Managing Data on the World Wide-Web

AJAX and JQuery

Elad Kravi
Based on S.Bussel Presentation
Overview

1. AJAX - introduction
2. Asynchronous AJAX
3. jQuery
4. AJAX with jQuery
• AJAX is a web development technique for creating interactive web applications

• AJAX is a misleading name - You don't have to understand XML to use AJAX

• With AJAX it’s possible to:
  – update a web page without reloading the page
  – request/receive data from a server - after the page has loaded
Where were we before AJAX?

• Static pages can give the illusion of interactivity through standard form submissions
  – Form submissions result in full page loads

• So what’s the problem?
  – Many actions only manipulate small portions of the page but the entire page must be reloaded
  – Server responses contain the entire page content rather than just the portion being updated
  – Loading entire pages typically results in several additional HTTP requests for images, style sheets, scripts, and any other content that may be on the page
Real-Life Examples of AJAX Apps

• Google Maps
  – http://maps.google.com/

• Google Suggest (Now integrated in Google’s homepage)

• Loading content when scrolling down the page

• Facebook’s notifications

• Many more…
AJAX Fundamentals

• AJAX uses a three-step process:
  1. Request a URL by the JavaScript code – **client side**
  2. Handle the URL on the server and write the response – **server side**
  3. Use the data received from the response – **client side**
    • E.g., integrate the data into the DOM

• In an AJAX request we don't refresh the entire page; instead, we update only part of the page
AJAX users may believe that AJAX, because it provides a more responsive user interface, reduces server-side traffic

However AJAX applications may have more server-side traffic because each AJAX request involves a trip to the server

- Because those requests are asynchronous AJAX creates the perception of a more responsive UI, though it typically does not reduce the load on the server
How AJAX Works

• JavaScript is used to:
  – Create and control instances of the XMLHttpRequest (XHR) object
  – Provide handlers for responses
  – Manipulate the DOM

• The XMLHttpRequest object:
  – Used to exchange data with a server behind the scenes
  – Supports HTTP operations
How AJAX Works

Browser
- An event occurs...
  - Create an XMLHttpRequest object
  - Send HttpRequest

Server
- Process HttpRequest
- Create a response and send data back to the browser

Browser (bottom)
- Process the returned data using JavaScript
- Update page content

Source: http://www.w3schools.com/ajax/ajax_intro.asp
Sending HTTP Requests

```javascript
var request = new XMLHttpRequest();
request.open("method","URL",isAsynchronous)
request.setRequestHeader("header","value")
request.send(content)
```

• Method can be any HTTP verb (i.e., GET, POST, etc.)
• URL must be in the domain of the current (or a relative URL) for security reasons
  – For extra reading: [https://en.wikipedia.org/wiki/Same-origin_policy](https://en.wikipedia.org/wiki/Same-origin_policy)
• isAsynchronous will be discussed soon
• Content is the posted in a POST request
  – can be null or empty
Reading the Response

- `request.responseText`: The response as flat text
- `request.responseTextXML`: The response as a (DOM) Document object
  - Available if response Content-Type is text/XML

```javascript
xmlDoc = xhr.responseXML;
txt = "";
x = xmlDoc.getElementsByTagName("ARTIST");
for (i = 0; i < x.length; i++) {
    txt += x[i].childNodes[0].nodeValue + "<br>");
}
document.getElementById("demo").innerHTML = txt;
```

```
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  ...
</CATALOG>
```
Reading the Response

- Status code `request.status`
- Status Text `request.statusText`
- Extracting headers:
  `request.getAllResponseHeaders()`
  `request.getResponseHeader("header")`
<body onload="init(); setJoke(1)">

<h2>
  Current date on server:
  <span id="serverTimeSpan"> ?</span>
</h2>

<h1>Select a Joke:</h1>
<div>
  <select onchange="presentServerTime(); setJoke(value);">
    <option value="1" selected="selected">
      Joke 1
    </option>
    <option value="2">
      Joke 2
    </option>
    <option value="3">
      Joke 3
    </option>
  </select>
</div>
<div id="jokediv"></div>
</body>
function presentServerTime() {
    var request = new XMLHttpRequest();
    request.open("GET", "current-time.jsp", false);
    request.send(null);

    if (request.status == 200) {
        timeSpan.innerHTML = request.responseText;
    } else {
        timeSpan.innerHTML = "<i>Cannot load server time...</i>";
    }
}

function setJoke(value) {
    var request = new XMLHttpRequest();
    request.open("GET", "joke" + value + ".txt", false);
    request.send(null);

    if (request.status == 200) {
        jokeDiv.innerHTML = request.responseText;
    } else {
        jokeDiv.innerHTML = "<i>Cannot load joke...</i>";
    }
}
<%= new java.util.Date() %>
<%
    long t0,t1;
    t0 = System.currentTimeMillis();
    do {
        t1 = System.currentTimeMillis();
    } while (t1-t0<10000); //wait for 10 seconds
%>
Overview

