****

**Barabari Web Development Program: MERN Stack Roadmap Semester 2 Blueprint 🏁**

### **📚 Semester 1 Recap – Foundation Building**

In the previous semester, students focused on building a strong foundation in web development through a **project-based approach**. They learned core **HTML and CSS** concepts by creating structured layouts. Every concept was paired with hands-on mini-projects, ensuring practical understanding.

Students also completed the **Responsive Web Design certification by freeCodeCamp**, which reinforced their skills through 15+ guided projects. Alongside this, they were introduced to essential developer tools like **Git and GitHub**, where they learned how to version control their code and contribute to repositories.

Additionally, students explored modern styling practices using the **Tailwind CSS utility-first framework**, which allowed them to rapidly build and customize UI components.

**🎯 JavaScript Fundamentals Roadmap**



# **Introduction to Programming**

* + What is Programming?
  + Why Learn Programming?

# **Difference Between Compiled and Interpreted Languages**

* + How Compilation Works
  + How Interpretation Works
  + Examples of Compiled vs Interpreted Languages

# **Introduction to JavaScript**

* + What is JavaScript and Why Use It?
  + History of JavaScript

# **How JavaScript Works**

* + JavaScript Engine
  + Execution Context
  + Call Stack

# **How JavaScript Code is Executed**

* + Step-by-step Execution of JS Code
  + Understanding Single-Threaded Behavior

# **6.** **Setting Up JavaScript**

* + Using the Browser Console
  + Setting Up VS Code for JavaScript
  + Installing and Using Node.js

# **7.** **Writing Your First JavaScript Program**

* + The "Hello, World!" Program



# **8.** **Variables and Data Types**

* + Declaring Variables (var, let, const)
  + Difference Between var, let, and const
  + Primitive Data Types (String, Number, Boolean, Null, Undefined, Symbol, BigInt)
  + Type Conversion (Implicit & Explicit)
  + Checking Data Type (typeof Operator)



# **9.** **Operators in JavaScript**

* + Arithmetic Operators (+, -, \*, /, %, \*\*)
  + Comparison Operators (==, ===, !=, !==, >, <, >=, <=)
  + Logical Operators (&&, ||, !)
  + Assignment Operators (=, +=, -=, \*=, /=, %=, \*\*=)
  + Unary Operators (++, --, typeof)



# **10.** **Control Flow (Conditional Statements & Loops)**

* + if, else if, else
  + Ternary Operator (condition ? trueValue : falseValue)
  + switch Statement
  + Entry Controlled and Exit Controlled Loops
  + for Loop
  + for...in Loop
  + for...of Loop
  + while Loop
  + do-while Loop
  + break and continue Statements



# **11.** **Functions in JavaScript**

* + Function Declaration vs Function Expression
  + Arrow Functions (() => {})
  + Function Parameters & Default Values
  + Function Parameters vs Arguments
  + Returning Values from Functions
  + Immediately Invoked Function Expressions (IIFE)
  + Callback Functions

* + Hoisting (Variables and Functions)



# **12.** **JavaScript Data Structures**

## **Arrays**

* + - Creating Arrays ([], new Array())
    - Array Methods (push, pop, shift, unshift, splice, slice, concat, map, filter, reduce)

## **Objects**

* + - Creating Objects ({ key: value })
    - Accessing Properties (dot notation, bracket notation)
    - Adding, Updating, and Deleting Properties
    - Object Destructuring



# **13.** **Asynchronous JavaScript (Basics)**

* + Synchronous vs Asynchronous Execution
  + Callbacks & Callback Hell
  + Promises (resolve, reject, .then, .catch)
  + async & await



# **14.** **JavaScript ES6+ Features**

* + Template Literals (Hello ${name})
  + Destructuring (Array & Object Destructuring)
  + Spread & Rest Operators (...)
  + Default Parameters in Functions
  + Modules (import / export)



# **15.** **Advanced JavaScript Concepts**

* + Higher-Order Functions (Functions as Arguments & Return Values)
  + this Keyword & How It Works
  + Prototypes & Prototype Inheritance



# **🎯 DOM JavaScript Roadmap with Projects and Assignments**

This roadmap covers the key topics in DOM JavaScript, with integrated projects and assignments that reinforce the concepts learned. Projects will be implemented during class, while take-home assignments provide additional practice.

