

WEB222 - Web Programming Principles

Week 12: AJAX

Agenda

- AJAX
 - Simulating a call to a Web service
- JSON

What is AJAX

- AJAX stands for **A**synchronous **J**ava**S**cript and **X**ML.
- Coined in 2005 by Jesse James Garrett.
- Not a technology , but a “new” way to use existing technologies.
- AJAX is a group of interrelated web techniques used on the client-side for creating fast and dynamic web pages.

Technologies used in AJAX

- HTML
- CSS
- JavaScript
- The Document Object Model (DOM)
- XML/JSON
- XMLHttpRequest object

AJAX Examples

- Google Maps

<http://maps.google.com/>

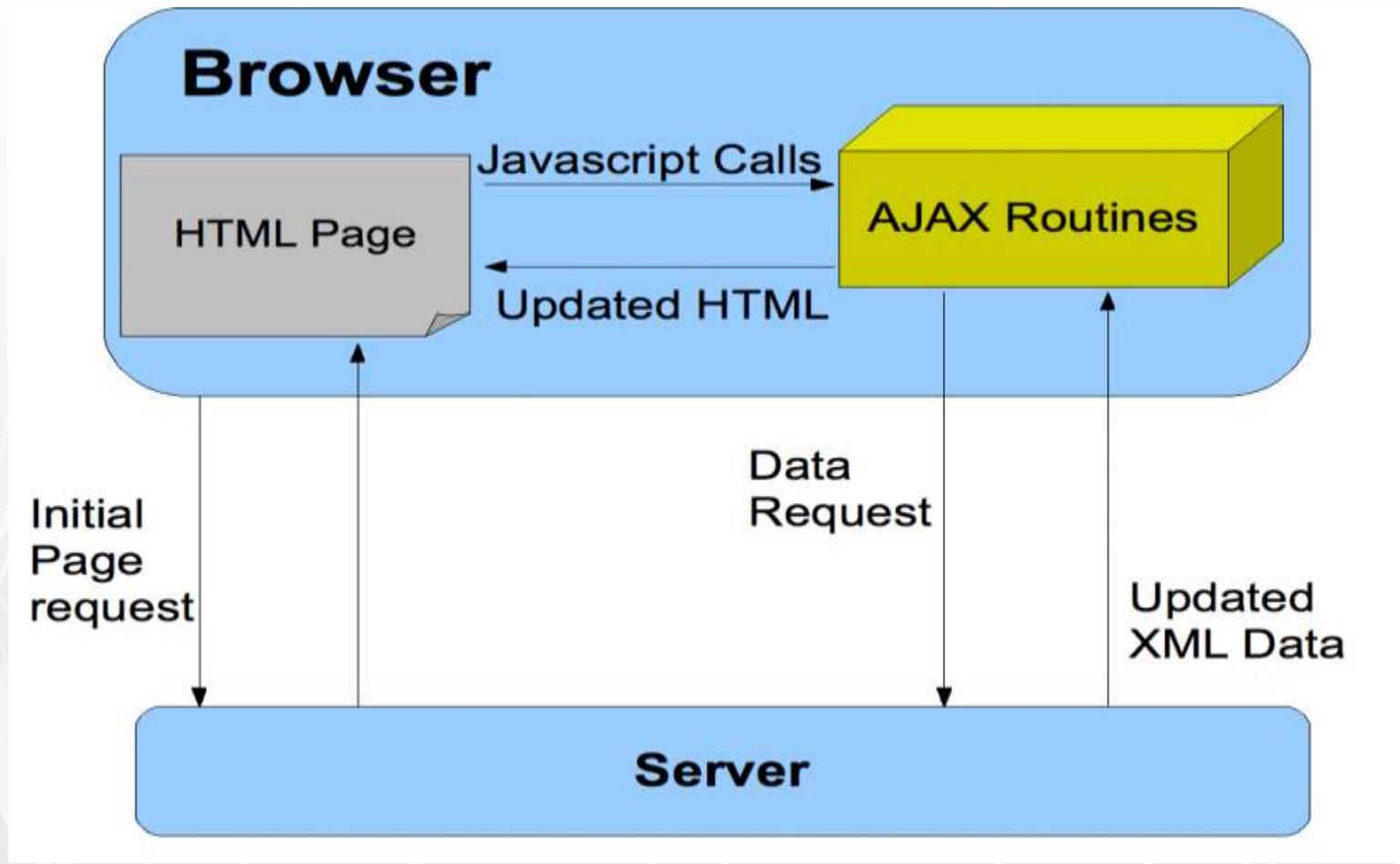
- Google Suggest

<http://www.google.com/>

How does it works?

- AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

The AJAX model



Comparing with classic apps

- Conventional web application transmit information to and from the sever using synchronous requests.
- This means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- Using AJAX – we are able to request data from the server & update the DOM with the result without leaving the page!

Packaging data in AJAX model

- XML was originally used as the format in AJAX.
- JSON is used more than XML nowadays .
 - Advantages of JSON: Concise, readable and a part of JavaScript.
- Other formats, e.g. plain text, can be used.

Features

- Make requests to the server without reloading the page
- Receive and work with data from the server
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background
- Intuitive and natural user interaction. No clicking required only Mouse movement is a sufficient event trigger.
- Data-driven as opposed to page-driven

XMLHttpRequest object

- It is the most important part of AJAX.
- It is a JavaScript object
 - Designed by MS and adopted by Mozilla, Apple, ...
- It provides an easy way to retrieve data from a URL without having to do a full page refresh.
- Creation:

```
var myRequest = new XMLHttpRequest();
```

XMLHttpRequest Methods

- `abort()`
- `getResponseHeader()`
- `open()` // Initializes a request.
- `overrideMimeType()`
- `send()` // Sends the request.
- ...

XMLHttpRequest properties

- `onreadystatechange` - event handler
- `readyState`
- `responseText` - used when receiving JSON or plain text
- `responseXML` - used when receiving XML
- `responseType` - set to change the response type

Writing AJAX

Step 1 – makes an HTTP request object

```
// creating a cross-browser instance
var httpRequest;

if (window.XMLHttpRequest) { // Mozilla, Safari, ...
    httpRequest = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE 8 and older
    httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Writing AJAX

- **Step 2 – register a request listener**

```
httpRequest.onreadystatechange = reqListener;
```

```
function reqListener () {  
    // process the server response  
};
```

- **Or:**

```
httpRequest.onreadystatechange = function(){  
    // process the server response  
};
```

Writing AJAX

➤ Step 3 - make the request

```
// Specifies the type of request
httpRequest.open('GET',
  'http://www.example.org/some.file', true);
```

```
// Sends the request off to the server.
httpRequest.send(null);
```

Parameters of open() and send()

- Parameters of the `open(method, url, async)` method:
 - 1st para: HTTP request methods - GET, or POST
 - 2nd para: the URL of the page this requesting
 - 3rd para: sets whether the request is asynchronous
- Parameters of the `send(string)` method:
 - The "*string*" para: used for POST requests.
 - any data you want to send to the server if using POST method. This can be as a query string, like:

```
name=value&anothername="+encodeURIComponent(myVar)+"&so=on"
```

Writing AJAX

➤ **Step 4 – handling the server response**

```
// check for the state of response
if (httpRequest.readyState === 4) {
  // everything is good, the response is received
  // do next process here
```

```
} else {
  // still not ready
}
```

- the readyState values:
 - ▶ 0 (uninitialized), 1 (loading), 2 (loaded), 3 (interactive), 4 (complete)

Writing AJAX

➤ **Step 4 – handling the server response (cont')**

```
// check is the response code of the HTTP server response
if (httpRequest.status === 200) {
    // perfect!
    // do next process here

} else {
    // there was a problem with the request,
    // for example the response may contain a 404 (Not Found)
    // or 500 (Internal Server Error) response code
}
```

Writing AJAX

- Set the button to start

```
<button type="button"  
       onclick="makeRequest();">Make a request</button>
```

Working with the XML response

- If the server response is in XML format, e.g.

```
<?xml version="1.0" ?>  
<root>  
    I'm a test.  
</root>
```

- Parsing XML data

```
var xmldoc = httpRequest.responseXML;  
var root_node = xmldoc.getElementsByTagName('root').item(0);  
console.log(root_node.firstChild.data);
```

Working with the JSON response

- If server response is in JSON format, e.g.

```
'{"name": "Kevin", "age": 22 }'
```

- Parsing JSON data:

```
// function JSON.parse(): convert JSON object to JavaScript object
var jsonObj = http_request.responseText;
var jsObj = JSON.parse(jsonObj);
var name = jsObj.name;
var age = jsObj.age;
```

About JSON



- JSON stands for **JavaScript Object Notation**, used for storing and exchanging data.
 - specified by **Douglas Crockford**.
 - a lightweight text based data-interchange format.
 - "self-describing" and easy to understand
 - smaller than XML, and faster and easier to parse
- JSON parsers and JSON libraries exists for many different programming languages.
 - e.g. C, C++, Java, Python, Perl, PHP etc.
- JSON filename extension is **.json**
- JSON's (Internet) Media type (or Content Type in HTTP header) is **application/json**

JSON Structures

- JSON is text, written with JavaScript object notation.
- JSON is built on two structures:
 - **JSON object**: a collection of name/value pairs.

