WEEK1

**Day 1: Introduction to JavaScript**

On the first day, we covered the basics of JavaScript, including its history and usage. We also discussed the syntax and structure of JavaScript code and how it interacts with HTML and CSS. We learned about variables, data types, operators, and how to use comments in our code.

**Day 2: Control Flow and Loops**

On the second day, we explored control flow and loops in JavaScript. We learned about conditional statements, such as if/else statements, and how to use them to control the flow of our code. We also discussed loops, such as for and while loops, and how they can be used to repeat code.

**Day 3: Functions**

On the third day, we focused on functions in JavaScript. We learned how to define and call functions, how to pass arguments to them, and how to use return statements to get values from functions. We also covered concepts such as function scope and closures.

**Day 4: Arrays and Objects**

On the fourth day, we delved into arrays and objects in JavaScript. We learned how to create and manipulate arrays, including methods like push(), pop(), and splice(). We also discussed how to create objects and how to access and modify their properties.

**Day 5: DOM Manipulation**

On the fifth day, we learned about the Document Object Model (DOM) and how to manipulate HTML elements with JavaScript. We covered concepts such as selecting elements with JavaScript, modifying their properties, and adding or removing elements from the page.

**Day 6: Events and Event Listeners**

On the sixth day, we covered events and event listeners in JavaScript. We learned how to use event listeners to trigger JavaScript code when specific events occur, such as clicking a button or submitting a form. We also discussed how to handle events and how to prevent default behaviours.

Week2

**Day 1: Advanced Data Types**

On the first day, we covered advanced data types in JavaScript such as Maps, Sets, and Symbols. We learned about the differences between these data types and how they can be used in various scenarios, including working with complex data structures and ensuring uniqueness.

**Day 2: Asynchronous Programming**

On the second day, we dove into asynchronous programming in JavaScript, including the use of callbacks, promises, and async/await. We discussed how to handle asynchronous code, avoid callback hell, and ensure code readability and maintainability.

**Day 3: Object-Oriented Programming**

On the third day, we focused on Object-Oriented Programming (OOP) in JavaScript, including class syntax, inheritance, and polymorphism. We learned how to define classes, create objects, and use OOP principles to write more modular and scalable code.

**Day 4: Functional Programming**

On the fourth day, we delved into functional programming in JavaScript, including higher-order functions, closures, and currying. We learned how to write code that is more declarative and expressive, and how to use functional programming concepts to solve complex problems.

**Day 5: Error Handling**

On the fifth day, we covered error handling in JavaScript, including try/catch blocks, throwing and catching errors, and handling asynchronous errors. We learned how to write code that is more robust and resilient, and how to debug and troubleshoot errors effectively.

**Day 6: ES6 and Beyond**

On the sixth day, we explored some of the newer features and syntax of JavaScript, including template literals, arrow functions, destructuring, and spread syntax. We also discussed the use of modules and the importance of keeping up-to-date with the latest developments in the language.

Week 3

**Day 1: Introduction to Mapty**

On the first day, we started by introducing Mapty, our map integration project that tracks user workout locations. We discussed the features of Mapty, including the use of APIs, the creation of custom markers, and the ability to add different types of workouts.

**Day 2: Setting Up the Environment**

On the second day, we focused on setting up the environment for our project, including installing necessary packages and configuring our development environment. We also discussed the importance of organizing our code and creating a clear project structure.

**Day 3: User Interface Design**

On the third day, we discussed the user interface (UI) design of our Mapty project. We covered topics such as designing responsive layouts, incorporating Google Maps into our UI, and styling our interface using CSS.

**Day 4: Working with APIs**

On the fourth day, we covered the topic of working with APIs in our Mapty project. We learned how to interact with third-party APIs to retrieve location data, and how to use this data to display workout locations on our map.

**Day 5: Implementing the Core Functionality**

On the fifth day, we implemented the core functionality of our Mapty project, including the ability to add and delete workouts, as well as to display workout information such as duration, distance, and calories burned.