## **Phase 1: Introduction to the DOM and Basic Interaction**

### **🧠 Introduction to the DOM**

**Concepts Learned:**

* What the DOM is and how JavaScript interacts with it.
* Basic DOM structure: elements, nodes, and tree structure.

**Key Topics:**

* document.getElementById(), document.querySelector()
* Accessing elements, modifying content, and attributes.

**💻 Project:** *None*

### **🛠️ Selecting and Manipulating DOM Elements**

**Concepts Learned:**

* Selecting and manipulating multiple DOM elements.
* Modifying styles and class names dynamically.

**Key Topics:**

* document.querySelectorAll(), .style, .classList
* Changing styles and classes, hiding/showing elements.

**💻 Project 1:** **Building a Digital Clock**

* DOM manipulation
* Real-time updates using setInterval()
* Working with the Date() object and dynamically updating the DOM.

**📝 Assignment 1 (Take-Home):** **Palindrome Checker**

* **Concepts:** String manipulation, recursion, DOM feedback display
* **Reason:** Solidifies understanding of recursion, string handling, and DOM updates.

## **Phase 2: Working with Events and Dynamic Content**

### **🧠 Event Handling in JavaScript**

**Concepts Learned:**

* Listening for user interactions (click, submit, etc.).
* Event listeners using addEventListener().

**Key Topics:**

* addEventListener(), this, event propagation (bubbling and capturing).

**💻 Project:** None

### **🛠️ Dynamic Content Updates**

**Concepts Learned:**

* Creating dynamic content based on user interaction.
* Updating inner HTML and creating/removing elements dynamically.

**Key Topics:**

* innerHTML, createElement(), appendChild(), removeChild()

**💻 Project 2:** **Building a To-Do List**

* Add/mark/delete tasks
* Dynamic DOM updates using events

**📝 Assignment 2 (Take-Home):** **Rock-Paper-Scissors Game**

* **Concepts:** Event-driven logic, DOM manipulation, game state handling
* **Reason:** Reinforces interaction handling through a fun game.

## **Phase 3: Advanced DOM Manipulation and Data Handling**

### **🧠 Local Storage and Data Persistence**

**Concepts Learned:**

* Storing data in the browser using localStorage.
* Retrieving and updating persistent data dynamically.

**Key Topics:**

* localStorage.setItem(), localStorage.getItem()
* JSON.stringify(), JSON.parse()

**💻 Project 3:** **Bookmark Manager**

* Add/remove bookmarks
* Persist data with localStorage and update DOM without page reload

**📝 Assignment 3 (Take-Home):** **Grade Analyzer**

* **Concepts:** Filtering, sorting, statistics, DOM updates
* **Reason:** Practice array operations, DOM rendering, and data persistence.

### **🛠️ Handling Complex DOM Interactions**

**Concepts Learned:**

* Representing entities with objects.
* Managing complex state and multiple DOM updates.

**Key Topics:**

* Object-based data management
* DOM updates based on object state

**💻 Project 4:** **Library Management System**

* Store/display a book collection
* Remove/update books from DOM using localStorage

**📝 Assignment 4 (Take-Home):** **Vehicle Rental System**

* **Concepts:** OOP with classes/inheritance, cost calculation
* **Reason:** Apply OOP principles with DOM interaction and data display.



### **🎯 Advance JS & Node**

### **Phase 1: Deep Dive into Error Handling and Event Handling**

1. **Advanced Error Handling**
   * **Concepts Learned:**
     + Types of JS errors: ReferenceError, TypeError, RangeError, etc.
     + Creating custom errors
     + Graceful degradation & fallback strategies
     + Error propagation
   * **Key Topics:**
     + throw with custom error classes
     + Nested try/catch
     + Centralized error handling patterns
     + Handling async errors in async/await
   * **Node Integration:**
     + Simulate file-not-found errors using fs.readFile()
     + Handle invalid user input in a CLI script using readline + try/catch
2. **Event Handling Beyond Basics**

* **Concepts Learned:**
  + Event bubbling, capturing, and delegation
  + Memory leaks with event listeners
  + Passive listeners and performance
* **Key Topics:**
  + event.stopPropagation(), event.preventDefault()
  + Event delegation pattern
  + Removing event listeners
  + Handling large-scale event systems

