Bootcamp Complete Roadmap

🔗 Daywise Roadmap

Episode 1 - Stage

1. Starting with HTML

- Understanding HTML and its use Cases.
- Creating first HTML page in VS Code
- Understand HTML Structure
- Understanding Tags and building simple HTML page - doctype , html , head , title , body
- Working with text elements - h tags , p tag , br tag , a tag , span , code , pre
- Working with HTML Lists(Ordered & Unordered lists) - ol , ul , li
- Understanding Concept of nested elements in HTML
- Working with Media Tags - img , video , audio
- HTML attributes - href , target , alt , src , width , height ,
- Navigating between pages

2. More on HTML

- Understanding semantic tags - article , section , main , aside , form , footer , header , details , figure
- Differentiating between block and inline elements
- Text formatting tags in HTML - b , string , i , small , ins , sub , sup , del , mark
- Working with HTML tables - table , td , tr , th

3. HTML Forms and Inputs

- What is Form and why its important
- Creating a simple Form with tags - form , input , textarea , select , button , label
- Types of input fields - checkbox, text , color , file , tel , date , number , radio , submit , range
- Attributes of Form Elements - method, actions, target, novalidate, enctype, name, required, placeholder

4. Media Tags in HTML

- Understanding with audio and video Tags
- Attributes if media tags - src, width, height, alt, muted, loop, autoplay, controls, media
- Using source element for alternative media files

5. Basics of CSS (Cascading Style Sheet)

- Introduction to CSS and Why it is important
- Understanding Syntax, Selectors and comments in CSS
- Adding CSS to HTML Page - Inline, Internal, External
- Understanding difference between selectors - class , id , element
- Understanding precedence of selectors
- How to style text using CSS - font family, font style , font weight , line-height , text-decoration , text-align , text-transform , letter-spacing , word-spacing , text-shadow

6. Styling With CSS

- Working with colors in CSS - name , rgb , etc.
- Working with css units - % , px , rem , em , vw , vh , min , max
- Working with borders and border styling
- Working with box properties - margin , padding , box-sizing , height , width
- Understanding Background properties - background-size , background-attachment , background-image , background-repeat , background-position , linear-gradient
- Implementing shadow property.

7. More about CSS

- Applying display properties - inline , grid , flex , none , inline-block , etc.
- Introduction to FlexBox for aligning and structure - flex-direction , order , flex-wrap , flex-grow , flex-shrink , justify-content , align-items , align-content , align-self , flex-basis , shorthand properties of flex
- Understanding Flex Grid for making grids using CSS.
- Working with positional properties - absolute , relative , static , sticky , fixed.
- Understanding Overflow - visible , hidden , scroll.

- Working with Grouping Selectors.
- Why we use Nested Selectors.

8. Interesting things about CSS ✌️

- Applying pseudo classes and Pseudo Elements [hover , focus , after , before , active] .
- Learning CSS Transitions (properties, duration, timing functions, delays).
- Creating with Transform (translate, rotate, scale, skew , transform , rotate).
- Working with 3D Transform (translate3d() , translateZ() , scale3d() , scaleZ(), rotate3d() , rotateZ() .
- Understanding CSS Animation (@keyframes).

9. Responsive with CSS 🖥️

- Difference Between Mobile-first and Desktop first Website(mobile-first vs desktop first).
- Measurement units for Responsive Design - px(pixel), in(inch), mm(millimetre), %, rem
- Using Viewport meta element for Responsive.
- Setting up Images and Typography for Responsiveness.
- What are Media queries [@media , max-width , min-width].
- Using Different function of CSS [clamp , max , min].
- Understand HTML structure for Responsive Design.

10. Working With SASS (SASSY) my favorite 🤩

- What is SASS? Variables ,Nesting ,Mixins ,Functions and Operators .
- Setting up environment for SCSS.
- SCSS or SASS? and Setting Up SCSS.

11. Basics of Javascript with ES6+ Features 🚀

- Introduction to JavaScript, Why it is Important! and What can it do for you?
- How to link javascript files using script-tag .
- Running JavaScript in the Browser Console .
- Variables and Keywords in Javascript [var , let , const].

