JAVASCRIPT

BEGINNER TO ADVANCED

Day 1: Introduction to JavaScript

- JavaScript: A high-level, interpreted programming language used to add interactivity and dynamic behavior to web pages.
- Variables: Containers for storing data values.
- Data Types: Types of values that can be used in JavaScript, such as numbers, strings, booleans, objects, etc.
- Operators: Symbols used to perform operations on variables and values.
- Functions: Blocks of code that can be defined once and executed multiple times.

Day 2: Control Flow and Conditionals

- Conditional Statements: **if**, **else if**, and **else** statements for executing different code blocks based on different conditions.
- Comparison Operators: Symbols used to compare values, such as **==**, **===**, **!=**, **!==**, **<**, **>**, **<=**, **>=**.
- Logical Operators: Symbols used to combine conditional statements, such as **&&**, **||**, **!**.
- Ternary Operator: Shorthand for writing conditional statements in a single line.

Day 3: Loops and Iteration

- for Loop: Executes a block of code a specified number of times.
- While Loop: Executes a block of code while a specified condition is true.
- do...while Loop: Similar to the **while** loop, but the block of code is executed at least once before the condition is tested.
- for...in Loop: Iterates over the properties of an object.
- for...of Loop: Iterates over the values of an iterable object like arrays or strings.

Day 4: Arrays and Objects

- Arrays: Ordered collections of values, accessed by numeric indices.
- Array Methods: Functions that can be called on arrays to manipulate their contents, such as **push()**, **pop()**, **shift()**, **unshift()**, **splice()**, **slice()**, etc.
- Objects: Collections of key-value pairs, where values can be accessed by keys.
- Object Methods: Functions that are properties of objects.

Day 5: Functions and Scope

- Function Declarations: Defining functions using the **function** keyword.
- Function Expressions: Defining functions as expressions, often assigned to variables.
- Arrow Functions: A more concise syntax for writing functions.
- Scope: The context in which variables are declared and can be accessed.
- Global Scope: Variables declared outside of any function, accessible throughout the entire program.
- Local Scope: Variables declared within a function, accessible only within that function.

Day 6: DOM Manipulation

- DOM (Document Object Model): A programming interface for web documents, representing the structure of an HTML document as a tree of objects.

- Selecting Elements: Methods like **getElementById()**, **getElementsByClassName()**, **getElementsByTagName()**, **querySelector()**, and **querySelectorAll()** to select elements from the DOM.

- Modifying Elements: Methods like **innerHTML**, **textContent**, **setAttribute()**, **classList**, etc., to modify the content and attributes of elements.

- Creating and Removing Elements: Methods like **createElement()**, **appendChild()**, **removeChild()**, etc., to dynamically create and remove elements from the DOM.

Day 7: Event Handling

- Events: Actions that occur as a result of user interactions or other triggers.
- Event Handlers: Functions that are executed when a specific event occurs.
- Event Listeners: Methods like **addEventListener()** to attach event handlers to elements.
- Event Object: An object containing information about the event, passed to event handler functions as an argument.
- Event Propagation: The order in which event handlers are executed, either capturing phase or bubbling phase.
- Event Delegation: Technique for handling events on multiple elements with a single event handler.

Day 8: Asynchronous JavaScript

- Callbacks: Functions passed as arguments to other functions and executed later.

- Promises: Objects representing the eventual completion or failure of an asynchronous operation.

- async/await: Keywords used with asynchronous functions to write asynchronous code in a synchronous style.

- XHR (XMLHttpRequest): Object used to interact with servers and make HTTP requests from web browsers.

- Fetch API: Modern alternative to XHR for making HTTP requests in JavaScript.

- AJAX (Asynchronous JavaScript and XML): Technique for updating parts of a web page without reloading the whole page.

Day 9: ES6 and Modern JavaScript

- ES6 (ECMAScript 2015): Major update to the JavaScript language, introducing new syntax and features.

- Arrow Functions: A more concise syntax for writing functions.

- Template Literals: Strings that allow embedded expressions.

- Destructuring Assignment: Extracting values from arrays or objects and assigning them to variables.

- Spread Operator: Expands an iterable (like an array) into individual elements.

- Classes & Inheritance: Prototypal inheritance in JavaScript using class syntax.

- Modules: Encapsulating code into reusable modules using import and export statements.

Day 10: Advanced JavaScript Concepts

- Closures: Functions that remember the scope in which they were created, even after that scope has closed.

- Prototypes and Prototypal Inheritance: The mechanism by which JavaScript objects inherit features from one another.

- Context (this): A reference to the object that owns the currently executing code.

- Execution Context and Hoisting: The context in which JavaScript code is executed and the process of moving variable and function declarations to the top of their containing scope.

- Event Loop: The mechanism that allows JavaScript to perform non-blocking operations.

- Memory Management: How JavaScript manages memory allocation and deallocation, including garbage collection.

Day 11 & 12: Functional Programming & Advanced JS

- Functional Programming: A programming paradigm focused on building software by composing pure functions and avoiding shared state, mutable data, and side-effects.
- Pure Functions: Functions that return the same output for the same input and do not produce side effects.
- Immutability: The principle that data should not be changed after it is created.
- Higher-Order Functions: Functions that take other functions as arguments or return functions.
- Map, Filter, & Reduce: Higher-order functions commonly used in functional programming for transforming and aggregating data.
- Recursion: A technique where a function calls itself in order to solve smaller instances of the same problem.
- Module Patterns: Techniques for encapsulating and organizing code into modules.
- Singleton Pattern: A design pattern that restricts the instantiation of a class to a single instance.
- Observer Pattern: A design pattern where an object, called the subject, maintains a list of its dependents, called observers, and notifies them of any state changes.
- Promises & Async/Await Patterns: Patterns for managing asynchronous code and handling asynchronous operations in JavaScript.
- Memoization: A technique of storing the results of expensive function calls and returning the cached result when the same inputs occur again.