Ajax & jQuery

Internet Engineering

Fall 2016

Bahador Bakhshi

CE & IT Department, Amirkabir University of Technology

Questions

- Q5) How to update a portion of web page?
  - Check new mails?
- Q5.1) Is it possible?
- Q5.2) Should we wait for server response?
- Q5.3) What does server return back?

Outline

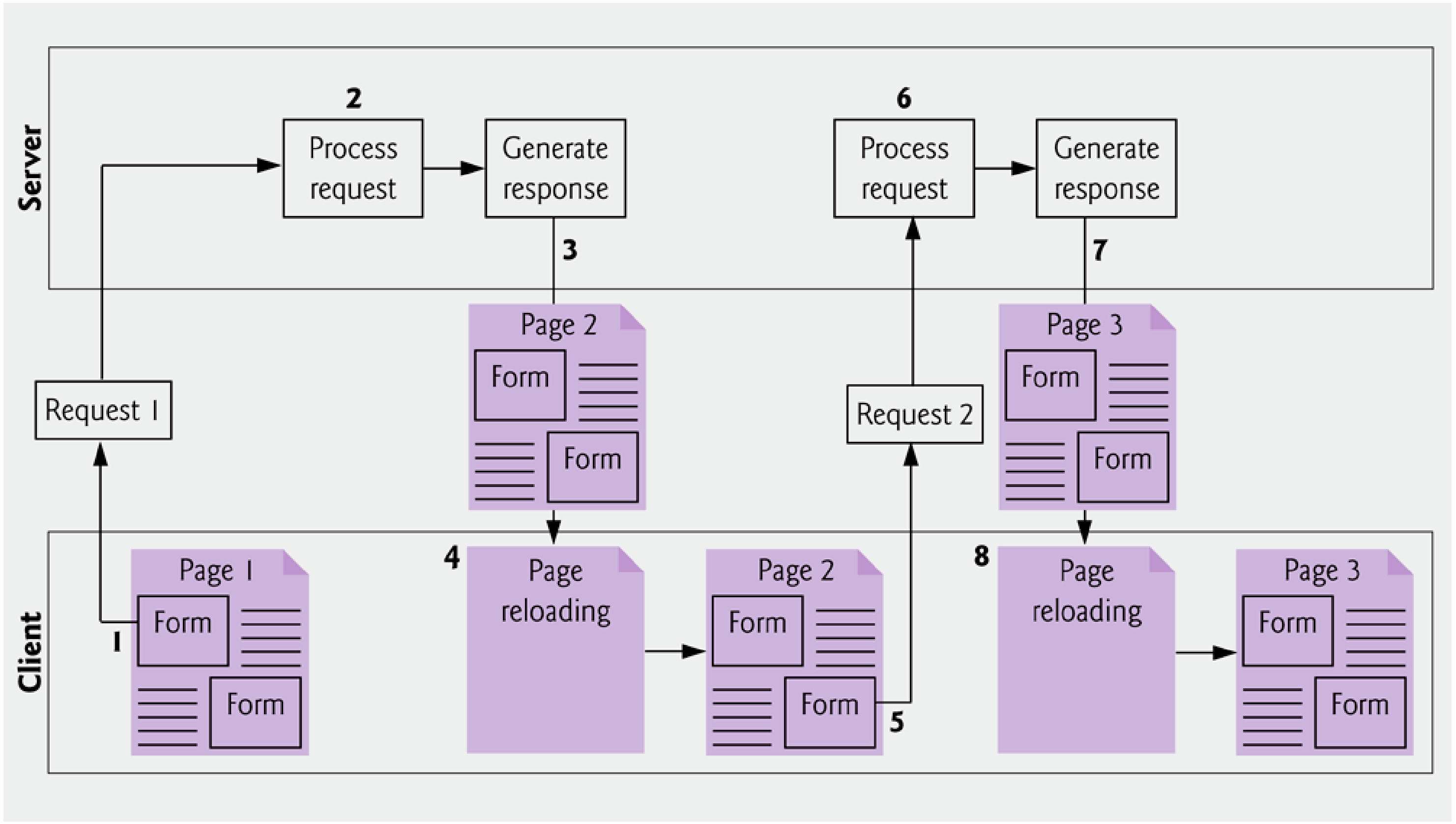
- Ajax
  - Introduction
  - Implementation
  - More details
  - Examples
- jQuery
  - Selection
  - Action
  - Ajax
  - Examples