1. AJAX - introduction
2. Asynchronous AJAX
3. jQuery
4. Ajax with jQuery
Asynchronous AJAX

• We used “false" for the third parameter of open():
  – `xhttp.open("GET", "test.php", false);`
  – this parameter specifies whether the request should be handled asynchronously
  – as you can notice by running this example, the joke only appear after the JSP response is received

• True means that the script continues to run after the send() method, without waiting for a response from the server
  – this can be seen by the ability to choose new joke before the previous one was returned
Asynchronous Requests

- Reading of a Web page can take a long time during which the browser is blocked
- Solution: launch the request **asynchronously**
- That is, the execution continues after `send` is called without waiting for it to complete
  
  ```javascript
  request.open("method","URL",true)
  ```
- When the request is completed, a predefined function is called
function presentServerTime()
{
    request2 = new XMLHttpRequest();
    request2.open("GET","current-time.jsp",true);
    // subscribing an event handler:
    request2.onreadystatechange = updateServerTimeSpan;
    request2.send();
}

function updateServerTimeSpan()
{
    if(request2.readyState < 4) {
        timeSpan.innerHTML = "Loading...";
        return;
    } else if(request2.readyState == 4 && request2.status == 200) {
        timeSpan.innerHTML = request2.responseText;
    } else {
        timeSpan.innerHTML = "<i>Can not tell server time!";
    }
}
**XMLHttpRequest States**

- The `XMLHttpRequest` goes through several states:
  0 – request not initialized
  1 – server connection established
  2 – request received
  3 – processing request
  4 – processing request

- In the request configuration, you can define a function to call upon state change:
  ```javascript
  request.onreadystatechange = functionName
  ```

- `request.readyState` - get the current state of the request
- `request.abort()` - stop the request
Overview

1. AJAX - introduction
2. A-Synchronous AJAX
3. jQuery
4. Ajax with jQuery
JavaScript Libraries

• Advanced JavaScript programming can often be very difficult and time-consuming to work with
  – (especially the complex handling of browser differences)

• To deal with these difficulties, a lot of JavaScript libraries have been developed
  – These JavaScript libraries are often called JavaScript frameworks
Some JS libraries

- **jQuery**, **Dojo** - cross-platform JavaScript libraries designed to simplify the client-side scripting of HTML
- **AngularJS** - aims to simplify both the development and the testing of single-page applications
- **Underscore.js** - provides utility functions for common programming tasks
  - Includes functions like map, filter, invoke, reduce, template, throttle, bind, extend, pick, clone
- And many others: Prototype, MooTools, Ember, ReactJS, Backbone, D3.js, Modernizr, Bootstrap
- **List of JavaScript libraries** grouped in categories (Wikipedia)
- **JavaScript Frameworks Comparison** (Wikipedia)
Market share (% in category)

- jQuery 43%
- Modernizr 10%
- backbone.js 2%
- MooTools 3%
- spin.js 3%
- Underscore.js 3%
- prettyPhoto 3%
- Lightbox 4%
- yepnope.js 6%
- jQuery UI 7%
- Other 17%

Source: https://wappalyzer.com/categories/javascript-frameworks
jQuery - Introduction

• **jQuery** is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML

• It was released in January 2006 at BarCamp NYC by John Resig (at the age of 22 (!))

• The most popular JavaScript library in use today, with installation on 65% of the top 10 million highest-trafficked sites on the Web
  – source: [http://w3techs.com/technologies/overview/javascript_library/all](http://w3techs.com/technologies/overview/javascript_library/all)
Why jQuery is so Popular?

- Simple + great documentation
- CSS3 selectors compliant
- Helpful Utilities and Features
  - HTML/DOM manipulation
  - HTML event methods
  - AJAX support
  - Effects and animations
  - General utilities (e.g., string trimming)
  - Cross-browser compatibility
- Widespread adoption
Getting Started with jQuery

• Local reference
  – Download latest jQuery.js and reference in the project
    `<script type="text/javascript" src="jquery.js" /></script>`

• You may also reference to a library located on the web, e.g. https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js
  – For more hosting: http://docs.jquery.com/Downloading_jQuery#CDN_Hosted_jQuery

• Production vs. debug version
jQuery Basics

• $ - an alias for the jQuery "class"
  – $.get( "test.php" ); is equivalent to jQuery.get( "test.php" );