```
'{ "firstName": "John",  
  "lastName": "Smith", "age": 25,  
  "address": { "street": "21 2nd street",  
    "city": "North York",  
    "province": "ON",  
    "postalCode": "M2M6T6" }  
'
```

- **JSON array**: an ordered list of values.

```
'[  
  {"name": "Kevin", "age": 22, "gender": "m"},  
  {"name": "Kate", "age": 22, "gender": "f"},  
  {"name": "Steven", "age": 25, "gender": "m"},  
  {"name": "Bill", "age": 22, "gender": "m"}  
'
```

JSON Objects vs JavaScript Objects

- JSON and JavaScript use the same syntax to describe data objects, including Arrays.
- But JSON objects are text-based or "string"s.
- The properties of JSON objects must be quoted.

e.g.

```
{"name": "Kevin", "age": 22 }
```

JSON Objects vs JavaScript Objects

- JavaScript built-in object: **JSON** - used to convert data between JSON and JavaScript objects.
 - Serializing JavaScript object (converting JavaScript object to JSON object) :

```
var jsObject1 = { language: "Java", course: "JAC444" };
```

```
var JSONString1 = JSON.stringify(jsObject1);
```

```
console.log("JSONString1: ", JSONString1);
```

- Parsing JSON string (converting JSON object to JavaScript object) :

```
var JSONString2 = '{ "language" : "C++", "course" : "OPP344" }';
```

```
var jsObject2 = JSON.parse(JSONString2);
```

```
console.log("jsObject2: ", jsObject2);
```

JSON vs XML

➤ A JSON object

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

➤ Equivalent XML document

```
<?xml version="1.0" ?>  
<root>  
  <menu id="file" value="File">  
    <popup>  
      <menuitem value="New" onclick="CreateDoc()" />  
      <menuitem value="Open" onclick="OpenDoc()" />  
      <menuitem value="Close" onclick="CloseDoc()" />  
    </popup>  
  </menu>  
</root>
```

JavaScript: simulating AJAX calls to Web Services

- Example 1: calling a web service with **JSON object** response
 - JSON data on Server("web service"):
<https://zenit.senecac.on.ca/~wei.song/web222/ajax/firstnation.json>
 - Example Code:
<https://zenit.senecac.on.ca/~wei.song/web222/ajax/ajaxjson.html>
- Example 2: calling a web service with **JSON array** response
 - JSON data on Server("web service"):
<https://zenit.senecac.on.ca/~wei.song/web222/ajax/nationArray.json>
 - Example Code:
<https://zenit.senecac.on.ca/~wei.song/web222/ajax/ajaxjson2.html>

More about AJAX calls

- For security reason, web browsers do not allow cross-origin AJAX calls.
 - The XMLHttpRequest follows the same-origin policy. An Ajax call can only request HTTP resources from the same origin that the web page was loaded from. **Same origin** means same protocol, same host and same port(if one is specified).
- The following Ajax calls will not work:
 - Run the code examples, ajaxjson.html or ajaxjson2.html, from your local PC.
 - Use <https://zenit.senecac.on.ca/~wei.song/web222/ajax/ajaxjson.html> to call <http://zenit.senecac.on.ca/~wei.song/web222/ajax/nationArray.json>
- To allow cross-origin AJAX calls, a web service should have a response header:
 - Access-Control-Allow-Origin: *

Exercise

- Download the JSON file:
<https://zenit.senecac.on.ca/~wei.song/web222/ajax/student.json>
- Create an HTML5 file that loads data using an AJAX call to fetch the JSON data.
- Update the 2 files to the public_html folder of your Matrix or Zenit account, and make your app work.
- The HTML page is showed at right.

Info of an ICT Student

Name	Kevin
Age	22
Program	CPD
Course 1	INT222
Course 2	DBS201
Course 3	OOP244

Make a request

Resource & Reference

- MDN: **AJAX - Getting_Started**
 - https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started
- **JavaScript Object Notation (JSON)**
- **Browser Object Model - Wikipedia, the free encyclopedia**

Thank You!