Outline

- Ajax
  - Introduction
  - Implementation
  - More details
  - Examples
- jQuery
  - Selection
  - Action
  - Ajax
  - Examples

Introduction

- Traditional web applications

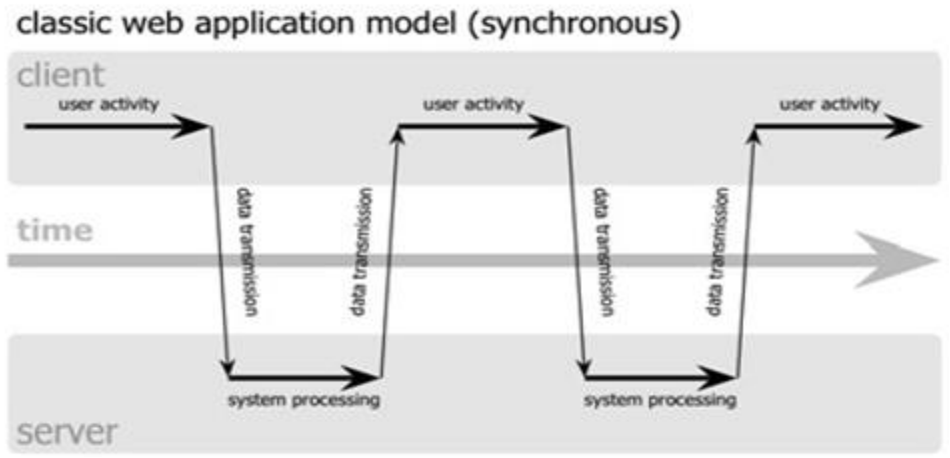


Introduction (cont'd)

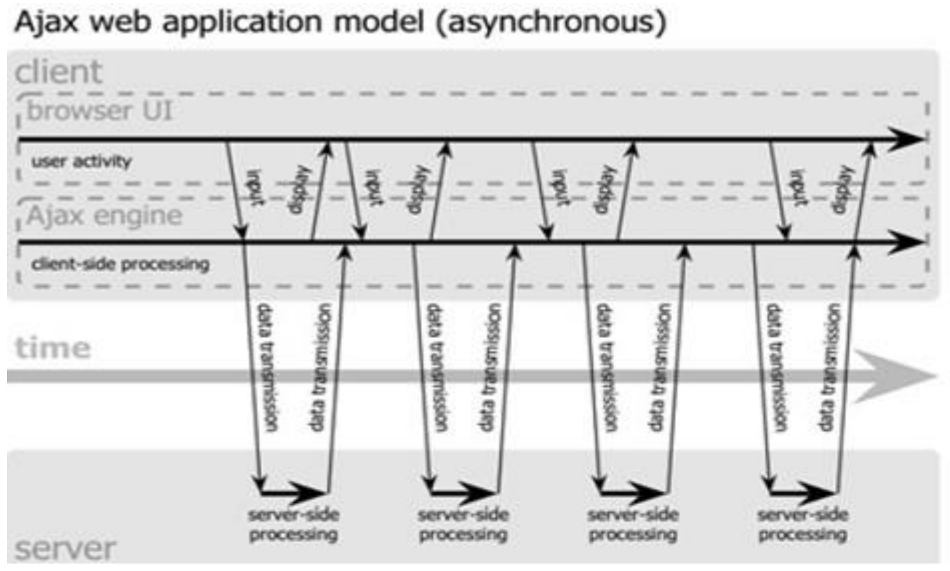
- Traditional web application is synchronous
  - User (request) & Server (response) are synchronized
    - User is filling forms → Server in idle mode
    - Server is processing → User is waiting
- Whole page must be reload to update *a section* of page
  - Check new mail in webmail → refresh the page!
  - Long response time & More BW overhead
- Typically user is involved in page dynamics!
  - No automatic update (because of page reload)!