• $( ) - returns a collection of matched elements either found in the DOM based on passed argument(s) or created by passing an HTML string
  – $( "h1" ).remove();

• Hello world:
  $(document).ready(function() {
    alert(‘hello world’);
  });
  – Called after DOM is created and before images are loaded
Selectors

- jQuery selectors are used to "find" (or select) HTML elements based on their id, classes, types, attributes, values of attributes and more

- It's based on the existing CSS Selectors, and in addition, it has some own custom selectors

- Useful links:
  - jQuery Selectors Reference
  - jQuery Selector Tester
Selectors

- `<name>` – by element name - `$(‘div’)`
- `<id>` - by identifier - `$('#jokesDiv')`
- `.<class>` - by class name - `‘.myClass’`
- `<name> <descendant>` – descendant of and element name - `$('div p')`
- `$(‘div > li’)` - selects all direct descendant `li` of the `div` elements
- More:
  - `$('div#jokesDiv’), $('#table tr’)`
• Until now:
  `window.onload = function(){ alert("welcome"); }`
  – The code doesn't run until all images are finished downloading, including banner ads

• With jQuery:
  ```javascript
  $(document).ready(function(){alert("welcome"); });
  ```
  – Here the code runs as soon as the document is ready to be manipulated
Events

• Adding events to elements:

```javascript
$("a").click(function() {
    alert("Hello world!");
});

$("#p1").mouseenter(function() {
    alert("You entered p1!");
});
```

• As opposed to:

```javascript
var element=document.getElementById("p1");
element.addEventListener("mouseenter", function(){
    alert("You entered p1!");
});
```
The on() method attaches one or more event handlers for the selected elements.

```javascript
$\("p\"\).on({
    mouseenter: function() {
        $(this).css("background-color", "lightgray");
    },
    mouseleave: function() {
        $(this).css("background-color", "lightblue");
    },
    click: function() {
        $(this).css("background-color", "yellow");
    }
});
```

- Complete list: [http://api.jquery.com/category/events/](http://api.jquery.com/category/events/)
• Hiding/showing matched elements:
  – $( "#target" ).hide();
  – $( "#target" ).show();

• Getting the combined text contents:
  $( "#target" ).text();

• Setting content:
  – $( "#target" ).html('<a href="example.html">Link</a>');
  – $( "#target" ).text('<a href="example.html">Link</a>');

• Adding/Removing classes:
  $( "#target" ).addClass( "myClass" );
  $( "#target" ).removeClass( "myClass" );

• Creates element and inserts it into the document
  $( '<i>hello</i>' ).insertAfter( 'div.chapter p' );
## Overview

1. **AJAX - introduction**
2. **Asynchronous AJAX**
3. **jQuery**
4. **Ajax with jQuery**

![HTTP://WWW Icon](image)
jQuery and AJAX

- jQuery provides several methods for AJAX functionality

- Without jQuery, AJAX coding can be tricky
  - Different browsers have different syntax for AJAX implementation
  - We will have to write extra code to test for different browsers
  - In jQuery this has already been taken care of
Basic Operations

- **jQuery.load()** - Load data from the server and place the returned HTML into the matched element
  - $( "#result" ).load( "test.html" );
  - $( "#result" ).load( "ajax/test.html", function() {
      alert( "Load was performed." );
    });
  - $( "#result" ).load( "ajax/test.html #container" );

- **jQuery.get()** - Load data from the server using a HTTP GET request
  - $.get( "test.html", function( data ) {
      $( "#result" ).html( data );
      alert( "Load was performed." );
    });
  - $.get( "test.php", { name: "John", time: "2pm" } );
Basic Operations

- jQuery.**post()** - Load data from the server using a HTTP POST request
  - ```
      $.post( "test.html", function( data ) {
        $( ".result" ).html( data );
        alert( "Load was performed." );
      });
      ```
  - ```
      $.post( "test.php", { name: "John", time: "2pm" } );
      ```
Basic Operations

- jQuery.ajax() - Perform an asynchronous HTTP (Ajax) request
- jQuery.get() and jQuery.post() are shorthand Ajax functions, which are equivalent to:

```
$.ajax({
    url: url,
    data: data,
    success: success,
    dataType: dataType
});
```

```
$.ajax({
    type: "POST",
    url: url,
    data: data,
    success: success,
    dataType: dataType
});
```
References

• http://www.w3schools.com/ajax/default.asp

• http://www.w3schools.com/jquery/default.asp

• https://jquery.com/

• http://firequery.binaryage.com/
  – jQuerify: enables to inject jQuery into any web page