ITMD 465/565
Rich Internet Applications

Lecture 2

Fall 2019 – August 28, 2019

Tonight's Agenda

- Very Quick HTML & CSS review
- Discuss browser developer tools
- JavaScript I

JavaScript I

We start with ES5 and then add a little ES6

JavaScript Introduction

- JavaScript is the behavioral layer of our web pages. HTML is structural, CSS is presentational
- Can access all the elements, attributes and text on a web page using the DOM (Document Object Model)
- Can test for browsers features and capabilities, progressive enhancement
- Modify elements and CSS properties to show, hide, and change element appearance
- Makes AJAX interactions possible
- Historically support between different browsers has sometimes been mixed.
  - Some browser implementations support some features and some use different names or syntax for a given feature.

JavaScript Introduction

- Not Related to Java Programming Language
- Originally named LiveScript and created by Brendan Eich at Netscape in 1995. Later renamed JavaScript for marketing reasons because of popularity of Java Language at the time.
- Standardized by ECMA technically ECMAScript
  - Latest widely supported version is ECMAScript 5 – JS 1.8.5
  - ES6 is the newer branch, use it now in most modern browsers or with a complier like https://babeljs.io
- Lightweight Object-oriented scripting language
  - Procedural, object-oriented (prototype-based), and functional style
- Dynamic Language
  - Doesn't need to be compiled to machine code
  - Loosely typed - Don't need to declare variable types
- Read and interpreted on the fly

JavaScript Introduction

- Mozilla JavaScript Guide
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript
- Wikipedia JavaScript Entry
  - http://en.wikipedia.org/wiki/JavaScript
  - https://en.wikipedia.org/wiki/ECMAScript
- Node.js for Command Line JavaScript Intro
  - http://javascript.cs.lmu.edu/notes/commandlinejs/
- JavaScript and Basic Programming Introduction Reading
  - http://eloquentjavascript.net/

JavaScript Introduction

- **Embedded** Scripts
  - Use script tags <script> JS Here </script>
- **External** Scripts
  - Use script tag with src attribute <script src="myscript.js"></script>
  - Script tag must be empty inside
  - Can be placed anywhere on the page, blocks when executing
  - Most commonly in head section or at the bottom of the body before the closing body tag
  - Type attribute was required in <= html4 but not html5
  - <script type="text/javascript"></script>
- There is an async attribute in html5, async="async" & defer="defer"
  - Async executes as page parsing. Defer executes script when page finishes parsing.
  - When either is not present (default), executes immediately then finishes parsing page
  - http://www.growingwiththeweb.com/2014/02/async-vs-defer-attributes.html

JavaScript Introduction

- JavaScript is **case-sensitive** “foo” not equal “Foo”
- Made up of statements which should end with a semicolon.
- Contains reserved words you can not use. Search for a list of JavaScript reserved words for details.
  - https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Reserved_Words
- Comments can be single or multi line
  - Single Line – two slashes // This is a comment
  - Multi Line – similar to css /* This is a comment */

JavaScript Language Features and Syntax

ES5 to start plus a little ES6

JavaScript Language

- Variables
- Statements
- Blocks
- Functions
- Operators
- Comparison
- Conditional Statements
- Looping
- Objects
- Events

JavaScript Variables

ES6 Introduces two new variable declarations

let
const

Introduces Block Scope

var vs let vs const article

- Variables hold values or objects
  - Declare with var keyword `var foo;`
  - Set value with single = sign `var foo = 5;`
  - Names are case sensitive
  - Names must begin with a letter or the underscore
  - Can be a set of very basic data types: numbers, strings, booleans, objects, functions, and undefined values
  - No special characters in name (! . , / \ + * =)
  - Has functional scope – not block scope (ES5 with var)
  - If a variable is declared in a function without var keyword it's global
- Array – grouping/list of objects
  - Arrays are defined with - `new Array()` or `[ ]`
  - Zero indexed so first element is - `arrayname[0]`

JavaScript Statements

- Statements are commands to the browser that are executed in order
  - Should end with a semicolon but not required. JS has ASI
  - ASI - https://stackoverflow.com/questions/2846283/what-are-the-rules-for-javascripts-automatic-semicolon-insertion-asi
  - May span multiple lines if written carefully
  - Multiple statements may be on the same line if separated by a semicolon.
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements

JavaScript Blocks

- Statements can be grouped together in blocks with the curly brackets { }
- Usually blocks are used when defining functions or using conditionals or loops
- **JavaScript does not use block scope** like most programming languages. **It has function scope.** This can change in ES6 but you need to understand what version you are running and how it will behave.
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/block

JavaScript Mathematical Operators

- Addition + (plus operator is also used to concatenate strings)
- Subtraction -
- Multiplication *
- Division /
- Modulus (division remainder) %
- Increment ++
- Decrement --
- Add to self and reassign +=
  - var car = 5; car += 2; car is now 7

JavaScript Functions

- Named blocks of code that can be called and executed by events or other code, may take parameters
- Functions are objects in JavaScript

```
function funcname (var1, var2, …) {
  code block (may make use of parameters)
}
```

- The return statement will stop executing the function and return the value

```
function addnum(n1, n2) {
  return n1 + n2;
}
```

JavaScript Function

- Can be created with Function Declarations or Function Expressions
- **Function Declarations**

```
function myFunction() {
  statements;
}
```

  - Can be defined after being used. Defined in initial parse phase.

- **Function Expression**

```
var myFunction = function(){
  statements;
};
```

  - Must be defined before being used. Little more clear the var myFunction holds a function. Defined during execution.