**Day 6: Advanced Features**

On the sixth day, we added advanced features to our Mapty project, such as geolocation, which allows the user to automatically detect the location, and the ability to save workouts to local storage so that they can be retrieved and viewed later.

WEEK 4

**Day 1: Introduction to Git**

On the first day, we introduced Git, a version control system that helps manage and track changes to code. We covered the basics of Git, including creating a repository, committing changes, and pushing changes to a remote repository.

**Day 2: Branching and Merging**

On the second day, we covered branching and merging in Git. We learned how to create branches, switch between branches, and merge branches to incorporate changes into the main codebase. We also discussed best practices for branching and merging to avoid conflicts.

**Day 3: Working with Remote Repositories**

On the third day, we discussed working with remote repositories in Git. We learned how to clone a remote repository, push changes to a remote repository, and pull changes from a remote repository. We also covered topics such as collaborating with others on a shared repository and managing conflicts in a collaborative environment.

**Day 4: Git Workflow Strategies**

On the fourth day, we covered Git workflow strategies. We discussed popular workflows such as the centralized workflow, feature branch workflow, and Gitflow workflow. We also discussed best practices for Git workflows, such as using descriptive commit messages and creating clear documentation.

**Day 5: Advanced Git Concepts**

On the fifth day, we covered advanced Git concepts such as rebasing, stashing, and cherry-picking. We learned how to use these advanced features to manage complex changes to code and resolve conflicts in a more efficient manner.

**Day 6: Git Hosting and Services**

On the final day, we discussed Git hosting and services. We covered popular Git hosting platforms such as GitHub, GitLab, and Bitbucket. We also discussed the benefits of using Git hosting services, such as easy collaboration and access to tools for issue tracking, project management, and code review.

Week 5

Day 1: Learning JavaScript for React

On the first day, I reviewed the fundamentals of JavaScript. I focused on the following topics:

Variables and data types: I learned about the different types of variables in JavaScript and how to use them to store and manipulate data.

Functions: I learned about the different types of functions in JavaScript and how they can be used to organize code and perform actions.

I also learned about the importance of variables in React and how they can be used to store and manipulate data. I practiced writing simple JavaScript code and manipulating variables in React.

Day 2: JavaScript Syntax and Structure

On the second day, I delved deeper into the syntax and structure of JavaScript. I focused on the following topics:

Object-oriented programming: I learned about the basics of object-oriented programming and how it can be used in React to create components.

Operators: I learned about the different types of operators in JavaScript and how they can be used to perform calculations and comparisons.Control flow: I learned about the different types of control flow statements in JavaScript and how they can be used to control the flow of execution in a program.

I practiced writing more complex JavaScript code and creating simple React components using object-oriented programming concepts.

Day 3: Asynchronous Programming and Event Handling

On the third day, I focused on more advanced JavaScript concepts, including asynchronous programming and event handling. I focused on the following topics:

Asynchronous programming: I learned about the basics of asynchronous programming in JavaScript and how it can be used in React to handle data from APIs and other sources.

Callbacks and promises: I learned about callbacks and promises, which are common techniques for handling asynchronous code in JavaScript.

Event handling: I learned about the basics of event handling in React, including how to create event listeners and handle user input.

Day 4: Basics of React Components and Folder Structure

On the fourth day, I started by learning about the basics of React. I focused on the following topics:

Component-based architecture: I learned about the component-based architecture of React and how it can be used to create reusable components.

JSX syntax: I learned about JSX, which is a syntax extension for JavaScript that allows for the creation of React components.

Basic folder structure: I learned about the basic folder structure of a React application, including the src folder, the public folder, and the index.js file.

I practiced creating simple React components and organizing them into a basic folder structure.

Day 5: State Management

On the fifth day, I delved deeper into React and learned about state management and React Hooks. I focused on the following topics:

State management: I learned about the importance of state in React and how it can be used to manage data in a component.

Day 6: React Hooks

React Hooks: I learned about React Hooks, which are a way to use state and other React features in functional components.

