BOSSCODER ACADEMY

Frontend

MASTERY

Success in 60 Days

Ultimate Interview Roadmap and Checklist

Disclaimer

Everyone learns uniquely.

Learn Frontend in a structured manner and master it by practically applying your skills.

This Doc will help you with the same.

DAY 1-4

Understand How The Web Works

Client-Server Architecture

Understand the basic architecture of the web, where clients (browsers) make requests to servers, which respond with data.

HTTP (Hypertext Transfer Protocol) Basics

Learn about the protocol used for communication between clients and servers. Understand HTTP methods (GET, POST, etc.) and status codes.

Communication and Domain Name Systems

Understand how data is exchanged between the client and server. Explore request and response cycles, including headers and payloads.

Resources

- 🔗 How the Web Works
- 🔗 Basics of HTTP
- 🔗 Client Server Architecture
- 🔗 What is DNS? Domain Name System, DNS Server, and IP Address Concepts Explained

DAY 5-9

Authentication

Learn about user authentication mechanisms on the web, including basic authentication, token-based authentication, and OAuth.

Cookies & Sessions

Understand how cookies and sessions are used to maintain stateful interactions between clients and servers.

XSS (Cross-Site Scripting)

Learn about XSS attacks, where malicious scripts are injected into web pages.

CORS (Cross-Origin Resource Sharing)

Explore CORS, a security feature implemented by web browsers to control cross-origin requests.

CSP (Content Security Policy)

Learn about CSP, a security standard that helps prevent various types of attacks, including XSS.

Caching & Compression

Explore techniques for caching content on the client and server sides.

📁 Resources

🔗 SOP, CORS, CSRF and XSS simply explained with examples | by Jun Zhao | Medium

🔗 Content Security Policy (CSP) - HTTP | MDN

🔗 HTTP caching

🔗 Using HTTP cookies

🔗 Compression in HTTP

DAY 10-15

Document Structure

- **DOCTYPE Declaration:** Specifies HTML version.
- **Head & Title:** `<head>` contains metadata; `<title>` defines document title.

Element Types

- **Block & Inline Elements**

Tags & Attributes

Importing Resources

- `<script>` for JavaScript.
- `<link>` for stylesheets.

Content Organization

- **Headings:** Hierarchy from `<h1>` to `<h6>` for content organization.
- **Lists:** Unordered `<ul>`, Ordered `<ol>` for structuring content.
- **Tables:** `<table>`, `<tr>`, `<th>`, `<td>` for data organization.

Interactivity

- **Anchors & Navigation:** `<a>` for hyperlinks, linking resources.
- **Forms & Inputs:** `<form>`, `<input>` (text, password, checkbox, radio, etc.).

Multimedia

- **Images:** `<img>` tag with attributes like src, alt, width.

Semantic Elements

- `<header>`, `<nav>`, `<section>`, `<article>`, `<footer>` for meaningful structure.

Client-Side Storage

- **Local & Session Storage:** Using `localStorage` and `sessionStorage` APIs.

📁 Resources

🔗 The HTML Handbook – Learn HTML for Beginners

🔗 HTML Tutorial

🔗 HTML

🔗 HTML Basics - GeeksforGeeks

🔗 Learn HTML Basics for Beginners in Just 15 Minutes

DAY 16-21

Basic CSS

- Explore basic CSS properties such as `color`, `background`, `margin`, `padding`, `border`, and `font-size`.

Selectors

- Understand how selectors target HTML elements for styling. Learn about basic selectors such as type, class, and ID selectors.

Box Model

- Learn about the CSS box model, which includes content, padding, border, and margin.

Fonts & Typography

- Explore properties like `font-family`, `font-size`, `font-weight`, and `line-height`.

Positioning

- Understand the different positioning schemes, including `static`, `relative`, `absolute`, and `fixed`.

Units (Absolute + Relative)

- Explore different units in CSS, including absolute units like pixels and relative units like percentages and ems.

Float, Display & Flex

- Explore the `display` property and its values, including `block`, `inline`, `inline-block`, and `none`. Understand the float property and how it affects the layout of elements.

Resources

🔗 CSS Tutorial

🔗 Learn to style HTML using CSS - Learn web development | MDN

🔗 HTML & CSS Full Course - Beginner to Pro

🔗 Learn CSS | web.dev

🔗 CSS-Tricks

DAY 22-27

Javascript

Primitives Scopes & Hoisting

- Understand JavaScript primitive data types, including numbers, strings, booleans, null, undefined, and symbols.
- Explore the concepts of variable scope and hoisting in JavaScript.

Variables (var, let, const)

- Explore variable declaration and assignment using `var`, `let`, and `const`.

Operators

- Learn about operators in JavaScript, including arithmetic, comparison, logical, assignment, and bitwise operators.

Type Conversions

- Explore type coercion and conversion in JavaScript.