- We will discuss ES6 arrow functions soon.

JavaScript Functions

- JavaScript has many built in functions as part of the language and specification.
- The functions available will vary depending on the execution environment.
- Useful basic built in functions:
  - `alert()`
  - `confirm()`
  - `prompt()`
  - `console.log()`
  - `Number() vs parseInt() or parseFloat()`
  - `Math and Date objects`
  - `Many more`
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions

JavaScript Comparison Operators

- Comparisons are used to compare the value of two objects and return true or false
- Comparison Operators
  - == Is equal to
  - != Is not equal to
  - === Is identical to (equal to and same data type)
  - !== Is not identical to
  - > Is greater than
  - >= Is greater than or equal to
  - < Is less than
  - <= Is less than or equal to
- alert(5 > 1); // Will alert “true”

JavaScript Conditional Statements

- Conditional statements
  - if statements
  - else statements
  - else if statements

```
if ( condition ) {
    run this block
} else if (condition) {
    run this block
} else {
}
```

JavaScript Loops

- **for** – loops through a block a specific # of times
- **while** – loops through a block while condition true
- **do...while** – loops through block once then repeats as long as a condition is true
- **for...in** – loops through all properties of an object, be careful with this one can be error prone. Do not use to loop through an array.

- For Loop Syntax

```
for (initialize the variable; test the condition; alter the value;){
    code to loop here
}
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Loops_and_iteration

JavaScript Objects

- Objects - All items except core data types in JavaScript are objects including functions
- Objects are basically a custom data structure
- No class system in JavaScript like in other programming languages. Uses Prototypes instead. Has a system of Prototype Inheritance. Class keyword added in ES6.
- The **browser** is the **`window`** object the **html page** is the **`document`** object
- Objects are composed of properties and methods
  - Properties are basically variables
  - Methods are basically functions
- Access an objects property – **`obj.propertyName`**
  - or **`obj["propertyName"]`**
- Execute an object method – **`obj.methodName()`**

JavaScript Objects

- Created by a function with new keyword
  - `var obj = new Object();`
- Created with an object literal
  - `var obj = {};`
  - `var obj = { key: value, key2: value2 };`
    - Key needs to be a string with no spaces, can not start with number or special character
    - `var obj = { color: "red", quantity: 5, instock: true };`
- Access or set properties with dot notation
  - `obj.color = "blue";` sets color of obj to blue
  - `obj.quantity;` would be equal to 5
  - Can also set or execute methods this way
  - You can also access properties with the array like syntax of `obj["color"]`
    - Useful when you need the property value to come from another variable

JavaScript Objects

- JavaScript Object Literal format
- An object literal is a comma separated list of name value pairs wrapped in curly braces.

```
var myObject = {
    stringProp: 'some string',
    numProp: 2,
    booleanProp: false
};
```

- Value can be any JavaScript Datatype including a function or other object.

JavaScript DOM

- Document Object Model (DOM)
- Object representation of a HTML or XML Document
- All elements are represented by objects
- DOM is an API that can be used in many languages
- JavaScript uses DOM scripting to modify the elements on a page
- DOM is a collection of nodes nested in a tree structure
- Also provides standard methods to traverse the DOM, access elements and modify elements
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

JavaScript DOM

- To access the DOM elements use methods of the document object
- Most common by id
  - `var a = document.getElementById("elementid");`
- Can also access by class, tag, selector
  - By Class - https://developer.mozilla.org/en-US/docs/Web/API/Element/getElementsByClassName
  - By Tag - https://developer.mozilla.org/en-US/docs/Web/API/Element/getElementsByTagName
  - By Selector - https://developer.mozilla.org/en-US/docs/Web/API/Document_object_model/Locating_DOM_elements_using_selectors
- Use the `object.getAttribute("src");` method to get a attribute's value from an object. Also a `setAttribute` to set or change one.
- Set of methods to manipulate DOM objects.
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

JavaScript Event Handling

- Three Methods
- As attribute on HTML element
- As a method attached to a DOM object
- Using the add event handler method of a object

```
object.addEventListener("click", myFunction, false);
```

- https://developer.mozilla.org/en-US/docs/DOM/element.addEventListener

JavaScript

- JavaScript & DOM Reference
- https://developer.mozilla.org/en/docs/JavaScript
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
- http://reference.sitepoint.com/javascript/domcore
- https://developer.mozilla.org/en-US/docs/Web/API/element

Additional Learning Materials

- CodeAcademy – Free online interactive training
  - https://www.codecademy.com/
- Khan Academy – Free online interactive training
  - https://www.khanacademy.org/
- Coursera – Paid and Free
  - https://www.coursera.org/
- Git tutorials
  - https://git-scm.com/docs/gittutorial
  - https://try.github.io/levels/1/challenges/1
  - https://www.atlassian.com/git/tutorials
  - https://guides.github.com/

Assignments

Reading/Assignments

- Read Eloquent JavaScript Book to support and expand on what we discussed in class.
- http://eloquentjavascript.net/ or download pdf from website or blackboard
- Read through the end of chapter 3, and read chapter 13, before next class. I would suggest reading ahead further. Do exercises if you need more reinforcement of ideas.
- Complete Lab 1 before next class – Lab 1 **will not be accepted after class** starts next week (Sept 4 at 6:30pm)
- Read any Mozilla developer links from the slides if you feel you need further information on that topic.

- Get Node JS installed and do git tutorial if you don't know how to use it