ITMD 441/541

Web Application Foundations

Week 11

FALL 2023 – NOVEMBER 6, 2023

Reminders

- Attendance is 15% of your grade and is being taken each week for in person students.
- Quiz Week 10 lecture. **Due: Thursday, November 9**
- Lab 4 will come soon
- **Final exam will be posted on Sunday, December 3 and due by Thursday, December 7 by midnight. (Final Exam week Dec. 4-9th)**
- **Very similar to the midterm except it will cover all class material and assignments. No project. 60-75 questions.**

Weekly's Agenda

- Additional JavaScript Objects and APIs
- NodeJS Introduction
- ES6 Modules

Location

- The Location object/interface represents the URL of the Window or Document object
- It provides properties and methods to access and change the individual parts of a URL in the browser's location bar
- `location.href` is a string version of the entire url and if replaced with a new value, the browser will navigate to the new location

- https://developer.mozilla.org/en-US/docs/Web/API/Location

History API

- The History API provides JS access to the session history in the browser
- It is exposed by the `window.history` object
- Provides methods and properties to allow you to navigate through the user's history and make changes to the user's history
- Move back and forward with the `back()` and `forward()` methods
- Go to a specific location with the `go()` method
- The `pushState()` and `replaceState()` methods can be used to add or modify a history state
- **READING MATERIAL**
- https://developer.mozilla.org/en-US/docs/Web/API/History_API
- https://developer.mozilla.org/en-US/docs/Web/API/History_API/Working_with_the_History_API

Date Object

- JavaScript Date object is the representation of one moment in time and is stored in a platform-independent format
- It contains a `Number` which is the milliseconds since January 1, 1970, UTC
- Provides numerous methods to access the individual date components in local time or UTC time
- See VsCode dateObject demo

- READING MATERIAL only no working demo
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

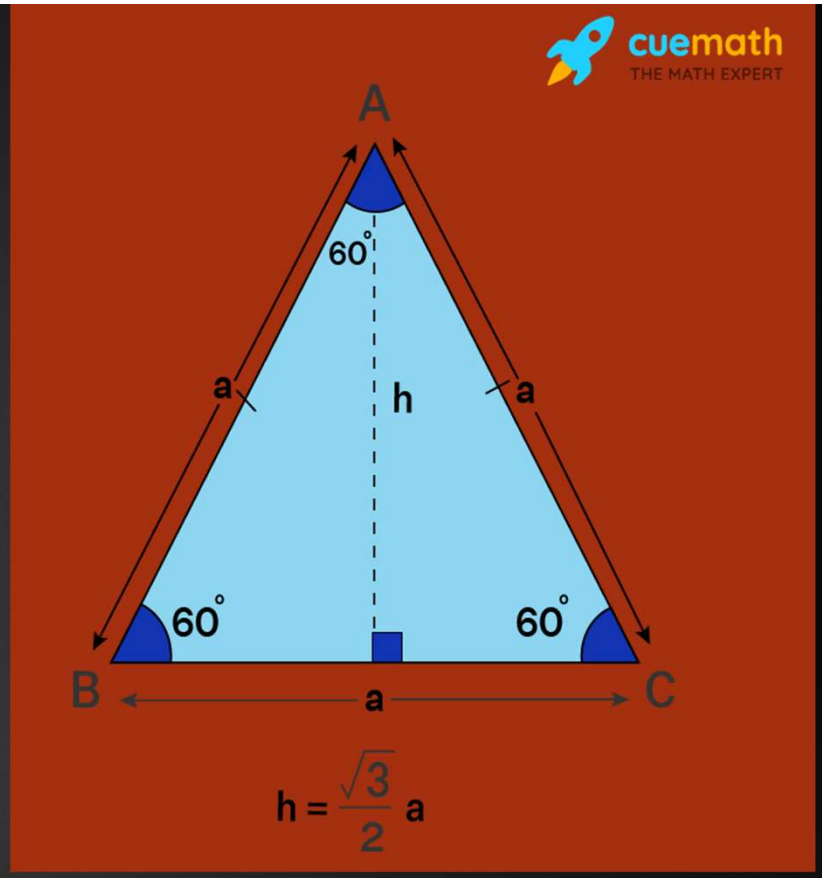
Math Object

- The `Math` object is a built-in object that provides many mathematical functions and constants
- It does not have a constructor. All properties and methods are called on the object not an instance of the object (static).
- Works with `Number` types but not `BigInt` types
- Some methods to take note of are `floor()`, `ceil()`, `random()`

