Week 4 Day 2

Javascript Intro

Javascript

REVATURE

Client-Side scripting/programming language

- Runs in the browser
  - Used to create dynamic webpages
- High-leveled
- Interpreted
- Multi-paradigmed
- Dynamically typed
- Singled Threaded
- Follows ECMAScript specifications

Javascript: Using with HTML

REVATURE

HTML supports the use of JS with the <script> tag

- Internal Javascript is written in the HTML file inside of the <script> tag
- External Javascript is written in an external JS file and imported with the src attribute in the <script> tag
  - Best practice to load it at the bottom of the body

Javascript: Syntax and Variables

REVATURE

Syntax Rules

- Case Sensitive
- Semicolon opitional
- White space doesn’t matter
- Single and multi-line comments

Variable Rules

- Used to store values
- Declare with var, let and const keywords
- Must have a name
  - Name cannot be a keyword
  - Cannot start with special characters
- Variable literals are your primitives

Javascript: Datatypes

REVATURE

1. String
   - Text in single or double quotes
2. Number
   - Positive or Negative
   - Decimal or Integer
   - Nan or Infinity
3. Boolean
   - True or false
4. Null
   - Nothing
5. Undefined
   - Declared but not initialized
6. Object
   - Key value pairs
   - Key is a string, value is anything
   - Access properties via .notation, or [brackets]
7. Symbol
   - Used to create unique ids for objects or iterators

Type Coercion

- Converting a value from one datatype to another
- Explicit: var num = new Number(“3”)
- Implicit: var div = “3”/4

Javascript: Arrays

REVATURE

Object that stores a list of values

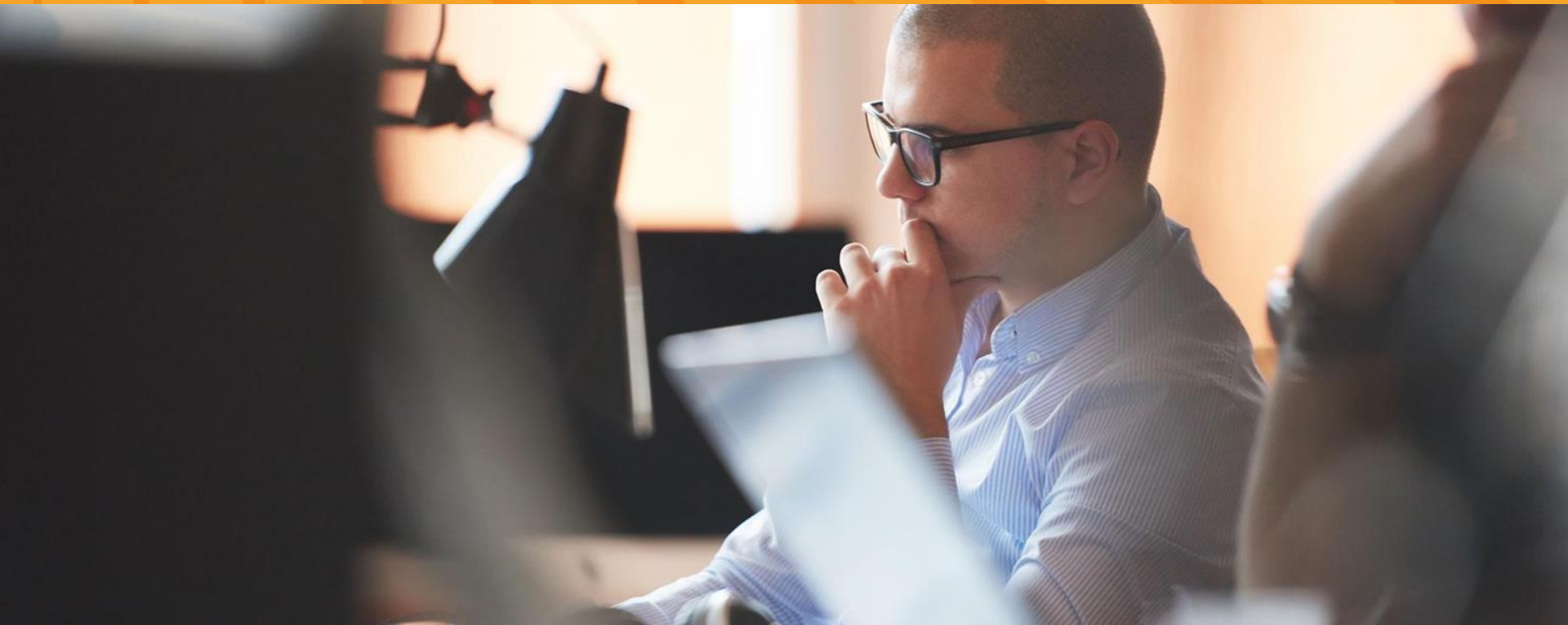
- Store any and all data types in a single array
- Index starting at 0
- Accessed via index inside of brackets
- Has a length property
- Dynamically sized

Useful built-in methods

- .push()
- .pop()
- .foreach()
- .filter()
- .map()
- .slice()

REVATURE

Javascript DEMO



Javascript: Operators and Control Flow

REVATURE

Operators perform some logic on single or multiple operands and produce a result

- Arithmetic:
  - -, +, *, /, %, ++, --
- Comparison:
  - ==, ===, !=, >, <, >=, <=
- Logical:
  - &&, ||, !
- Assignment:
  - =, +=, -=, *=, /=, %=
- Ternary Operator
  - Condition ? Value : Value

Control flow breaks the regular flow of our JS scripts

- if/else
- for loop
- for in loop
- for of loop
- while loop
- do while loop

Javascript: == vs ===

REVATURE

Both == and === are used to compare objects

- == compares two variables but does not take their types into account
- == can convert datatypes to make two variables “equal”
- === compares variables AND their datatypes
- === will only return true if the variables are strictly equal

Javascript: Truthy and Falsey

REVATURE

Any expression or value that results in the Boolean false is considered falsey

- Boolean false
- Empty string “”
- Undefined
- Null
- NaN
- 0

Everything else is truthy

Javascript: Functional Programming

REVATURE

- Functional Programming
  - The idea that programs can be broken down into callable expressions called functions
- Functions
  - Use the function keyword
  - Can use the return keyword to return a value
- Function Expression
  - Anonymous functions
  - No identifier or name, stored in a variable
- Immediately Invoked Function Expression (IIFE)
  - Anonymous function that instantly calls itself
- Callback Function
  - Function passed as a parameter to another function
  - Gets executed once the original function is finished
- Default Parameters
  - Allow us to set a default value for a parameter to a function in JS

REVATURE

Javascript with Functions DEMO

Javascript: Variable Scopes

REVATURE

Variable scope defines the lifetime and visibility of the variable

- Global Scope
  - Accessible everywhere in the application
- Local
  - Accessible in their location
  - Includes Function and Block
- Function Scope
  - Only accessible inside of the function they are defined in
- Block
  - Only accessible inside of the code block they are defined in
  - Possible due to the let and const keywords

Javascript: Hoisting

REVATURE

Mechanic where function and variable declarations are moved to the top of their scope

- Only the declarations are hoisted, not assignments
- Only variables declared with var are hoisted out of block scope

Javascript: ES6

REVATURE

Ecmascript 6 introduced many important features

- let and const keywords
  - let allows block scoped variable
  - const allows constant block scoped variables
- Arrow functions
  - Simplified way to write function expressions
  - Uses arrow notation let func = (args) => {expression}
- Template literals
  - Create multiline strings, and easily perform string interpolation
  - Uses backticks `` and allows for embedded expressions with the ${} notation

ES6: Spread and Rest Operator

REVATURE

- Spread Operator
  - Used to combine aways

```
const arr = [1, 2, 3];
const arr2 = [...arr]; // like arr.slice()

arr2.push(4);
//  arr2 becomes [1, 2, 3, 4]
//  arr remains unaffected
```

- Rest Operator
  - Acts like var args from Java

```
function f(a, b, ...theArgs) {
  // …
}
```

ES6: strict keyword

REVATURE

- use strict in Javascript disables the use of
  - Undefined variables
  - Any keyword as variable or function name
  - Some other niche javascript features

REVATURE

Javascript ES6 DEMO