### **Phase 2: Asynchronous JavaScript (Real-World Usage)**

1. **Mastering Promises and Async Patterns**
   * **Concepts Learned:**
     + Promise chaining and composition
     + Error handling in long chains
     + Promise.all, Promise.any, Promise.allSettled, Promise.race
   * **Key Topics:**
     + Creating utility functions that return promises
     + Handling multiple async tasks with different needs
   * **Node Integration:**
     + Demonstrate fs.readFile() with callbacks and fs.promises.readFile() with async/await
     + Show sync vs async file access (fs.readFileSync)
   * **Activity:** Read and display data from files using both async/await and callback methods
2. **Async/Await in Practice**

* **Concepts Learned:**
  + Chaining and sequencing async/await operations
  + Error handling with async/await
  + Conditional and concurrent awaits
* **Key Topics:**
  + Refactoring nested promises into async functions
  + Using await in loops correctly

**5. Working with Fetch and Axios**

* **Concepts Learned:**
  + Making HTTP requests to APIs
  + Handling loading, error, and success states
  + CORS, headers, and request config
* **Key Topics:**
  + fetch with async/await + try/catch
  + Axios: configuring base URLs, interceptors
  + Comparison: fetch vs axios

Project break : **Recipe Finder** project

Focus: Making API calls using fetch() with async/await, Implementing UI states: loading spinner, error handling, success rendering, Filtering data in real-time based on user input, Navigating between views using dynamic data, Simulating network delay with Promises for UI testing.

### **Phase 3: Functional Patterns and Closures**

1. **Closures in Real-World Scenarios**
   * **Concepts Learned:**
     + Closures for data privacy and state retention
     + Memory considerations
   * **Key Topics:**
     + Implementing factory functions
     + Closures in event handlers and loops
     + Avoiding common closure pitfalls
   * **Node Integration:**
     + Use closures inside modules to simulate private state
   * **Activity:** Build a CLI command handler with closure-based logic
2. **Functional Programming with JavaScript**

* **Concepts Learned:**
  + Pure functions, immutability, and composition
  + Currying and partial application
* **Key Topics:**
  + Writing reusable utility functions
  + Using.reduce(),.filter() and .map() to build complex structures
  + Function chaining and pipelines
* **Node Integration:**
  + Apply these functions on data read from files using fs

Project break : **Budget Calculator** project

Focus: Using localStorage for persistent state across sessions, Working with complex data structures (objects, arrays) in the DOM, Dynamic DOM manipulation based on form inputs and calculations, Input validation and real-time UI updates, Prepares mental models for state-based UI—critical for transitioning to React.

### **Phase 4: Advanced ES6+ Techniques and Patterns**

1. **Advanced Module Management**
   * **Concepts Learned:**
     + Dynamic imports
     + Import maps and lazy loading
     + Organizing code into modules effectively
   * **Key Topics:**
     + import() (dynamic)
     + Exporting classes, default/named exports
     + Practical module bundling concepts (Vite/Webpack basics)
   * **Node.js Integration**
     + CommonJS Modules: require, module.exports
     + Use built-in modules like path and fs
     + File systems with node js
     + Read/write files using fs
     + Compare synchronous and asynchronous methods
2. **JavaScript Patterns and Best Practices**
   * **Concepts Learned:**
     + Module pattern, revealing module pattern
     + Singleton, factory, observer (intro level)
   * **Key Topics:**
     + Writing maintainable code using patterns
     + Encapsulation using IIFE + closure

Project break : **Movie Tracker** project

Focus: Modular architecture using ES6 modules for scalable code organization, Leveraging modern JavaScript features like arrow functions, destructuring, spread/rest operators, and Sets, Implementing dynamic filtering and sorting with efficient array methods, Managing session persistence for user data, Responsive and accessible UI design with event-driven programming, Reinforces real-world patterns and best practices essential for advanced frontend development and preparing for frameworks like React.