Arrays + Methods

- Understand how to work with arrays in JavaScript. Learn about array methods such as `push`, `pop`, `shift`, `unshift`, `slice`, `splice`, and others.

Objects + Methods

- Explore JavaScript objects, including object creation, properties, and methods. Learn about object manipulation and access using dot notation and bracket notation.

Functions + Arrow Functions

- Understand function declaration and expression syntax. Explore function scope, parameters, and returns.

Timeout & Interval

- Learn how to use `setTimeout` and `setInterval` functions to execute code asynchronously after a specified delay or at regular intervals.

📁 Resources

🔗 JavaScript Tutorial

🔗 Learn JavaScript by Building 7 Games - Full Course

🔗 JavaScript — Dynamic client-side scripting - Learn web development | MDN

🔗 Learn JavaScript - Full Course for Beginners

🔗 Learn JavaScript

DAY 28-31

Version Control

Cloning

- Learn to clone remote repositories to create a local copy.

Committing Changes

- Make changes and commit them to your local Git repository.

Syncing with Remote

- Understand pulling (fetching + merging) and pushing changes.

Branching

- Create, list, switch, and delete branches.

Merging & Rebase

- Merge changes between branches or rebase to incorporate changes.

Workflow Strategies

- Explore common workflows like feature branching and Gitflow.

Advanced Operations

- Delve into specific actions like cherry-picking, stashing, squashing, and tagging.

📁 Resources

🔗 Learn Git and Version Control in an Hour

🔗 Git Tutorial for Beginners: Learn Git in 1 Hour

🔗 Git for Professionals Tutorial - Tools & Concepts for Mastering Version Control with Git

🔗 Introduction to version control with Git - Training | Microsoft Learn

🔗 What is Git? A Beginner's Guide to Git Version Control

DAY 32-36

Components & JSX

Understand the basics of React components and JSX (JavaScript XML), a syntax extension for JavaScript that looks similar to XML/HTML.

State & Props

Learn about React component state and props. Understand how state and props are used to manage and pass data within a React application.

Class Components & Lifecycle

Explore class components in React and understand the component lifecycle methods, such as componentDidMount and componentDidUpdate.

Functional Components

Learn about functional components, a simpler way to define components using JavaScript functions.

📁 Resources

- 🔗Quick Start – React
- 🔗What Is React? [Easily Explained]
- 🔗Setting up the development environment · React Native
- 🔗React for Beginners – A React.js Handbook for Front End Developers
- 🔗React JSX
- 🔗React Components
- 🔗ReactJS State vs Props - GeeksforGeeks

DAY 37-43

Hooks Basics

Explore hooks, focusing on useState, useEffect, and useRef.

Advanced Hooks

Dive into advanced hooks and custom hook creation.

Event Handling

Explore event handling in React and data passing to event handlers.

Forms & Validation

Work with forms, controlled components, and validation

Resources

🔗 Built-in React DOM Hooks

🔗 React Hooks Course - All React Hooks Explained

🔗 Reusing Logic with Custom Hooks – React

🔗 Responding to Events – React

🔗 React Events

🔗 React Forms

🔗 How to Validate Forms in React – A Step-By-Step Tutorial for Beginners

🔗 Passing Data Deeply with Context – React

🔗 useContext – React

DAY 43-48

Context API

Manage global state in React, avoiding prop drilling in nested components.

Styling in React

Apply CSS in React, enhance with SASS, or use Styled Components for component-based styling.

React Router

Enable navigation and routing in React applications for seamless page transitions.

📁 Resources

🔗 Styling React Using CSS

🔗 Adding a Stylesheet | Create React App

🔗 Adding a Sass Stylesheet | Create React App

🔗 How to Style Your React Apps with CSS Like a Pro

🔗 Tutorial v6.21.1 | React Router

🔗 ReactJS Router - GeeksforGeeks

DAY 49-50

Deploying a React Application

Topic: Deploying a React Application

📁 Resources

🔗https://create-react-app.dev/docs/deployment

🔗https://create-react-app.dev/docs/proxying-api-requests-in-development

🔗https://www.netlify.com/with/react/

DAY 50-60

Work on a project!

Now that you have a solid understanding of React concepts and have explored various app ideas, it's time to put your knowledge into action by working on your React project.

Ideas of Apps you can work on:

1. To-Do List/Notes App:
2. Weather App
3. Recipe Finder
4. Expense Tracker

Steps you can follow to build your app:

Project Setup and Planning

1. Set up your project environment. If you're using Create React App or a similar tool, initialize your project.
2. Plan your project's structure and components. Sketch out a rough design or wireframe for your app.
3. Create a GitHub repository to version control your project.