- Logging with javascript - [console.log() , console.info() , console.warn() , prompt , alert]
- Working with String in JS and there -[splice , slice , template string , split , replace , includes]
- What are Statement and Semicolons in JS
- How to add Comments in JavaScript
- What are Expression in Js and difference between expression and statement
- JavaScript Data Types - [float , number , string , boolean , null , array , object , Symbol , Undefined]
- Some Important Values - [undefined , null , NaN , Infinity]
- Relative and Primitive Data Type in JavaScript
- Basic Operators(Arithmetic, Assignment, Increment, Decrement, Comparison, Logical, Bitwise) - [+ , , , / ,++ , , == , === , != ,and more]
- Variable hoisting in JavaScript

12 . Loops & Conditionals in Javascript

- Understanding Condition Operator in Javascript - [if , else , if-else , else-if , Ternary Operator , switch]
- for Loop in JavaScript
- while Loop in JavaScript
- do...while in JavaScript
- forEach in JavaScript
- for in Loop in JavaScript
- for of Loop in JavaScript
- Recursion in JavaScript
- Loop control statements - [break , continue]

13. Functions in JavaScript

- Understanding Function in JavaScript and why its widely used - [parameters , arguments , rest parameters , hoisting , Variable Hoisting , Function Hoisting]
- Parameters in JavaScript - [required , destructured , rest , default]
- Arguments in JavaScript - [positional , default , spread]

- Classic Function, Nested Function(function within function), Scope Chain in Javascript.
- Understanding Immediately Invoked Function Expression(IIFE).
- More Functions in JavaScript - [Arrow Function , Fat Arrow , Anonymous , Higher Order , Callback , First Class , Pure Function , Impure Function]
- Understanding Scoping in JS - [Global scope , Function scope]
- Understanding Closures, Scoping Rule .

14. Arrays and Objects in JavaScript

- What are Arrays in JavaScript and how to Create an Array.
- Understand How to Accessing Elements in Array.
- Functions on Arrays - [push , pop , shift , unshift , indexOf , array destructuring , filter , some , map , reduce , spread operator , slice , reverse , sort , join , toString]
- Iterating Over Arrays using - [For Loop , forEach]
- Understanding What are Objects in JavaScript - [key-value pair]
- Creating Objects, Accessing Properties, Deleting Property and Nested Objects.
- Recognise How Objects Are Stored, Traverse Keys of an Object, Array as Object.
- Timing Events - setTimeout() , setInterval() , clearTimeout() , clearInterval()
- Operation in Objects - [freeze , seal , destructuring , object methods , this keyword]

15. Document Object Model Manipulation

- Introduction to DOM in JavaScript
- Understanding DOM Structure and Tree - [nodes , elements , document]
- Fetching Elements in DOM - [document.getElementById, document.getElementsByTagName, document.getElementsByClassName, document.querySelectorAll, document.querySelector]
- DOM Tree Traversal - [parentNode , childNodes , firstChild , nextSibling]
- Manipulating DOM Element in JavaScript - [innerHTML , textContent , setAttribute, getAttribute , style property , classList]
- Create and Removing DOM Elements - [createElement() , appendChild() , insertBefore() , removeChild()]

16. Event Handeling in JavaScript

- Event Handling in JavaScript - [addEventListner(), event bubbling, event.target]
- Understanding Scroll Events, Mouse Events, Key Events and Strict Mode.
- Working with Forms and Input Elements [Accessing Form Data , Validating Forms , preventDefault() ,onsubmit, onchange]
- Working with Classes ****Adding, Removing , Toggling (classList methods)
- Browser Events - [DOMContentLoaded , load , resize , scroll]

17. Using Browser Functionalities in JavaScript

- Browser Object Model - [window , navigator , history , location , document]
- Window Object - [window.location , window.history]
- Working with Storage - [Local Storage , Session Storage , Cookies]
- Web APIs in DOM - [Fetch API]