Synchronous vs. Asynchronous

- Synchronous whole page update that interrupts user operation



- (Automated) Asynchronous update of a portion of page without interrupting user
  - E.g. updating list of emails while reading/composing other emails



Asynchronous Implementation

- How to implement the asynchronous method?
- What are required to implement it?
- 1) Send request to server from *inside* a web page
  - Links or forms *do not* work
    - Browser sends request but it reloads whole page!
- 2) Process server's responses
  - Typically the response is not HTML, it is data
- 3) Update part of page using the processed data
  - We already know it, access DOM using JavaScript

Asynchronous Implementation: Ajax

- Ajax: Asynchronous JavaScript And XML
- Concept is new
  - Be able to send asynch. request from web pages
  - To build Internet applications with much more appealing user interfaces
- But the technology is *not* new!
  - A mix of well-known programming techniques
  - Is based on JavaScript & HTTP requests
    - Get data by HTTP (which contains XML)
    - Update page without reloading by JavaScript

Ajax

- Ajax: Asynchronous
- User has not to wait for response from server
  - We can send request, continue other jobs, and process the response when is ready
- Server requests are not necessarily synchronized with user actions
  - Ajax application may already have asked of the server, and received, the data required by the user
    - Periodic tasks (e.g., Automated "check new emails")
- Ajax can be synchronous!!!
  - However, typically is asynchronous

Ajax (cont’d)

- Ajax: JavaScript
- Ajax is implemented by JavaScript
- JavaScript functions using a special object
    - Generate HTTP request to server
    - Get response from server
    - Process the response & update the page
        - Using DOM
- JavaScript can be replaced by other client-side scripting languages!

Ajax (cont’d)

- Ajax: XML
- Initial idea/design: Ajax is used to update page using data formatted as XML
- Response from server in XML format
- XML is not the mandatory response format
  - Server can send back any file format
    - Text, HTML, Image, …
  - JavaScript must be aware of the data type

Ajax Operation

- An event occurs in web page, e.g.,
  - User clicks on a button, fills a form, …
  - Automated/Periodic task just started
- JavaScript event handler creates & sends an HTTP request to the server
- The server responds with a *small amount* of data, rather than a complete web page
- JavaScript uses this data to modify the page
- This is faster because less data is transmitted and because the browser has less work to do

Ajax Applications

- Everywhere we need dynamic content from server in a portion of a web page
- Google Suggest
- Web mails (Gmail)
- Google Docs
- RSS Readers
- Rich Internet Application (RIA)
- …

Outline

- Ajax
  - Introduction
  - Implementation
  - More details
  - Examples
- jQuery
  - Selection
  - Action
  - Ajax
  - Examples

`XMLHttpRequest`

- Ajax is implemented by the **`XMLHttpRequest`** object
  - Allows JavaScript to formulate HTTP requests and submit them to the server
  - Provides a mechanism to get response and some facilities to process it
- Requests can synch. or asynch. and any type of document can be requested

`XMLHttpRequest`: Methods

- `open('method','URL','isAsync')`
  - method: specifies the HTTP method
    - E.g., GET, POST, …
  - URL: target URL, where the request is handled
  - isAsyc:
    - 'true': asynchronous operation
    - 'false': synchronous operation
- `send(content)`
  - Sends the request, optionally with POST data

`XMLHttpRequest`: Operation Mode

- **`XMLHttpRequest`** supports both synchronous and synchronous operation modes
  - isAsyn: true / false?
- In synchronous mode
  - The **`send()`** method is blocking
  - Does not return until the request is sent and a response is received
- In asynchronous mode
  - The **`send()`** method is not blocking
  - Just sends the request and returns

`XMLHttpRequest`: Methods

- `setRequestHeader('x','y')`
  - Sets a parameter and value pair x=y and assigns it to the header to be sent with the request
- `abort()`
  - Stops the current request
- `getAllResponseHeaders()`
  - Returns all headers as a string
- `getResponseHeader(x)`
  - Returns the value of header x as a string

`XMLHttpRequest`: Properties

- `status`
  - HTTP status code returned by server
- `statusText`
  - HTTP reason phrase returned by server
- `responseText`
  - Data returned by the server in text string form

```
xmlhttp = new XMLHttpRequest();
…
var doc  = xmlhttp.responseText;
```