### **Phase 4: HTTP Server Basics and Package Management**

1. **Creating an HTTP server**
   * **Concepts Learned:**
     + Http module basics
     + Handling requests and responses
     + Setting status codes and Headers
   * Project: Mini Static Website Server
     + Description: Create a server using the http module that serves HTML files based on the URL path (/, /about, /contact) and handles 404 errors.
     + Concepts Covered: http module, request and response handling, status codes, file reading with fs, routing.
2. **NPM and Package Management**

**Concepts Learned:**

* + - Install local/global packages
    - Explain package.json, package-lock.json, semantic versioning
    - Use packages like chalk or nodemon in prior projects

### **🎯 React Roadmap**

**UNIT - 1**

#### **1. Introduction to React**

* What is React and Why Use It?
* Brief History of React
* SPA (Single Page Applications) Concept

#### **2. Setting Up a React Environment**

* Installing Node.js & npm
* Using Create React App (CRA)
* Folder Structure Overview

#### **3. JSX Basics**

* What is JSX?
* Embedding Expressions in JSX
* JSX vs HTML Differences

Project break

Focus: React setup, component structure, and JSX.

**UNIT - 2**

#### **4. Components in React**

* Functional vs Class Components
* Creating & Using Components
* Props in Components
* Reusability and Composition

#### **5. State Management**

* Using useState Hook
* Updating State
* State vs Props

Project break

Focus: Props, useState, component interaction.

**UNIT - 3**

#### **6. Handling Events**

* Event Handlers in JSX
* Passing Parameters to Handlers

#### **7. Conditional Rendering**

* if-else, Ternary, && Operator
* Rendering Based on Props/State

#### **8. Lists and Keys**

* Rendering Lists with map()
* Importance of Keys in Lists

Project break

Focus: State toggling, event handling and conditional rendering to dynamically switch UI themes

**UNIT - 4**

#### **9. Forms in React**

* Controlled vs Uncontrolled Components
* Handling Form Submission
* Validating Form Data

Project break

Focus: Controlled forms, list rendering, useState

**UNIT - 5**

#### **10. useEffect Hook**

* Side Effects in Functional Components
* useEffect with Dependencies
* Cleanup Functions

#### **11. Lifting State Up**

* Sharing State Between Components
* Prop Drilling

Project break

Focus: useEffect, lifting state up, prop drilling.

**UNIT - 6**

#### **12. Component Lifecycle (Class Components)**

* Mounting, Updating, Unmounting
* Lifecycle Methods: componentDidMount, etc.

#### **13. Routing in React**

* React Router Basics
* Route, Link, useParams, useNavigate

Project break

Focus: React Router, lifecycle methods for class components.

**UNIT - 7**

#### **14. Context API**

* Creating and Using Context
* useContext Hook
* Avoiding Prop Drilling

#### **15. Custom Hooks**

* Creating Custom Hooks
* Best Practices

Project break

Focus: Simple login simulation using Context API and custom hooks.

**UNIT - 8**

#### **16. React Performance Optimization**

* Memoization (React.memo, useMemo, useCallback)
* Lazy Loading & Code Splitting

#### **17. React and APIs**

* Fetching Data with fetch/axios
* Handling Loading and Error States

#### **18. Advanced Patterns**

* Render Props
* Higher-Order Components (HOCs)
* Compound Components

Project break

Focus: useContext, useMemo, API integration, performance.

## **🔥 Full Backend Roadmap: Express + MongoDB + APIs**

### **Backend Development**

#### **Unit 1: Introduction to Backend**

* What is backend development?
* Frontend vs Backend
* Power of JavaScript in backend
* Client-server architecture
* What is an API?
* REST API fundamentals

🧠 **Mini Reflection Task:** Draw and explain client-server flow for a simple login form.

**Outcomes:**

* Understand the role of backend in web applications
* Explain client-server communication flow
* Define REST and explain how APIs work

#### **Unit 2: Intro to Express.js**

* What is Express.js?
* Setting up an Express server
* Understanding Middleware (use, next)
* Request and response objects
* Handling routes (GET, POST, PUT, DELETE)
* Serving static files

💡 **Checkpoint Activity:** Serve a static HTML + CSS login form using Express.

**Outcomes:**

* Set up a basic Express.js server
* Handle different routes and requests
* Serve static frontend content from the backend

#### **Unit 3: Build Your First API**

* What is JSON data?
* Using body-parser or express.json()
* CRUD operations (without DB)
* Error handling and HTTP status codes
* Modularizing routes and controllers
* Using Postman to test APIs

🧪 **Task:** Use Postman to test all CRUD endpoints for a basic Task Manager API.