18. Object Oriented Concepts in JavaScripts

- Introduction to OOPS in JavaScript
- Understanding classes and objects in JavaScript
- Understanding Constructor and Prototypes - [this keyword , call , apply , bind]
- More Topics in OOPS - [class expression , hoisting , inheritence , getter & setter]

19. Asynchronous Programming JavaScript

- Introduction to Asynchrony in JavaScript.
- Introduction to callbacks and Problems in Callbacks
- Understanding promises - pending , resolved , rejected
- How to prevent callback hell using async & await .
- setInterval & setTimeout in JavaScript

20. Error Handling in JavaScript

- Introduction to Error Handling
- Common types of errors in JavaScript - [Syntax errors , Runtime errors , Logical errors]
- Understanding the Error object - [message , name , stack]
- Handling exceptions using try-catch , try-catch-finally

- How to Throw Errors in JavaScript
- How to create custom error in JavaScript
- Error Handling in Asynchronous Code

21. Kuch Baatein Advance JavaScript Pr ⚙️

- Throttling and Debouncing uses in JavaScript
- JSON Handeling and JavaScript - [JSON.parse() , JSON.stringify()]

22. Git and Github

- What is Git and Github?
- Concepts - Git commits , Understanding branches, Making branches, merging branches, conflict in branches, understanding workflow, pushing to GitHub.
- How to use GitHub with team members, forking, PR(pull requests) open source contribution, workflow with large teams.

Episode 2 - Commit

1. Introduction of React 🪫

- What is React, and Why Use It?
- What are Components and types of Components - class component , function components
- Understanding Single Page Applications (SPAs), Single Page Applications Vs Multi-Page Applications.
- Difference between Real DOM and Virtual DOM
- NPM Basics | Installing Packages.
- How does updates work in React? and More ES6+ features like Import & Exports ,
- Difference Between React and Other Frameworks (Angular, Vue).
- Learning Some Basic Terminal Commands - pwd , ls , cd , clear
- Setting Up React Environment with nodejs.
- Install React-Vite Boilerplate and Installing React Developer Tools.
- Understanding JSX or JavaScript XML and Its Importance - Fragments , Components Naming .
- Creating and Understanding best practices for Components in React.

2. Styling in React 🐼

- Different Styling Approaches.
- Importance of component-based styling. Inline Styles ,CSS Modules
- Dynamic Styling Based on Props or State.
- Responsive Design in React
- Media queries with CSS and styled-components.

3. React Basics 🔦

- Create Components with functions.
- Importing css file/stylesheet in react and Adding a CSS Modules Stylesheet - Styled Components , Dynamic styling with styled-components .
- Creating a state and Manage State using setState - What is State? , setState , useState .
- Creating Parameterised Function Components in React.
- React Props: Passing Data to Components.
- Function chaining in React and Conditional Rendering - Rendering Array Data via map , Eliminating Array Data via filter.

4. More on React 📽️

- Higher Order Components in React.
- Reusing Components, Lists and Keys in React.
- Sharing Data with child components : Props Drilling .
- Rendering a List, Mapping and Component Lifecycle - Mounting , Updating , Unmounting.
- Understanding React Component Lifecycle .
- Different Lifecycle Methods like componentDidMount .

5. Useful Hooks in React 🪝

- Understanding React Hooks
- Rules of hooks.
- Commonly Used Hooks:
    - useState

    - useEffect
    - useContext
    - useRef
    - useCallback
    - useMemo
- Custom Hooks: When and How to Create Them
- Understanding and Applying Context API.

6. Navigation in the React with React Router 🚧

- Introduction to React Router.
- Setting Up and Configuring React Router setup of react-router-dom .
- Navigating Between Pages with .
- Passing Data while Navigating
- Dynamic Routing
- URL Parameters and Query Strings
- Nested Routes
- Programmatic Navigation Using useNavigate.
- Handling 404 Pages : fallback route for unmatched paths, Customizing the "Page Not Found" experience.