Building Components and State Management

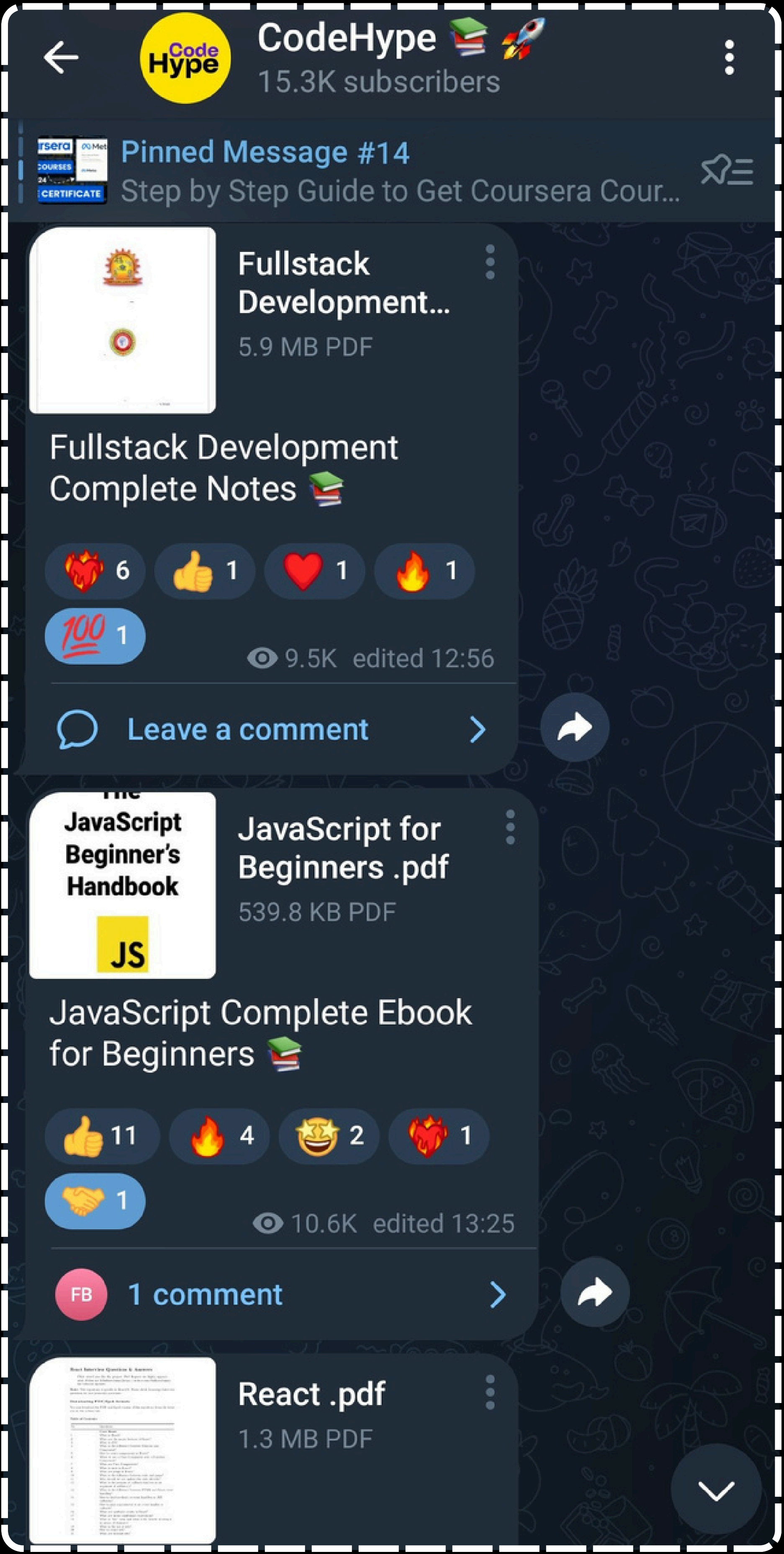
1. Build the foundational components of your app based on your project plan.
2. Implement component logic, including state management using React's state and props as appropriate.
3. Set up any forms or user input components required for your app.

Implement Features and Functionality

1. Implement the core features and functionality of your app. This may involve fetching data from APIs, handling user interactions, and managing application state.
2. Test your app's features as you go, fixing any bugs or issues that arise.

Styling, Testing, and Deployment

1. Apply CSS or styling frameworks to make your app visually appealing and user-friendly.
2. Conduct thorough testing and debugging to ensure your app is robust and functional.
3. Once you're confident in your project's functionality and design, deploy it to a hosting platform of your choice. Document the deployment process for future reference.


CodeHype 📚🚀
15.3K subscribers

Pinned Message #14
Step by Step Guide to Get Coursera Cour...

Fullstack Development...
5.9 MB PDF

Fullstack Development Complete Notes 📚

❤️‍🔥 6 👍 1 ❤️ 1 🔥 1 💯 1

9.5K edited 12:56

Leave a comment

JavaScript for Beginners .pdf
539.8 KB PDF

JavaScript Complete Ebook for Beginners 📚

👍 11 🔥 4 🤩 2 ❤️‍🔥 1 🤝 1

10.6K edited 13:25

1 comment

React .pdf
1.3 MB PDF