`XMLHttpRequest`: Properties

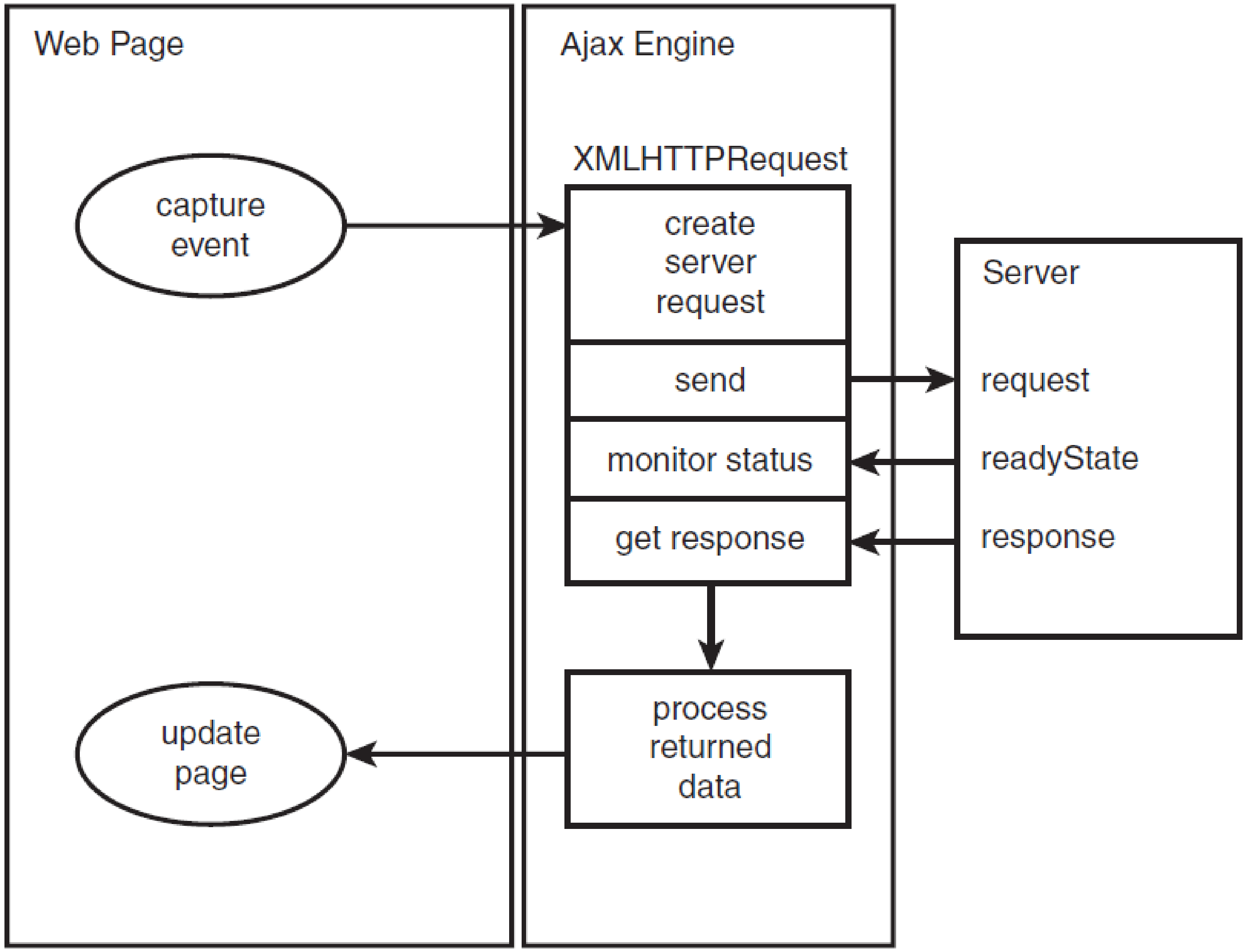
- `responseXML` returns the response as XML
  - Can be treated and parsed using the DOM
  - Content-Type of response is important
    - Content-Type="text/xml"

```
var xmlDoc =
  xmlhttp.responseXML.documentElement;
var value = xmlDoc.getElementsByTagName
("tagname")[0].childNodes[0].nodeValue;
```

`XMLHttpRequest`: Properties

- **`readyState`**
  - Integer reporting the status of the request:
  - 0 = The request is not initialized, before **`open`**
  - 1 = The request has been set up, before **`send`**
  - 2 = The request has been sent, **`send`** called
  - 3 = The request is in process, request is sent
  - 4 = The request is complete, response is ready
- **`Onreadystatechange`**
  - The event handler will be called when the object's **`readyState`** property changes