7. State Management Using Redux. 🏪

- Introduction to Redux , What is redux?, When and Why use redux?
- Understand Principles of Redux and Redux Flow.
- Understanding State Management in React using Redux.
- Why Use State Management Libraries?
- Why Redux need reducers to be pure functions.
- Redux Basics: Actions , Reducers , Store , Currying , Middleware , Async Actions: Thunk
- Connecting Redux to React Components with react-redux.
- Introduction to Redux Toolkit.

- Alternatives: Recoil, Zustand, or MobX.

8. Form controls in the React : Building Dynamic Forms 📋

- Introduction to Forms in React.
- Building Basic Forms.
- Creating form elements like input, textarea, select, etc.
- Two way binding with react [input , textarea].
- Handling Form Events [onChange , onSubmit , event.preventDefault()].
- Validation in React Forms : client-side form validation.
- Integrating Forms with APIs.
- Sending form data to a backend using fetch or axios.
- Handling loading states and success/error feedback.

9. Performance Optimization 🏎️

- Code Splitting with React Lazy and Suspense
- Avoids redundant calculations by caching Using Memoization Techniques:
    - React.memo
    - useMemo
    - useCallback
- Avoiding Re-Renders using useState ,
- Optimizing Component Structure
- Performance Profiling Tools using Chrome DevTools , Lighthouse , Web Vitals ,Largest Contentful Paint (LCP), First Input Delay (FID)

10. Deploying React projects 🚨

- Preparing a React App for Production .
- Building React Applications.
- Environment Variables in React.
- Deployment Platforms: Netlify ,Vercel , GitHub Pages ,

11. Real-World Project with React 👷

- Building a Complete React Project

- Combining All Concepts (Routing, State Management, API, etc.)
- Styling and Responsiveness ,
- Optimizing and Deploying the Project.

12. Animations 🔥

- Animation and Transitions Using libraries like framer-motion or gsap for advanced animations.

13. Basic SEO Principles

- On-Page Optimization in SEO.
- Guide to SEO Meta Tags.
- Image SEO Best Practices.
- Internal Link Building SEO.
- Create An SEO Sitemap For a Website.

14. Three.js and React Three-Fiber

- Understanding what is Scene.
- Using 3d models for animation.
- Controlling view with Orbit controls.
- Applying Lights inside the scene.
- Understanding different types of Cameras.
- Animating the mesh with GSAP or Framer motion.
- Different types Geometries.
- Using different Materials for animation.

15. Introduction to TypeScript

- Introduction to TypeScript and its role in modern frontend development
- Why TypeScript is important for React applications
- Difference between JavaScript and TypeScript in real-world projects
- Advantages of static typing in UI-based applications
- Understanding how TypeScript improves scalability and maintainability
- TypeScript adoption in industry and enterprise React applications

---

16. Setting Up TypeScript Environment

- Installing and configuring TypeScript in a project
- Understanding TypeScript compiler workflow
- Introduction to tsconfig.json
- Understanding compiler options and strictness levels
- Target environments and module systems
- Integrating TypeScript with modern build tools
- TypeScript with React project setup

---

17. TypeScript Basic Types & Data Structures

- Understanding TypeScript type system
- Primitive types in TypeScript
- Difference between any, unknown, never, and void
- Nullable types and undefined handling
- Arrays and typed collections
- Tuples and fixed-length data structures
- Read-only types and immutability concepts

---

18. Object Typing in TypeScript

- Typing objects in TypeScript
- Required vs optional properties
- Read-only properties
- Nested object typing
- Index signatures
- Excess property checks
- Structural typing concept

---

19. Advanced Type Definitions

- Type aliases and their purpose
- Union types and use cases
- Intersection types and composition
- Literal types and value constraints
- Optional chaining and null safety concepts
- Type narrowing and control flow analysis

---

20. Interfaces in TypeScript

- Introduction to interfaces
- Defining object contracts using interfaces
- Interface extension and inheritance
- Declaration merging
- Interfaces vs type aliases
- Best practices for using interfaces in React projects

---

21. Enums and Constants Management

- Introduction to enums
- Numeric vs string enums
- Use cases for enums in frontend applications
- Alternatives to enums
- Managing constants in TypeScript projects