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

Math Object

Here, the side of the given equilateral triangle is 6 cm.
a = 6 cm
h = (√3/2)a = (√3/2) × (6) = 3√3 cm



- Using trigonometric functions
- Tan(60) = h/3
- H = 3*tan(60)
- Calculate using VSCode and math objects. Go to VSCode Math Demo

Image Object

- The `Image` object is used to create a new instance of the `HTMLImageElement`
- Equivalent to using `document.createElement('img')`
- As soon as the src attribute is set the image will begin to download and you can listen for an onload event to do something
- We saw an example of using the Image object in the canvas examples

- https://developer.mozilla.org/en-US/docs/Web/API/HTMLImageElement/Image

Array Object

- `Array` is an object in JavaScript, not a primitive
- Allow for a collection of objects/values to be stored in a single variable and in a set order
- Resizable and can contain mixed data types
- Must be accessed using the non-negative integer indexes and the array is zero-indexed
- Length property will return the number of items in the array
- Built-in array copy operations make shallow copies
- Created with the `Array()` constructor or just `[]`
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

- `Use VSCode to demonstrate. Go to Array Demo.`

Array Object

- Provides many methods and properties to work with arrays
- Useful static methods include `from()`, `isArray()`, `of()`
- Can use the `at()` method to get an item at a given index (pos or neg)
- `forEach()` method will run a given function for each item in the array
- Multiple methods for searching through arrays
- `push()` and `pop()` to add an element to the end or remove and return the last element
- `splice()` for adding or removing elements from the array at given positions, works on existing array, vs `slice()`

Reminders

- Lab 4 assigned this weekend, Due Tues Nov 21
- Plan to send Zoom Link for class Nov 21 and provide video
- Quiz 9 posted and due end of day Thursday, Nov 9

Quick Quiz

1. MATH.SQRT(-0) = -0
   A. True
   B. False

2. MATH.SQRT(-25) = -5
   A. True
   B. False

Helpful Array Methods

- Here are a few array methods you should know. The important part here is they work in a functional way. **They do not modify the input but instead return a new array.**
- map()
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map
- filter()
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter
- reduce()
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce
- concat()
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/concat

Bind, Call, Apply

- All methods implemented on the global Object and used with functions
- `bind()` – creates and returns a new function with the value of this set to the argument https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/bind
- `call()` – way to call a function with a value of this given as an argument. **Also takes additional arguments individually.** https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/call
- `apply()` – way to call a function with a value of this given as an argument. **Also takes additional arguments as a single array.** https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/apply

NodeJS Introduction

Node JS

- Open-source JavaScript runtime environment built for back-end use instead of in-browser use.
- Built on the V8 JavaScript Engine from Chrome
- Can be used for CLI applications or server-side scripting and applications
- Event-driven asynchronous architecture
- Optimizes throughput and scalability
- Initially written by Ryan Dahl in 2009
- Governed by the OpenJS Foundataion after it was created by the merger of the Node.js Foundation and the JS Foundation
- https://en.wikipedia.org/wiki/Node.js
- https://nodejs.org/

Node Project Setup

- NPM – or "Node Package Manager" – is the default package manager for JavaScript's runtime Node.js.
- Go over basics of creating node-based projects
- **npm init** to initialize projects
- **npm install** or **npm i** to install local node modules to your project
- --save to save dependency to package.json - defaults to this without flag now in current versions
- --save-dev to save dependency as a development dependency
- npx will run local project binaries found in the node_modules folder
- There is an alternative package manager called yarn that many people use instead of npm. It was created by Facebook to enhance npm. Npm has now adopted many of the thing's yarn introduced. Fully compatible with npm.
- Has an example:
- https://www.w3schools.com/nodejs/nodejs_npm.asp