Overall Operation View



Synchronous Mode Code Skeleton

```
function synchronousAjax(){
   xmlhttp = new XMLHttpRequest();
   xmlhttp.open("GET","URL",false);
   xmlhttp.send(null);
   if(xmlhttp.status == 200){
        var response = xmlhttp.responseText;
        ...
   }
   else{
     window.alert("Error: "+ xmlhttp.statusText);
   }
}
```

Asynchronous Mode Code Skeleton

```
function asynchronousAjax(){
   var xmlhttp=new XMLHttpRequest();
   xmlhttp.onreadystatechange = process;
   xmlhttp.open("GET","URL",true);
   xmlhttp.send(null);
}
function process(){
   if(this.readyState == 4){
       if(this.status == 200){
              var response = this.responseText;
              ...
       }
       else{
              window.alert("Error: "+ this.statusText);
       }
   }
}
```

Example 1: Load Static File

```
<div id="test">
<h2>Click to let Ajax change this text</h2>
</div>
<button type="button"
  onclick="loadTextDocSynch('test1.txt')">
  Click Me(test1.txt)</button>
<button type="button"
  onclick="loadTextDocAsynch('test2.txt')">
  Click Me(test2.txt)</button>
<button type="button"
  onclick="LoadXMLDocAsynch('test3.xml')">
  Click Me(test3.xml) </button>
```

Example 1: Load Static File

```
function loadTextDocSynch(url){
  xmlhttp=new XMLHttpRequest();
  xmlhttp.open("GET",url,false);
  xmlhttp.send(null);
  if(xmlhttp.status == 200){
     document.getElementById('test').
  innerHTML=xmlhttp.responseText;
  }
  else{
     window.alert("Error "+
  xmlhttp.statusText);
  }
}
```

Example 1: Load Static File

```
function loadTextDocAsynch(url){
   var xmlhttp=new XMLHttpRequest();
   xmlhttp.onreadystatechange = process;
   xmlhttp.open("GET",url,true);
   xmlhttp.send(null);
}

function process(){
   if(this.readyState == 4){
       if(this.status == 200){
             document.getElementById('test').
             innerHTML=this.responseText;
       }
       else{window.alert("Error "+ xmlhttp.statusText); }
   }
}
```

Example 1: Load Static File

```
function LoadXMLDocAsynch(url){
   var xmlhttp=new XMLHttpRequest();
   xmlhttp.onreadystatechange = processXML;
   xmlhttp.open("GET",url,true);
   xmlhttp.send(null);
}
function getNodeValue(doc, name){
   return (doc.getElementsByTagName(name))[0].childNodes[0].nodeValue;
}
function processXML(){
   if(this.readyState == 4){
        if(this.status == 200){
                var xmlDoc = this.responseXML.documentElement;
                var res = "Name: "+ getNodeValue(xmlDoc, "name") +"<br />";
                res += "Model: "+ getNodeValue(xmlDoc, "model") +"<br />";
                res += "OS: "+ getNodeValue(xmlDoc, "OS") + " - "+
   getNodeValue(xmlDoc, "version");
                document.getElementById("test").innerHTML = res;
        }
        else{ window.alert("Error "+ xmlhttp.statusText); }
   }
}
```

Outline

- Ajax
  - Introduction
  - Implementation
  - More details
  - Examples
- jQuery
  - Selection
  - Action
  - Ajax
  - Examples

More Details: Sending Data

- Since Ajax uses HTTP requests, it can send data
  - Query part of the URL
  - Body of POST
    - The content is passed as the argument to **`send`**
- Encoding is important in both GET and POST
  - E.g. some characters are not legal in URL: URL encoding
  - The **`escape`** method does these replacements

```
xmlhttp.setRequestHeader('Content-Type', 'application/x-
   www-form-urlencoded');
xmlhttp.send("var1=" + escape(value1) + "&var2=" +
   escape(value2));
```

More Details: Other HTTP Methods

- In addition to GET and POST, other HTTP methods can also be used
- For example to analyze HTTP headers
  - Send a “HEAD” request

```
xmlhttp.open("HEAD","URL",true);
```

