Reminders

- Attendance is 15% of grade
- Quiz 4 Due Thursday
- Lab 1 Due Tuesday by midnight
- T.A. Hours
  - Monday and Wednesday 4-5 pm
- Zoom Classes Week 6, September 26 and 28th
  - We will be using the Collaborate Ultra feature on the blackboard.
  - For this week attendance will automatically be taken. You may only be a max of 10 mins late and you must be present for at least 50% for the attendance to count. The video for the class will be under the Collaborate Ultra tab in Blackboard.
- Lab 2 will be assigned by Friday night. It will be due on Tuesday, Oct. 3.
- **Special note for midterm and final exam**. Date given is the due date. Exams will be posted 4-5 days before that date.

ITMD 441/541

Web Application Foundations

Week 5

FALL 2023 – SEPTEMBER 18, 2023

This week's Agenda

- JavaScript Essentials
- Assignments

JavaScript Essentials

BEHAVIORS OF A WEB PAGE

JavaScript Introduction

- JavaScript is the behavioral layer of our web pages. HTML is structural, CSS is presentational
- Scripting language that enables dynamic content on the page
- 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 and background content updates
- 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
  - Almost universally supported legacy version is ECMAScript 5 – JS 1.8.5
  - ES6+ are the newer modern versions, use it now in most modern browsers or with a complier like babel
  - Newest version is 13 edition, June 2022 or ECMAScript 2022 language Specification
- Lightweight Object-oriented scripting language
  - Procedural, object-oriented (prototype-based), and functional style
  - Interpreted - Doesn't need to be compiled to machine code
  - Dynamic Loosely Typed Language - Don't need to declare variable types

JavaScript Introduction

- JavaScript is the language and basic built-in objects and functions you can use.
- First and mostly existed as a browser language but is now being used in many other environments like NodeJS
- The environment that the JavaScript is being executed in will provide some APIs
  - The browser engine gives you many such as DOM, Geolocation, Canvas, Audio, Video, and many more
- 3rd party APIs and code can be added to your pages to provide other capabilities like the Google Maps API for example. Let's try one.
  - https://www.w3schools.com/js/tryit.asp?filename=tryjs_api_geo_coordinates

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
- JavaScript and Basic Programming Introduction Reading. Read over chap 1-2:
  - http://eloquentjavascript.net/
- ECMAScript 13th edition – 2022 Specification
- ECMAScript 6th edition – 2015 Specification
- ECMAScript 5.1 edition – 2011 Specification

Adding JavaScript to a page

- **Embedded/Internal** 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 are async and defer attributes in html5, async="async" & defer="defer"
  - Async downloads as page is parsing then blocks to execute
  - Defer downloads as page is parsing then executes when page finishes parsing
  - When either is not present (default), executes immediately and blocks then finishes parsing page
  - http://www.growingwiththeweb.com/2014/02/async-vs-defer-attributes.html

JavaScript Language Features and Syntax

JavaScript Language

- Values and Variables
- Objects
- Statements
- Blocks
- Functions
- Operators
- Comparison
- Conditional Statements
- Looping

JavaScript Basics

- 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 Variables

- Variables hold values or objects
- Declared with `var` keyword (pre ES6)
- Set value with single = sign
- Names are case sensitive
- Names must begin with a letter or the underscore
- Can be a set of very basic data types: numbers, bigints, strings, booleans, objects, functions, symbols, undefined, and null 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

```
var foo;
var foo = 5;
```

JavaScript Variables

- ES6 + Introduces new keywords to declare variables and constants
- These new keywords introduced block scoping
- `let` keyword
  - let declares a block scoped variable
- `const` keyword
  - const declares a block scoped constant
  - Once a value is assigned it can not be changed
  - Object properties can be changed though, we will see an example later
- https://dev.to/sarah_chima/var-let-and-const--whats-the-difference-69e

JavaScript Values

- Consist of primitive values and objects

- undefined – used for unintentionally missing value
- null – used typically for intentionally missing values
- booleans – true/false, used for logical operations
- numbers – (25, 3.1415) all numbers but limited precision as you get further from 0 and used for math
- bigints – (uncommon, new) large integers for more precision and math on big numbers 232131231233123123n
- strings – text and immutable, all possible strings exist as values
- symbols - (uncommon)
- functions – used to refer to code and execute that code (functions are objects)
- objects – group related data and code
- typeof operator will tell you the type of value a variable points to
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures
- https://developer.mozilla.org/en-US/docs/Glossary/Primitive

JavaScript Array

- Array – grouping/collection of objects in a single variable name
- Arrays are objects
- Not Associative arrays so you must use integer indexes
- Arrays are defined with either the new array constructor or array literal
  - `new Array()` or `[ ]`
- Zero-indexed so first element is: `arrayname[0]`
- Each location in an array can hold any value
- Arrays have methods and properties to manipulate it, see reference
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

JavaScript Objects

- Object - All items except primitive 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 `obj.run()`
  - 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 is most common
- 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 Objects

- Object Reference Documentation
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object

- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects

Standard Objects

- JavaScript has many standard built-in objects
- Some of these objects provide methods to manipulate basic data like strings, numbers, dates, and more or act as wrappers around primitives
- Some important basic ones to study are Array, String, Number, Math, Date
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

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 with the new variable keywords, but you need to understand what version and syntax you're using and how it will behave.
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/block

JavaScript Basic Math Operators

- Addition + (plus operator is also used to concatenate strings)
- Subtraction -
- Multiplication *
- Division /
- Modulus (division remainder) %
- Exponent **
- Increment ++ and Decrement --
- Add to self and reassign +=
  - var car = 5; car += 2; car is now 7
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Math

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 Functions

- 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 clearer that the var myFunction holds a function. Defined during execution.

JavaScript Functions

- There is a third way to declare a function
- ES6 Introduced arrow function expressions
- Don't introduce their own binding to the **`this`** keyword
- If the arrow function is one line it will have an implicit return
- ```
  (a, b) => {
    return a + b;
  }
  ```
- ```
  (a, b) => a + b;
  ```
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

JavaScript Functions

- JavaScript has many built in functions and objects 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
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

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"
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators#comparison_operators

JavaScript Conditional Statements

- Basic conditional statement is the if/else/else if
- Used to branch code on conditions
- Else and else if are completely optional

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

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Control_flow_and_error_handling#conditional_statements

JavaScript Basic 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...of`** – loops over an iterable object
- 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

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Looping_code

Additional Resources

- MDN Web Technology Tutorials
- https://developer.mozilla.org/en-US/docs/Web/Tutorials

- MDN JavaScript Learning Guides
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript

- JavaScript Reference Documentation
- https://developer.mozilla.org/en-US/docs/Web/JavaScript

Assignments

Reading/Assignments

- Quiz 4 – Week 4 Content – Due end of day September 21
- Reading:
  - Read through MDN sections on learning JavaScript
    - https://developer.mozilla.org/en-US/docs/Learn/JavaScript
    - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
  - Start reading through Eloquent JavaScript
    - http://eloquentjavascript.net/
- Quiz 5 – Week 5 Content – Assigned this weekend. Due Thursday, September 28
- Lab 2 will be assigned by Friday night. Due on Tuesday, Oct. 3.