🏅 **Milestone:** ✅ *Complete Postman Student Expert Certification*

***Outcomes:***

* *Build a functional REST API in Express*
* *Modularize routes and controllers*
* *Test endpoints using Postman*

#### **🔨 Mini Project Break 1: Task Manager API**

* In-memory CRUD
* Modular controllers
* Postman documentation

### **Working with Databases**

#### **Unit 4: Introduction to MongoDB**

* What is a database?
* SQL vs NoSQL
* What is MongoDB?
* Installing MongoDB locally or using Atlas
* Collections vs Documents
* Basic queries: find, insert, update, delete
* Using MongoDB Compass

📚 **Practice Task:** Insert, update, and delete a Books collection using Compass.

🏅 **Milestone:** ✅ *Earn a badge from* [*MongoDB University*](https://learn.mongodb.com/) (e.g., "Introduction to MongoDB")

**Outcomes:**

* Understand MongoDB structure and concepts
* Perform database operations via CLI and Compass
* Know how and when to use NoSQL databases

#### **Unit 5: Connecting Express to MongoDB**

* What is Mongoose?
* Connecting to MongoDB Atlas
* Defining schemas and models
* CRUD with Mongoose
* Schema validation and error handling

🛠️ **Refactor Task Manager API:**

* Replace in-memory storage with MongoDB
* Add proper schema validation

**Outcomes:**

* Connect an Express app to a real MongoDB database
* Create and validate schemas using Mongoose
* Refactor API to store and retrieve data from MongoDB

#### **🔒 Unit 6: Authentication & Authorization**

**Concepts Covered:**

* What is authentication vs authorization?
* Hashing passwords using bcrypt
* Generating and verifying JWT tokens
* Creating login and signup routes
* Protecting routes with middleware
* OAuth 2.0 login using Google (via Passport.js or Firebase Auth)

**Outcomes:**

* Implement secure login/signup using hashed passwords and JWT
* Create protected routes that only authenticated users can access
* Understand and integrate OAuth-based login (e.g., Google Sign-In)
* Differentiate between manual and third-party authentication flows

🧪 **Practice Task:** Add auth to Book or Task Manager API. Allow each user to manage their own data only.

#### **🔨 Mini Project Break 2: Book Review API**

* Users can register, login
* Authenticated users can CRUD books
* Users can leave reviews (nested schema)
* Include pagination, filtering, and validation

### **🌟 Capstone Project**

**🧱 Project Title:** "EduConnect API" (or any creative theme)

**Features:**

* Auth system (signup, login, JWT)
* Users can CRUD content (books/tasks/posts)
* Nested resources (e.g., reviews or comments)
* Pagination, search, filter
* Postman collection with tests

### **🎯 AI in Web Development**

* **Understanding AI in Web Development:**
  + Time-saving, error reduction, and enhanced creativity.
  + Code generation, design assistance, performance optimization, and accessibility improvements.
* **AI Tools:**
  + GitHub Copilot
  + Claude AI
  + Gemini
  + ChatGPT

**Project:** Optimizing current projects using these AI tools.

# **🧭 Full DSA Roadmap for Students**

## **📘 Module 0: Introduction to DSA**

**Topics:**

* What is Data Structures and Algorithms?
* Why is DSA important?  
  + For problem-solving, efficiency, job interviews, competitive coding, logical thinking
* Real-life examples  
  + Maps, Auto-complete, Uber ETA, etc.
* How to learn DSA effectively  
  + Practice → Understand → Optimize → Repeat
* Tools to use  
  + VS Code
  + JS Console
  + LeetCode
  + SkillCaptain

## **✅ Module 0.5: Basic Math for DSA**

**Topics:**

* Number Properties & Divisibility  
  + Prime numbers
  + GCD & LCM (Euclidean Algorithm)
  + Divisibility rules
* Modular Arithmetic  
  + Basics of % operator
  + Properties: mod with addition and multiplication
  + Fast Exponentiation (Binary exponentiation)
* Bit Manipulation Basics  
  + Binary representation
  + AND, OR, XOR, NOT
  + Use-cases: check even/odd, count bits, XOR swap
* Basic Math Formulae  
  + Sum of N natural numbers
  + Sum of squares, cubes (briefly)