  - In the response, analyze the HTTP headers

```
getAllResponseHeaders()
getResponseHeader(x)
```

More Details: Concurrency

- We create a request object, and append our request information to it
- When the server responds, its result is also in the request object
- **Question:** What happens if, before we get a response, we use the request object to send off another request?
- **Answer:** We have overwritten the request object, so the response to the original request is lost
- **Solution:** We will need to create and use more than one request object

More Details: Avoid HTTP Caching

- We send a request using GET, and it works
- We want to get a new value, send the same request again
- Nothing happens! Why not?
  - **Answer:** The browser/cache server has cached our URL; it sees that we use it again *without change*, and gives us the cached response
  - **Wrong solution:** Turn off browser caching
  - **Correct solution:**
    - Change the URL in some unimportant way; a commonly used trick: adding of a parameter with a random and meaningless value to the request data
      - `url = url + "?dummy=" + (new Date()).getTime();`
      - `url = url + "?dummy=" + (new Math()).random();`
      - The server is free to ignore this parameter
    - Control caching
      - `setRequestHeader("Cache-Control", "no-cache");`

More Details: Security

- URL in `open()` can be a relative path or a complete URL
  - For security reason, browsers only allow to request URL in the same domain of the page
  - To access other sites, server side proxy is needed
- Method `open()` may also take an additional 4th and 5th parameters
  - `userid` and `password`
  - The two parameters are used to bypass HTTP authentication

Outline

- Ajax
  - Introduction
  - Implementation
  - Remarks
  - Examples
- jQuery
  - Selection
  - Action
  - Ajax
  - Examples

Example 2: Live Suggestion: Client

```
function showHint(str){
  if(str.length==0){
    document.getElementById("txtHint").innerHTML=""; return;
  }
  xmlhttp=new XMLHttpRequest();
  xmlhttp.onreadystatechange=function(){
    if (xmlhttp.readyState ==4 && xmlhttp.status==200)
      document.getElementById("txtHint").innerHTML =  xmlhttp.responseText;
  }
  xmlhttp.open("POST","gethint.php",true);
  xmlhttp.setRequestHeader('Content-Type', 'application/x-www-form-
   urlencoded');
  xmlhttp.send("query="+escape(str));
}
============================================================
<form>
First name:
    <input type="text" onkeyup="showHint(this.value)" size="20" />
</form>
<p>Suggestions: <span id="txtHint"></span></p>
```

Example 2: Live Suggestion: Server

```php
<?php
$a[]="Ahmad";
...
$a[]="Sajjad";
$q=$_POST["query"];
if (strlen($q) > 0){
  $hint="";
  for($i=0; $i < count($a); $i++){
    if (strtolower($q)==strtolower(substr($a[$i],0,strlen($q)))){
      if($hint==""){ $hint=$a[$i];}
      else{ $hint=$hint." , ".$a[$i];}
    }
  }
}
if ($hint == ""){ $response="no suggestion"; }
else { $response=$hint; }
echo $response;
?>
```

Example 3: Run Remote JavaScript

- Two text files
- msg1.js

```
window.alert("Hi, I am a window.alert
Message");
```

- msg2.js

```
var newp = document.createElement("p");
newp.innerHTML="I am a HTML message";
b = document.getElementsByTagName("body")[0];
b.appendChild(newp);
```

Example 3: Run Remote JavaScript

```
<script type="text/javascript">
function runJsAsynch(url){
   var xmlhttp=new XMLHttpRequest();
   xmlhttp.onreadystatechange = process;
   xmlhttp.open("GET",url,true);
   xmlhttp.send(null);
}
function process(){
   if(this.readyState == 4){
        if(this.status == 200) eval(this.responseText);
        else window.alert("Error "+ xmlhttp.statusText);
   }
}
</script>
```

Example 3: Run Remote JavaScript

```
<body>
  <button type="button"
  onclick="runJsAsynch('msg1.js')">Alert
  Message</button>
  <button type="button"
  onclick="runJsAsynch('msg2.js')">HTML
  Message</button>
</body>
```

Example 4: XSLT Transform

```
<body>
   Select Course:
   <select name="course">
       <option value="IE">Internet Engineering</option>
       <option value="NM">Network Management</option>
       <option value="C">C Programming</option>
   </select>
   <input type="button" onclick="displayResult()"
   value="Result" />
   <div id="resutl" style="border-style:solid;
   width:50%;"><span id="dummy"/></div>
</body>
```