Props: I learned about the importance of props in React and how they can be used to pass data between components.

I practiced using state and props to manage data in a React component and created simple React Hooks to manage state in a functional component.

**WEEK 6**

Day 1: Today, I focused on learning the basics of React, including how to create components, render them on a page, and use props to pass data between components. I also learned about React state, which allows components to manage and update their own data. Additionally, I started to explore different ways of managing state, including using Redux.

Day 2: I continued my exploration of state management today by delving deeper into the different approaches and techniques available for managing state in React. This included learning about React context, which allows data to be passed down the component tree without having to pass props through each level.

Day 3: Today, I started building a project to practice my React skills. I chose to create a basic restaurant UI, which involved creating components for the menu, a reservation form, and a contact form. As I built out the project, I experimented with different state management techniques, including using Redux and React context.

Day 4: I spent today learning how to debug React applications, including how to use the React Developer Tools and other browser-based debugging tools. I also learned about common debugging strategies, such as logging and using breakpoints to pause code execution.

Day 5: I continued working on my restaurant project today, focusing on improving the user experience by implementing various hooks. This included using the useEffect hook to update the component when the user interacts with it, as well as using the useState hook to manage the component's internal state.

Day 6: On my final day of learning React, I focused on learning more advanced hooks, such as the useContext hook, which allows components to access context data without having to pass it down as props. I also explored other hooks, such as the useReducer hook, which can be used to manage more complex state. Overall, I feel much more confident in my React skills and look forward to continuing to build projects and explore the React ecosystem.

**WEEK 7**

Day 1:

Spent the day getting familiar with the React library and its core concepts such as components, props, and state

Learned about Redux and its purpose as a state management tool for React applications

Started working through the official Redux tutorial to gain a deeper understanding of how it works

Day 2:

Continued working through the Redux tutorial, focusing on concepts like actions, reducers, and the store

Began experimenting with Redux in a small React application, creating basic actions and reducers to manage state

Day 3:

Expanded on the previous day's work with Redux by learning about middleware and how it can be used to extend Redux functionality

Incorporated middleware into the React application to handle async actions and log changes to the Redux store

Day 4:

Moved onto advanced Redux concepts such as selectors and the Redux DevTools extension

Implemented selectors to efficiently retrieve specific parts of the Redux state and utilized the DevTools to debug the application

Day 5:

Shifted focus to authentication in React applications and the various approaches that can be used to secure user data

Explored different authentication methods such as session-based and token-based authentication

Day 6:

Wrapped up the week by implementing authentication in the React application using the Firebase authentication service

Configured Firebase and set up the authentication flow, allowing users to sign up, log in, and log out of the application

**WEEK 8**

Day 1:

Spent the day getting familiar with TypeScript and its syntax

Learned about the benefits of using TypeScript with React and how it can improve code quality and maintainability

Created a basic React application with TypeScript and configured the development environment

Day 2:

Continued working with TypeScript and React, focusing on the use of interfaces and type annotations to define data types and function signatures

Reviewed best practices for using TypeScript in React projects and common mistakes to avoid

Day 3:

Moved onto Next.js, a popular React-based framework for building server-side rendered applications

Learned about the core features of Next.js such as server-side rendering, automatic code splitting, and static site generation

Set up a Next.js project and explored the directory structure and configuration options

Day 4:

Explored testing methodologies for React applications and the different testing frameworks available such as Jest and Enzyme

Learned about the different types of tests that can be performed including unit tests, integration tests, and end-to-end tests

Set up a basic testing environment for the React and Next.js application

Day 5:

Shifted focus to deployment of React applications and the different hosting options available such as AWS, Netlify, and Heroku

Reviewed the different deployment strategies and best practices for deploying a React application to production

Deployed the React and Next.js application to a hosting provider of choice

Day 6:

Wrapped up the week by reviewing the concepts learned and working on a personal project that incorporated TypeScript, Next.js, testing, and deployment

Debugged and fixed any issues encountered during the project and optimized the code for performance and maintainability