---

22. Generics in TypeScript

- Understanding generic types
- Why generics are important
- Generic functions and reusability
- Generic interfaces and type safety

- Generic constraints
- Practical use cases of generics in React applications

---

23. TypeScript with React Fundamentals ⚛️

- Understanding how TypeScript works with React
- Difference between .ts and .tsx files
- JSX typing fundamentals
- Typing functional components
- Understanding React typings ecosystem
- React + TypeScript best practices

---

24. Props Typing in React with TypeScript

- Understanding props contracts
- Required and optional props
- Read-only props
- Children prop typing
- Reusable prop types
- Handling complex prop structures

---

25. State Management Typing

- Typing component state
- Primitive vs object-based state typing
- Nullable state handling
- Derived state typing
- State updates and immutability
- Best practices for typed state

---

26. Event Handling & Forms Typing

- Typing React events
- Mouse events and keyboard events
- Form events and submission handling
- Input and controlled form typing
- Validation and error state typing
- Preventing common typing mistakes in forms

---

27. Context API with TypeScript

- Introduction to Context API with TypeScript
- Typing context values
- Provider and consumer typing
- Handling default values safely
- Avoiding undefined context access
- Scalable context architecture patterns

---

28. Redux with TypeScript

- Why TypeScript is important for global state
- Typing Redux store
- Typing reducers and actions
- Typed dispatch and selectors
- Redux Toolkit with TypeScript
- Best practices for scalable state management

---

29. API Integration & Data Typing

- Defining API data contracts
- Typing API responses
- Handling loading, success, and error states
- Error response typing

- Pagination and metadata typing
- Maintaining consistency across frontend and backend

---

30. TypeScript Best Practices 🧠

- Understanding strict mode in TypeScript
- Benefits of strict type checking
- Avoiding misuse of any
- Using safer alternatives to any
- Writing clean and predictable type definitions
- Type reuse and centralization strategies

---

31. Project Structure & Architecture

- Organizing React + TypeScript projects
- Feature-based folder structure
- Managing shared types
- Separation of concerns
- Scalable architecture patterns
- Industry-standard project layouts

---

32. Common TypeScript Errors & Debugging

- Understanding TypeScript compiler errors
- Debugging type mismatches
- Handling null and undefined safely
- Working with third-party library typings
- Resolving common React + TypeScript issues

Episode 3 - Push

1. Starting with Node.js - The Beginning 🏁

- Introduction to Node.js and Getting Our Tools - Node.js LTS , Postman , Editor
- Setting up the Tools for our Environments
- Running script with nodejs - “Namaste Duniya”
- NPM Basics | Installing Packages.
- Creating and Managing package.json.

2. Creating Server - Writing Our First Server 📱

- What is Server and how it works?
- Setting Up Our First Node.js Server using HTTP
- Serving A Response to the Browser and Understanding Responses.
- Routing in HTTP Servers.
- Understanding Status Code - 1XX , 2XX , 3XX ,404 - Not Found , 200 - success , 500 - Internal Server error , 422 - Invalid Input , 403 - the client does not have access rights to the content , etc.
- Installing Nodemon for Automatic Server Restarts.

3. Some talk on Different Architectures 🏯

- Different Architectures in backend like MVC and SOA.
- Understanding MVC Architecture Model , View ,Control.
- MVC in the context of REST APIs.

4. Web Framework - Express.js 🚀

- what is Express.js and why to use it.
- Setting Up Express Server .
- Returning Response from the server.
- Using Query Parameters and URL Parameters.
- HTTP Request - Some Important part of requests , Different Types of Requests - Get , Post , PUT , Patch , Delete.
- Serving Static Files with express.static() .

5. Template Engine - EJS 🚜

- What is Template Engine and What is the use of Template Engine.
- Template Engine Option - Handlebars , EJS , Pug , jade but We’ll use EJS .

- Setting Up Template Engine - Installed EJS template engine.
- Rendering Our First Page using EJS and Some important syntax - <%= %> , <% %> , <%- %>.
- Loop statement, Conditional statement and Locals in views - EJS.
- Accessing the Static Files Inside EJS file.

