Foundations checklist

Key Notes:

- Keep track of your progress by checking off completed items
- Practice regularly to reinforce your learning
- Seek help from peers, mentors, or online resources when needed
- Take breaks and prioritize self-care to avoid burnout
- Focus on understanding concepts rather than just memorizing syntax
- Write clean and well-organized code
- Use version control and commit regularly to avoid losing work
- Test your code thoroughly to catch errors and bugs early
- Continuously seek to improve and learn new skills
- Learn pattern matching techniques to simplify complex code blocks
- Happy coding!

Week 1

This checklist includes basic Git commands and introductory JavaScript concepts.

Git Commands

☐ Initialize a new Git repository

☐ Add files to the staging area

☐ Commit changes with a message

☐ Check the status of the repository

Introductory JavaScript Concepts

☐ Declare variables using `let` and `const`

☐ Use data types such as strings, numbers, and booleans

- □ Utilize arithmetic operators for mathematical operations
- □ Write conditional statements with `if`, `else if`, and `else`
- □ Create loops with `for` and `while`

Week 2

This checklist includes concepts related to functions, objects, and arrays.

Functions

- □ Define a function using the `function` keyword
- □ Utilize parameters and arguments in function definitions
- □ Return values from a function using the `return` keyword
- □ Assign functions to variables or object properties
- □ Use arrow function syntax

Objects

- □ Create an object using object literal notation
- □ Access object properties using dot notation and bracket notation
- □ Modify object properties using dot notation and bracket notation
- □ Utilize object methods

Arrays

- □ Create an array using array literal notation
- □ Access array elements using bracket notation
- □ Modify array elements using bracket notation
- □ Utilize array methods such as `push`, `pop`, `shift`, and `unshift`
- □ Utilize built-in array methods such as `forEach`, `map`, `filter`, and `reduce`

Week 3

This checklist includes concepts related to HTML and CSS.

HTML

- □ Understand the basic structure of an HTML document
- □ Utilize HTML tags for headings, paragraphs, and lists
- □ Create hyperlinks using the `a` tag
- □ Add images to a webpage using the `img` tag
- □ Utilize semantic HTML tags such as `header`, `nav`, `main`, and `footer`

CSS

- □ Understand the basic syntax of CSS
- □ Apply CSS styles to HTML elements using selectors and declarations
- □ Utilize CSS properties such as `color`, `font-size`, and `background-color`
- □ Use CSS to layout and position HTML elements
- □ Understand responsive design and media queries

DOM Manipulation

- □ Understand the Document Object Model (DOM)
- □ Access and modify HTML elements using JavaScript
- □ Utilize event listeners to respond to user actions
- □ Create and remove HTML elements dynamically

JavaScript in the Browser

- □ Understand the basics of the browser environment
- □ Utilize the console for debugging and logging
- □ Understand Axios and asynchronous programming

Week 4

This checklist covers concepts related to Node.js, npm, Express, and RESTful APIs.

Node.js and npm

□ Understand the basics of Node.js and npm.

□ Install and manage packages using npm.

□ Create a Node.js project using `npm init`.

□ Utilize `npm run` scripts.

Express

□ Understand the basics of Express.

□ Create a basic HTTP server using Express.

□ Utilize middleware.

□ Use request and response objects to handle HTTP requests and responses.

□ Access body, parameters and queries.

□ serve static files to the client.

RESTful APIs

□ Understand the principles of RESTful APIs.

□ Define API endpoints and HTTP methods.

□ Parse request bodies to access data sent by clients.

□ Understand HTTP response codes.

□ access and work with external API's.

Week 5

This checklist covers concepts related to PostgreSQL, Sequelize, and key database concepts.

PostgreSQL

□ Understand the basics of relational databases and PostgreSQL.

□ Install and set up PostgreSQL on your machine.

□ Create a new database and tables using SQL.

□ Understand how to query a PostgreSQL database.

Sequelize

□ Understand the basics of Sequelize.

□ Set up a Sequelize project.

□ Create, read, update, and delete records using Sequelize.

Key Database Concepts

□ Understand the fundamentals of database design.

□ Know the difference between primary and foreign keys.

□ Understand one-to-one, one-to-many, and many-to-many relationships.

□ Be familiar with SQL data types.

Week 6

This checklist covers concepts related to QA & Testing, automation using Selenium, and basic deployment and DevOps.

QA & Testing

□ Understand the importance of quality assurance and testing in software development.

□ Know the different types of testing, such as unit testing, integration testing, and end-to-end testing.

□ Understand the testing pyramid and how it relates to different types of testing.

□ Know how to write test cases and test plans.

Automation Using Selenium

□ Understand the basics of Selenium.

□ Set up a Selenium project.

□ Write and execute automated tests using Selenium.

Basic Deployment and DevOps

□ Understand the basics of deployment and DevOps.

□ Know how to deploy a web application to a server.

□ Understand the basics of cloud computing and how to deploy web applications to the cloud.

Week 7

This checklist covers computer science topics such as data structures, algorithms basics, and Big-O notation.

Data Structures

□ Understand the basics of data structures such as arrays, linked lists, stacks, and queues.

□ Know the advantages and disadvantages of different data structures in different scenarios.

□ Be able to implement data structures in JavaScript.

Algorithms Basics

□ Understand the basics of algorithms such as sorting, searching, and recursion.

□ Know the advantages and disadvantages of different algorithms in different scenarios.

□ Be able to implement algorithms in JavaScript.

Big-O Notation

□ Understand the basics of Big-O notation and time complexity analysis.

□ Know how to analyze the time complexity of code using Big-O notation.

□ Understand the differences between different time complexity classes.

□ Understand the basics of space complexity and how it relates to Big-O.