ES6 Modules

ES6 Modules

- ES6 Modules provide a way to split up your JavaScript code into separate modules and only import what is needed when you need it.
- NodeJS has a module system implemented with CommonJS.
- There have been other libraries and frameworks (CommonJS, AMD, UMD) that have enabled modules in JS but ES6 Modules is a native implementation.
- Most modern browsers support the ES6 syntax directly.
- ES6 modules are used heavily with frameworks like React and tools like webpack
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules

ES6 Modules

- What do we get?
- Modules can be more easily reusable. Code can be put in smaller files that encapsulate functionality and can be shared between different applications.
- All the code in a ES6 module is private by default and executes in strict mode. You must explicitly export what you want to use outside the module.
- Helps solve issues with namespace and naming conflicts.

ES6 Modules

- Creating Modules
- ES6 modules are stored in JS files. There is exactly one module per file and one file per module.
- You can have named exports and/or a single default export
- You can export things like variables, objects, classes, functions.
- Use the export keyword
- Named exports
  - `export function add(){}`
  - `export const PI = 3.1415;`
- Default exports – use export default, can be anonymous, only one per module
  - `export default function(){}`
  - export multiple named exports also as default
    - `export default {add, Car, PI};`

ES6 Modules

- Importing Modules
- Import statement used to access what export has made available in a module
- Imports must be in the top level of a module (first lines in file or script tag)
- Import and export can only be used in script tags that add the type="module" attribute. This goes for both external and embedded.
- ES Modules do not work with file:// protocol
- Automatically runs in strict mode
- Adding module code to an HTML page
- `<script type="module" src="./module.js"></script>`
- ```
  <script type="module">
  import statements
  Embedded JS Code
  </script>
  ```
  ```

ES6 Modules

- Import Statements
- Default import:
  - `import localName from 'src/my_lib';`
- Namespace import: imports the module as an object (with one property per named export).
  - `import * as localName from 'src/my_lib';`
- Named imports:
  - `import { name1, name2 } from 'src/my_lib';`
  - Renamed imports
  - `import { name1 as localName1, name2 } from 'src/my_lib';`
  - `import { default as localName } from 'src/my_lib';`
- Combining imports (default must come first):
  - `import theDefault, * as localName from 'src/my_lib';`
  - `import theDefault, { name1, name2 } from 'src/my_lib';`
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import
- https://exploringjs.com/es6/ch_modules.html#sec_importing-exporting-details
- **ES6 Modules Examples**

Examples

The "modules" directory contains a series of examples that explain how JavaScript modules are used. The subdirectories are as follows:

- basic-modules: Simple example that demonstrates module basics, including default exports (run the example live).
- renaming: Shows how exports can be renamed to avoid conflicts, using x as y syntax (run the example live).
- module-objects): Shows how to import an entire module as an object using import * as x from 'y.js' syntax (run the example live).
- classes: Provides an example of importing a class from a module (run the example live).
- module-aggregation: Shows how sub module features can be aggregated into a parent module using export { x } from 'y.js' syntax (run the example live).
- dynamic-module-imports: Demonstrates dynamic module loading using import().then() (run the example live).
- top-level-await: An example of using the await keyword within a module (run the example live).

ES6 Module Resources

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules
- https://exploringjs.com/es6/ch_modules.html#sec_modules-in-browsers
- Important information to read. Read over the 3 Scenarios.
- https://www.freecodecamp.org/news/how-to-use-es6-modules-and-why-theyre-important-a9b20b480773/
- https://www.sitepoint.com/understanding-es6-modules/
- https://www.bignerdranch.com/blog/default-exports-or-named-exports-why-not-both/

Assignments

Reading/Assignments

- Lab 4 assigned this weekend, **Due Tues Nov 28 (NEW DATE)**
- Plan to send Zoom Link for class Nov 21 and provide video
- Quiz 9 posted and due end of day Thursday, Nov 9
- Quiz 10 will come this weekend to cover tonight's materials
- Read MDN links in slides regarding JavaScript objects we talked about
- Read MDN about modules

- Read over "Basic Example Structure". Examples on GitHub
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules