AIM

CODE SCHOOL

AN ACCREDITED AIM INSTITUTE PROGRAM

Javascript Specialization Syllabus

Class Description: This 90-hour course will cover foundational knowledge of javascript, building on the material of the Foundations course, preparing students to become full-stack developers.

Class Goals: This JavaScript Course will cover the intricacies of JavaScript and the technologies that power a modern full-stack development workflow. This includes client and server side scripting, single web-page applications utilizing the MVN structure, Node package management, and JSON data storage. Students will finish the class having built a fully functional client-server application.

Class Objectives:

- Participants will come out of this module with foundational knowledge equivalent to that needed of a front end developer.
- Participants will build a foundation of knowledge on which to further grow programming language knowledge

Prerequisites: Students will need to have taken the Foundations of Web Development class before taking this course.

Needed Materials:

- Laptop with the following minimum specs:

    - 3 years old or newer and have 2 GB of RAM (8GB or more recommended) as well as 2.5 GB available hard-disk space for installation; additional free space required during installation (cannot install on a volume that uses a case-sensitive file system or on removable flash storage devices).
- Access to portal.aimcodeschool.org
- Pens
- Paper

Grading:

- Grading will be on a pass-fail basis and determined on participation, attendance and completion of assigned course work.
- All assignment deadlines are final, no extensions will be granted.
- If a participant misses one day, it is up to him/her to reach out to the instructors for makeup assignments.
- Pass: A student must successfully complete at least 70% of the class objectives, tasks, and knowledge requirements
- Fail: a student who does not successfully complete at least 70% of the class objectives, tasks, and knowledge requirements

Assignments:

- This class is designed to give participants ample opportunity to complete all assignments in class. However, should a participant be unable to complete the assigned course work in class, it will be up to them to complete at home.

Topics Covered in This Course

Unit One

Basics

- Dynamic
- Numbers
- Only one type - no integer vs float
- Number limits
- Number operators
- Precision
- NaN
- Infinity/-Infinity

- Strings
- How they are written
- Breaking up code lines within stings
- Escaping
- Concatenation
- Template literals
- Unicode
- Numeric strings
- Adding numbers and strings
- Boolean
- Comparison Operators
- Logical Operators
- Arrays
- How written
- Zero-based
- Section 2: (4 Hours) - Data Types & Operators
- Objects
- Name:value pairs
- How written
- Typeof Operator
- Empty values
- Null and Undefined
- Difference between
- Operators
- Binary
- Unary
- Type conversion
- How expressions are evaluated (left to right)
- Value Comparison
- == vs ===
- Practice Examples

Unit Two

Programming Structure

- Definition of expression
- Bindings
- Var

- Const
- Let
- Binding names
- Functions
- Function definitions
- invoking/ calling
- Methods
- This keyword
- Accessing methods
- invoking/calling/applying
- Event
- Called
- Self-invoked
- Purpose of ()
- Accessing function without ()
- Arguments/Parameters
- Return values

- Control Flow
- Conditional Execution
- if/else
- For loops
- Types
- For
- for/in
- for/of
- while/do-while
- Differences
- break/continue
- Switch Statements
- Case
- Default
- Break
- Matching multiple cases
- Strict comparison
- Indentation and camelCase

Unit Three
Object Oriented Programming

- Function Syntax
- Defining Functions
- Parameters
- VS arguments
- Bindings and Scopes
- Global
- Local
- Function
- Block
- Nested scope
- Functions as values
- Function declaration
- Hoisting
- Var, let/const
- Initialization
- Arrow Functions
- Call stack
- Optional arguments
- Closure
- Recursion
- Controlling function size
- Functions and side effects
- Objects and Arrays
- Data sets/Arrays
- Creating
- Using [ ] for new array
- Multiple lines
- Keyword ‘new’
- Accessing elements of an array
- Changing array elements
- Accessing full array

- Looping through arrays
- Properties
- name/value pairs
- Objects
- Differences between arrays and objects
- Names indexes vs numbered indexes
- Object literals
- Accessing object members
- Mutability
- Identity
- Array Methods
- Length
- pop/push
- Substring
- shift/unshift
- indexOf/lastIndexOf
- forEach
- toString
- Join
- delete
- concat
- Sort
- Compare function
- Reverse
- String methods
- Slice/substring/substr
- differences
- indexOf/search
- differences
- Trim
- split/join
- Length
- toUpperCase/toLowerCase
- Repeat
- Replace
- Concat

- padStart/padEnd
- charAt/charCodeAt
- Number methods
- toString
- toExponential
- toFixed
- toPrecision
- valueOf
- Number method
- parseInt
- parseFloat
- Rest parameters
- Math object
- Namespace - definition
- Math functions
- min/max
- Sqrt
- Random
- Not really random
- floor/ceil
- cos/sin
- round
- pow
- Abs
- Math constants
- PI

Unit Four

Regular Expression/Asynchronous Programming

- Regular expressions
- Purpose
- Creating
- Testing for matches
- Sets of characters
- Modifiers
- Metacharacters
- Quantifiers

- Repeating parts of a pattern
- Grouping subexpressions
- Matches and groups
- Date class
- Word and string boundaries
- Choice patterns
- How matching works
- Backtracking
- Replace method
- Greed
- Search method
- lastIndex property
- Asynchronous programming
- Definition
- Callbacks
- Promises
- Event loop
- setTimeout

Unit Five
Client-Side Programming

- JSON
- Definition
- Structure
- serialize/deserialize
- Methods
- Stringify
- Parse
- Data Object
- Creating
- Different ways to create
- How dates are stored
- Date methods
- toString/toUTCString/toDateString/toISOString
- getFullYear/getMonth/getDate/getHours/getMinutes/getSeconds/getMilliseconds/getTime/getDay

- Date.now
- UTC Date methods
- Set Date methods
- setDate
- setFullYear
- setHours
- setMilliseconds
- setMinutes
- setMonth
- setSeconds
- setTime
- ISO dates
- Time Zones
- Long Dates
- Parsing Dates
- DOM
- Trees
- Traversal
- Getting elements
- Changing the document
- Creating nodes
- Attributes
- Layout
- Styling
- querySelectors
- Event handling
- Events and DOM nodes
- Event objects
- Propagation
- Default actions
- Types of events
- Key events
- Pointer events
- Scroll events
- Focus events
- Load event

- Event handler attributes
- Onclick
- Onchange
- Onmouseover
- Onmouseout
- Onkeydown
- onload
- Events and the event loop
- Timers
- Debouncing
- Encapsulation
- Prototypes
- Classes
- constructor
- Using
- Class notation
- Overriding prototype properties
- Maps
- Polymorphism
- Iterators (next)
- Getters and setters
- Static
- Inheritance
- instanceOf
- Bugs & Eros
- Console.log
- Dev tools
- Strict Mode
- Types
- try/catch/finally/throw
- Validation
- Testing
- Debugging
- Error propagation
- exceptions

Unit Six
Server-Side Programming

- Using MVC to Structure Our Application
- Node JS

- Introduction
- What is Node JS?
- Advantages of Node JS
- Traditional Web Server Model
- Node.js Process Model
- Setup Dev Environment
- Install Node.js on Windows
- Installing in mac os
- Working in REPL
- Node JS Console
- Modules
- Functions
- Buffer Module
- Module Types
- Core Modules
- Local Modules
- Module.Exports
- NPM
- What is NPM
- Installing Packages
- Locally Adding dependency in package.json
- Installing packages globally
- Updating packages
- Creating Web Server
- Creating web server
- Handling http request
- Sending request
- Events
- EventEmitter class

- Returning event emitter
- Inhering events
- Express JS
- Configuring routes
- Working with express
- Express application structures - routes, views, and static files
-
- Serving Static Resources
- Serving static files
- Working with middle ware

Unit Seven
Debugging Node JS Application

- Core Node JS debugger
- Node Inspector
- Debugging with Visual Studio
- Set breakpoints to pause execution
- Debug variables
- Debug functions
- Unit Testing with Mocha
- Test Driven Development
- Automate and Organize Tests
- Write Expressive Tests
- Learn TDD with Mocha

Unit Eight
Database /RESTful Services

- Database Connectivity
- installing MySQL Workbench
- Create sample Database
- Connection string
- Configuring
- Working with select command

- RESTful Web Services
- Postman
- REST API creation and consumption using JSON format.
- Connection
- Collections
- Updating records
- Deleting records

Unit Nine
Template Engines

- Why template engine
- What is jade
- Simple tags
- Adding attributes to tags
- Blocks of text
- Powerful Features
- Loops
- Javascript
- Interpolation
- Mixins
- Putting it all together
- What is vash
- Configuration
- Using vash with express.js
- Template options
- Helpers
- API
- Example

Final Project
Students will get more than a week to complete their final project but that last week will be used primarily for this. It will also be a time to ask questions pertaining to final project code or any errors etc.