Example 4: XSLT Transform

```
function loadXMLDoc(dname){
   xhttp = new XMLHttpRequest();
   xhttp.open("GET",dname,false);
   xhttp.send("");
   return xhttp.responseXML;
}
function displayResult(){
       name=document.getElementsByName("course")[0].value;
       xml=loadXMLDoc(name+".xml");
       xsl=loadXMLDoc("course.xslt");
       xsltProcessor=new XSLTProcessor();
       xsltProcessor.importStylesheet(xsl);
       resultDocument =
   xsltProcessor.transformToFragment(xml,document);
       resultdiv=document.getElementById("resutl");
       resultdiv.replaceChild(resultDocument,
                              resultdiv.children[0]);
}
```

Outline

- Ajax
  - Introduction
  - Implementation
  - Remarks
  - Examples
- jQuery
  - Selection
  - Action
  - Ajax
  - Examples

Introduction

- jQuery is a JavaScript Library that simplifies JavaScript programming (*Write Less, Do More*)
  - HTML element selections
  - HTML element manipulation
  - CSS manipulation
  - Event handling
  - JavaScript effects and animations
  - HTML DOM traversal and modification
  - Ajax

How to Use jQuery

- Download the library from jQuery.com
  - A text file: jquery.js
- Include the library in the HTML

```
<head>
  <script type="text/javascript"
  src="jquery.js"></script>
</head>
```

Outline

- Ajax
  - Introduction
  - Implementation
  - Remarks
  - Examples
- jQuery
  - Selection
  - Action
  - Ajax
  - Examples

jQuery Syntax

- The jQuery syntax is made of selecting HTML elements and performing some actions on the element(s)
- Basic syntax is: $(selector).action()
  - A dollar sign to define jQuery
  - A (selector) to "query (or find)" HTML elements
  - A jQuery action() to be performed on the element(s)
    - So many pre-defined actions!
  - Examples:
    - `$("p").hide()` - hides *all* paragraphs
    - `$("p.test").hide()` - hides *all* paragraphs with class="test"

jQuery Selectors

- `$(this)`: the current selected element
- jQuery Element Selectors
  - Similar to *CSS* selectors to select HTML elements
    - `$("p")` → all <p> elements
    - `$("p.intro")` → all <p> elements with class="intro"
    - `$("p#demo")` → the <p> elements with id="demo"
- jQuery Attribute Selectors
  - Similar to *XPath* expressions to select elements with given attributes
    - `$("[href]")` → all elements with an href attribute
    - `$("[href='#']")` → all elements with an href value equal to "#"
    - `$("[href!='#']")` → all elements with an href attribute NOT equal to "#"

jQuery Selectors

Selector	Example
`*`	$("*")
`#id , .class, element`	$("#lastname"), $(".intro"), $("p"), $(".intro.demo")
`:first, :last, :even, :odd`	$("p:first"), $("p:last"), $("tr:even")
`[attribute], [attribute=value], [attribute!=value]`	$("[href]"), $("[href='default.htm']"), $("[href!='default.htm']")
`:input, :text, :password, :radio, :checkbox, :file`	$(":input"), $(":text"), $(":password")

Outline

jQuery HTML Manipulation

- $(selector).`html ()`
  - Returns HTML content of the selected item(s)
- $(selector).`html`(content)
  - Changes HTML content of the selected item(s)
- $(selector).`append`(content)
  - Appends content to the inside of matching HTML elements
  - $(selector).`prepend`(content)
    - Prepends content to the inside of matching HTML elements
- $(selector).`after`(content/HTML code)
  - Inserts content/HTML code after all matching elements
- $(HTML tag)
  - Generate on the fly DOM elements
  - $(HTML code).`appendTo`(selector)
    - Appends the new element as a child to all selected nodes

jQuery HTML Manipulation

Method	**Description**
`html()`	Sets or returns the content of selected elements
`val()`	Returns value of input
`addClass()`	Adds one or more classes to selected elements
`hasClass()`	Checks if any of the selected elements have a specified class
`attr()`	Sets or returns an attribute and value of selected elements
`after()`	Inserts content after selected elements
`append()`	Inserts content at the end of (but still inside) selected elements
`before()`	Inserts content before selected elements
`prepend()`	Inserts content at the beginning of (but still inside) selected elements
`empty()`	Removes all child elements and content from selected elements
`remove()`	Removes selected elements
`removeAttr()`	Removes an attribute from selected elements
`removeClass()`	Removes one or more classes (for CSS) from selected elements

