

CMSC335

Web Application Development with JavaScript



JavaScript7

Department of Computer Science
University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

Announcements

- Acid3 Test
 - Checks Web Browser's compliance with web standards
 - <http://acid3.acidtests.org/>
- <http://caniuse.com/>
 - Provides up-to-date information on features you can use in different browsers tables for support of front-end web technologies

Traditional Server/Client Interaction

- Nothing happens until we submit data (e.g., using submit button)
- We must wait until the server request is processed
 - Cannot do anything with the page (page freezes)
- **Example:** AjaxExamples/Synchronous
 - directoryLookup.html, directory.php, processMemo.php
- A page must be completely loaded even if most of the content is identical to the previous page
 - Compare with a desktop application
- Can we do better? Can the page be updated without requiring a page load?
 - Ajax is the answer
- **Example:** AjaxExamples/TypicalASynchronous
 - directoryLookup.html, directory.php, processMemo.php

Ajax

- Ajax - Asynchronous JavaScript and XML
- Ajax - Generates an HTTP request for us
- Combination of technologies
- Adds a layer between the browser and the web server, handling server requests and processing the results
 - Layer Name → Ajax Framework/Ajax Engine
- The requests are not synchronized with user actions (e.g., clicking on links, buttons, etc.). User can continue interacting with the browser while request is being processed

Ajax

- In the traditional client/server model we submit server requests by clicking on a link or via submit (this generates the HTTP request for us)
 - We get a new web page as a result
- **XMLHttpRequest**
 - JavaScript object that will issue the HTTP request
 - No page load is generated as a result of the request
 - Can only issue request to URLs within the same domain
- There is nothing the server needs to do just because the request is associated with Ajax. The server is just receiving an HTTP request
- Ajax application just cares about receiving an HTTP response

Ajax

- When/how to get the results
 - We need to add code to detect when the request has been completed
 - We need to add code to process the results (e.g., update page components)
- Creating the object (JavaScript code)
`let requestObj = new XMLHttpRequest();`

XMLHttpRequest Properties

- **readyState** - Request's status
 - 0 - **uninitialized**, assumed when initial server request is submitted
 - 1 - **loading**, placing data in **XMLHttpRequest** object
 - 2 - **loaded**, loading completed
 - 3 - **interactive**, object interaction is possible
 - 4 - **completed**
- **onreadystatechange**
 - Event handler called when readyState property changes
- **responseText** - results in text form
- **responseXML** - results in XML
- **status** - HTTP status returned by the server
- **statusText** - HTTP reason phrase

XMLHttpRequest Functions

- **open** - Initializes the **XMLHttpRequest** object
 - open(httpMethod, targetURL, requestMode)
 - httpMethod → GET or POST
 - requestMode → true for asynchronous, false for synchronous
 - If **get** method is used targetURL must include any necessary parameters
- **Send** - sends the request
 - For **get** method request data is set to null
- **Example:** directoryLookup.html, directory.php, processMemo.php

Submitting a Synchronous Request

- We can also submit a synchronous request using Ajax
 - You want to get results from the server but cannot proceed without them
- **Example:** AjaxExamples/Synchronous
 - directoryLookup.html, directory.php, processMemo.php

Errors

- **Example:** Errors1.html
- Error types
 - Error - Base type for errors
 - URIError
 - TypeError
 - EvalError - Error while using eval()
 - RangeError
 - SyntaxError
 - ReferenceError
- **Example:** Errors2.html
- You can use throw to throw an error
- You can define your own errors
 - throw new InvalidDataError("Positive value expected");
 - throw "NOOOO!";
- **Example:** Errors3.html

FileReader API

- Using HTML5 FileReader API
- **Examples:** FileReading folder
 - **Example:** FileReadingText.html (use data.txt)
 - **Example:** FileReadingImage.html (use img1.jpg and img2.jpg)
 - **Example:** FileReadingJSON.html (use person.json)

Sound

- **Example:** Sound.html
- Source for the sound can be an online reference

Retrieving Form Data using JS

- **Example:** RetrievingFormDataJS
 - Check at home the provided options