the page.
- AJAX uses asynchronous data transfer (HTTP requests) between the browser and the web server, allowing web pages to request small bits of information from the server instead of whole pages.
  - The AJAX technique makes Internet applications smaller, faster and more user-friendly.
  - AJAX is a browser technology independent of web server software.

AJAX is based on Web Standards

- AJAX is based on the following web standards:
  - **JavaScript**
  - **XML**
  - **HTML**
  - **CSS**
- The web standards used in AJAX are well defined, and supported by all major browsers. AJAX applications are browser and platform independent.

AJAX is about better Internet applications

- Web applications have many benefits over desktop applications; they can reach a larger audience, they are easier to install and support, and easier to develop.
- However, Internet applications are not always as "rich" and user-friendly as traditional desktop applications.
- With AJAX, Internet applications can be made richer and more user-friendly.

AJAX Uses HTTP Requests

- In traditional JavaScript coding, if you want to get any information from a database or a file on the server, or send user information to a server, you will have to make an HTML form and GET or POST data to the server. The user will have to click the "Submit" button to send/get the information, wait for the server to respond, then a new page will load with the results.
- Because the server returns a new page each time the user submits input, traditional web applications can run slowly and tend to be less user-friendly.
- With AJAX, your JavaScript communicates directly with the server, through the JavaScript **`XMLHttpRequest`** object
- With an HTTP request, a web page can make a request to, and get a response from a web server - without reloading the page. The user will stay on the same page, and he or she will not notice that scripts request pages, or send data to a server in the background.

What does *Asynchronous Request* means ?

- *Asynchronous Request* means the browser does not have to wait the request to be completed before start new operations.
- Usually the user make a request using a link or a form or refreshing a page and then he/she waits the server answers before making a new request.
- While using asynchronous request the user (or better the code contained inside a page can make several requests without waiting for their competition).
- With this approach the waiting time is less important but however a special care have to be taken during the designing of such dynamic web pages to synchronize information if some of them are dependent from others.

The `XMLHttpRequest` Object

- **By using the `XMLHttpRequest` object, a web developer can update a page with data from the server after the page has loaded!**
- Example: *Google Suggest* is using the `XMLHttpRequest` object to create a very dynamic web interface: When you start typing in Google's search box, a JavaScript sends the letters off to a server and the server returns a list of suggestions.
- The `XMLHttpRequest` object is supported in Internet Explorer 5.0+, Safari 1.2, Mozilla 1.0 / Firefox, Opera 8+, and Netscape 7.

AJAX Browser support

- Different browsers use different methods to create the XMLHttpRequest object.
- Internet Explorer uses an **ActiveXObject**, while other browsers uses the built-in JavaScript object called **XMLHttpRequest**.
- To create this object, and deal with different browsers, we are going to use a "try and catch" statement.

AJAX Browser support – JavaScript code

```
function ajaxFunction() {
  var xmlHttp;
  try { // Firefox, Opera 8.0+, Safari
    xmlHttp = new XMLHttpRequest();
  } catch (e) { // Internet Explorer
    try {
      xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
      try {
        xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
      } catch (e) {
          alert("Your browser does not support AJAX!");
          return false;
      }
    }
  }
}
```


AJAX Browser support – onLoad check

```
onLoad = function() {
  var ajax = ajaxFunction();
    if(ajax) {
      // your Ayax Application
  }
}
```

- If the `ajax` object is not assigned or if it is `null` or `false`, the application does not run.

Ajax properties

- `onreadystatechange`
- `readyState`
- `responseText`
- `responseXML`
- `status`
- `statusText`

Ajax property: `readyState`

- The `readyState` property holds the status of the server's response. Each time the `readyState` changes, the `onreadystatechange` function will be executed.
- Here are the possible values for the `readyState` property:

State	*Description*
0	The request is not initialized
1	The request has been set up
2	The request has been sent
3	The request is in process
4	The request is complete

Ajax properties: `status` and `statusText`

- `status` property contains the HTTP error code sent back from the http server:
  = 200 if the the request was successful.
- `statusText` is the descritpion of the `status` property.

Ajax properties: `responseXML` and `responseText`

- These properties contain the data produced by the server.
- `responseText` is a string (a text), while
- `responseXML` is a XML object, otherwise it is a `null` object.

Ajax property: onreadystatechange

- After a request to the server, we need a function that can receive the data that is returned by the server.
- The onreadystatechange property stores your function that will process the response from a server. This is not a method, the function is stored in the property to be called automatically. The following code sets the onreadystatechange property and stores an empty function inside it:

```
ajax.onreadystatechange = function() {
  // We are going to write some code here
}
```


Example of onreadystatechange use

```
ajax.onreadystatechange = function() {
  if(ajax.readyState === 4) {
    if(ajax.status == 200)
      alert("Operazione effettuata con successo");
    else
      alert("Operazione fallita, errore numero " +
  ajax.status);
  }
}
```

Ajax methods

- **`abort`**
- **`getAllResponseHeaders`**
- **`getResponseHeader`**
- **`open`**
- **`send`**
- **`setRequestHeader`**

Ajax method: open

- It was defined by W3C.
- It is used to make an asyncronous call.
- Syntax:

  `open(method, uri [,async][,user][,password])`

where:

- `method` → `get` or `post`
- `uri` → it is the requested `uri`
- `async` → `true` or `false`

Ajax methods: `send` and `setRequestHeader`

- It is used to make a HTTP request; syntax:

```
send(data)
```

- By default it make a GET request, but if it used in conjunction of `setRequestHeader`, it makes a POST request; example:

```
ajax.open("post", “folder/ajax.html, true);
ajax.setRequestHeader("content-type", "application/x-
  www-form-urlencoded");
ajax.send(“read=Joyce");
```


Ajax method: `abort`

- It is used to abort the sending or receiving operations.
- Syntax:

```
abort();
```

- Example:

```
ayax.abort();
```

Ajax methods: `getAllResponseHeaders` and `getResponseHeader`

- They provide the server's headers after the http answer phase. Example:

```
// List of all headers provided by the server
  alert(ajax.getAllResponseHeaders());
// Information about a single header
  alert(ajax.getResponseHeader("content-type"));
```


A simple complete example – client side

```
<html><body><script type="text/javascript">
  function ajaxFunction() {
var xmlHttp;
try {
  xmlHttp = new XMLHttpRequest();
  } catch (e)  { /* see previous code */ }
xmlHttp.onreadystatechange=function() {
  if(xmlHttp.readyState==4)
    document.myForm.time.value=xmlHttp.responseText;
}
xmlHttp.open("GET","time.php",true);
xmlHttp.send(null); // Ghost request
}</script><form name="myForm"> Name: <input type="text"
  onkeyup="ajaxFunction();" name="username" /> Time:
  <input type="text" name="time"
  /></form></body></html>
```

A simple complete example – server side

```
<?php
header("Cache-Control: no-cache, must-revalidate");
 // Date in the past
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");

$d = date("h:m:s");
echo $d;
?>
```