jQuery CSS Manipulation

- $(selector).`css()`
- The `css()` method has three different syntaxes, to perform different tasks
  - `css`(name)
    - Return CSS property value
  - `css`(name,value)
    - Set CSS property and value
  - `css`({property1: value1; property2: value2;...})
    - Set multiple CSS properties and values

jQuery Events

- $(selector).event(a function name)
- $(selector).event(function() {..some code... })

Event Method	Description
$(document).**ready**(function)	Binds a function to the ready event of a document
$(*selector*).**click**(function)	Triggers, or binds a function to the click event of selected elements
$(*selector*).**dblclick**(function)	Triggers, or binds a function to the double click event of selected elements
$(*selector*).**focus**(function)	Triggers, or binds a function to the focus event of selected elements
$(*selector*).**mouseover**(function)	Triggers, or binds a function to the mouseover event of selected elements

jQuery Effects

- jQuery has a few built in effects
  - Fading, hiding, …
- $(selector).effect(delay, callback)
  - Delay: optional, millisecond
  - Callback: optional, runs after effect completes
- Popular effects
  - **`hide(), show(), fadein(), fadeout(), slideUp(), slideDown(), slideToggle(), animate(), ...`**

jQuery Effects

Function	Description
$(selector).**hide**()	Hide selected elements
$(selector).**show**()	Show selected elements
$(selector).**toggle**()	Toggle (between hide and show) selected elements
$(selector).**slideDown**()	Slide-down (show) selected elements
$(selector).**slideUp**()	Slide-up (hide) selected elements
$(selector).**slideToggle**()	Toggle slide-up and slide-down of selected elements
$(selector).**fadeIn**()	Fade in selected elements
$(selector).**fadeOut**()	Fade out selected elements
$(selector).**fadeTo**()	Fade out selected elements to a given opacity

jQuery & DOM

- $(selector).`parent()`
  - The parent of the selected element
- $(selector).`children`(filter)
  - Array of the direct children of selected element that matched to the filter
- $(selector).`find`(filter)
  - Array of the descendant s of selected element that matched to the filter
- $(selector).`each(`function(){..}`)`
  - Loops over the array of selected elements and runs the custom function

Outline

- Ajax
  - Introduction
  - Implementation
  - Remarks
  - Examples
- jQuery
  - Selection
  - Action
  - Ajax
  - Examples

Example

- Course homepage using jQuery
  - Sliding elements
  - Automatic row highlighting
  - Automatic new items counter

Outline

- Ajax
  - Introduction
  - Implementation
  - Remarks
  - Examples
- jQuery
  - Selection
  - Action
  - Ajax
  - Examples

Answers

- Q5) Should we reload whole page to update a part of it?
  - No! High performance (bandwidth, time) overhead
- Q5.1) If not, how?
  - Using Ajax get data from server
  - Update HTML though DOM
- Q5.2) Should we wait for server response?
  - Ajax supports both synchronous and asynchronous modes
- Q5.3) What does server return back?
  - Anything, by default text and XML is supported

What are the Next?!

- Ajax Libraries & Frameworks
  - Ajax.OOP: OOP-style programming for Ajax
  - Ample SDK: Ajax framework for Rich Internet application development
  - Bindows: Enterprise Ajax framework
- jQuery UI
  - collection of GUI widgets, animated visual effects, and themes implemented with jQuery & HTML & CSS
  - The second most popular JavaScript library on the Web

References

- Reading Assignment: Chapter 10 of “Programming the World Wide Web”
- Ajax Standard: https://xhr.spec.whatwg.org/
- Ryan Asleson, Nathaniel T. Schutta, “Foundations of Ajax”
- Phil Ballard, “Sams Teach Yourself Ajax in 10 Minutes”
- Bear Bibeault and Yehuda Katz, “jQuery in Action” (Farsi translation is also available)
- w3schools.com/ajax/default.asp
- w3schools.com/jquery/default.asp