## **📘 Module 1: Big-O Notation & Core Fundamentals**

**Topics:**

* Time and Space Complexity
* Big-O Notations:  
  + O(1), O(n), O(log n), O(n log n), O(n²)
* Case analysis  
  + Best, worst, average cases
* Analyze JS code snippets

## **📘 Module 2: Arrays & Problem-Solving Patterns**

**Topics:**

* Array Operations  
  + Traversal, Insert, Delete
  + Built-in methods (push, pop, splice, slice, etc.)
* Two Pointer Technique  
  + Reverse array
  + Pair sum
  + Move zeros
* Sliding Window  
  + Max sum subarray
  + Longest substring without repeat
* Common Interview Questions  
  + Duplicates
  + Missing number
  + Intersection

## **📘 Module 3: Linked Lists**

**Topics:**

* Singly Linked List  
  + Insert at head/tail
  + Delete
  + Search
* Doubly Linked List
* Problems:  
  + Reverse a list (iterative & recursive)
  + Merge two sorted lists
  + Detect cycle (Floyd’s Tortoise and Hare)

## **📘 Module 4: Stacks and Queues**

**Topics:**

* Stack  
  + LIFO, Use cases (undo, brackets)
  + Implementation with array
  + Problems: Valid Parentheses, Next Greater Element
* Queue  
  + FIFO, Use cases (scheduling, printers)
  + Circular Queue concept
  + Implementation with array
* Practice Problems  
  + Min Stack
  + Queue using Stacks
  + Stack using Queues

## **📘 Module 5: Hashing & Sets**

**Topics:**

* Hash Tables in JS  
  + Map, Object
* Set for uniqueness
* Frequency Counter Pattern
* Common Problems  
  + Anagrams
  + First non-repeating character
  + Longest consecutive sequence
  + Count frequency
  + Group anagrams

## **📘 Module 6: Algorithms**

**Recursion**

* Stack frames, base cases, tail recursion
* Factorial, Fibonacci, Array sum
* When to use recursion vs iteration

**Backtracking**

* Subsets, Permutations
* N-Queens
* Rat in a Maze
* Sudoku Solver
* Binary Strings without consecutive 1s

**Searching Algorithms**

* Linear Search
* Binary Search (sorted input required)

**Sorting Algorithms**

* Bubble, Selection, Insertion (intro sorts)
* Merge Sort (Divide & Conquer)
* Time/space comparisons

**Kadane’s Algorithm**

* For max subarray sum

## **📘 Module 7: Trees**

**Topics:**

* Basics  
  + Node, Root, Parent, Child, Leaf, Height, Depth
  + Binary Tree vs BST
* Tree Traversal Techniques  
  + Inorder (LNR), Preorder (NLR), Postorder (LRN)
  + Level Order Traversal (using Queue)
  + Recursive & Iterative implementation
* Common Problems  
  + Max depth
  + Check balanced tree
  + Lowest Common Ancestor (LCA)
  + Path sum
  + Same tree check

## **📘 Module 8: Graphs**

**Topics:**

* Introduction  
  + Graph Terminology: nodes, edges, directed/undirected, weighted
  + Adjacency List vs Matrix representation
* Graph Traversal Techniques  
  + BFS (Breadth First Search)
  + DFS (Depth First Search) – recursive & iterative
* Applications & Problems  
  + Detect Cycle (directed & undirected)
  + Count connected components
  + Shortest Path (BFS-based for unweighted graphs)
  + Word Ladder
  + Number of Islands
  + Clone Graph

## **📘 Module 9: Greedy Algorithms**

**Topics:**

* Activity Selection
* Minimum Coins
* Jump Game
* Interval Scheduling
* When Greedy fails vs when it works

## **📘 Module 10: Dynamic Programming (DP)**

**Topics:**

* Introduction  
  + Memoization vs Tabulation
  + Overlapping subproblems & optimal substructure
* Classic Problems  
  + Fibonacci (recursion → memoization → tabulation)
  + Climbing Stairs
  + 0/1 Knapsack
  + Subset Sum
  + Longest Common Subsequence
  + Minimum Path Sum