6. Middleware in Express.js (one of my favorite) 🐵

- Understanding the middleware in express.
- Implementing middleware with express.
- Different types of middleware : builtIn middleware , third-party middleware ,custom middleware .
- Different level of middleware : Application-Level , Router-Level .
- Handeling Errors and Security with middleware : Error-Handling , Helmet , CORS.

7. Handling file with Express 📁

- Understand Multer and its usecase?
- Uploading file with multer.
- Understanding Memory and Disk Storage.
- Accessing uploaded file req.file.
- Working with express.static.
- Using Cloudinary or Imagekit for Real-time media processing APIs and Digital Asset Management.

8. Beginning of Database Basics (Bohot km theory) 🗄️

- Relational and non-relational Databases : mongodb & mysql .
- What is MongoDB? Why Use It?
- Installing Compass and Understand how to access DB using terminal.
- Setting Up MongoDB Locally and in the Cloud.
- Understanding Datatypes Collections and Documents.
- Connecting MongoDB to Node.js with Mongoose .
- Database Relations - One to One , One to Many OR Many to One , Many to Many , Polymorphic .

- Handling Relationships with Mongoose (populate).

9. API Development(REST) ⛓

- What is a REST API?
- Versioning in RESTful APIs - /v1/
- Using Postman for API Testing and developing - Send Requests , Save Collections , Write Tests .
- Understanding and Working With Status code , 2xx (Success) , 4xx (Client Errors) , 5xx (Server Errors) .
- Validating API Inputs Using libraries like express-validator or Sanitization .
- Security Handling - Rate Limiting with express-rate-limit ,XSS Attack , CSRF Attack , DOS Attack.

10. Database Optimization for Fast response 🧘

- Indexing for Performance with MongoDB :- Single-Field Indexes , Compound Indexes , Text Indexes ,Wildcard Indexes.
- Best practice with Indexing explain().
- Learning MongoDB Aggregation.
- Comparison Operators - [$eq , $ne , $lt , $gt , $lte , $gte , $in , $nin]
- Logical Operators - [$not , $and , $or and $nor]
- Array[$pop, $pull, $push and $addToSet]
- Stages in Aggregation pipeline :- $match , $group ,$project ,$sort ,$lookup.
- Creating Database on Local and Atlas
- Creating parallel pipeline with $facet .
- Learning MongoDB Operators.
- Understanding Different types of Operators :- Comparison ,Regex ,Update ,Aggregation.

11. Logging Backend : Express.js

- Why is Logging Important?
- Setting Up Logging with Libraries winstone ,Pino ,Morgan .
- Different mode of morgan ,**dev ,short ,tiny .**

- Error Handling and Logging.

12. Production Wala Project Structure and Configuration 🗼

- Understanding the Basic Structure of application.
- Learning File Naming Conventions, Git Configuration,
- Understanding Important Folders :- src/ ,config/ ,routes/ ,utils/ .
- Role of package.json , ENV and .gitignore .
- Production Environment - PM2 , Error & Response Handling Configuration , CORS Configuration , async-handler.js.
- Using and Configuring ESLint and Prettier for code formatting.
- Testing APIs using Postman.

13. Authentication and Authorization 🪪

- Difference Between Authentication & Authorization
- Working with Passwords and Authentication - Cookie Authentication , OAuth Authentication
- Understanding Session and Token Authentication.
- Implementing JWT Authentication :- jsonwebtoken JWT_SECRET.
- Securing user password with bcrypt hashing salt.
- Role-Based Access Control (RBAC).
- Authenticating user with Express middleware .
- Understanding Passport.js and its usecase?
- Glancing through and Installing Passport.js
- Setting up Passport.js - passport-local, local-strategy , google-OAuth
- express-sessions and using passport for authentication.

14. Working Real time communication : WebSockets and socket.io 💬

- Understanding WebSockets protocol for realtime applications?
- Learning handshake ,Persistent connection ,Bidirectional communication ,HTTP polling .
- Understanding difference between WebSocket Vs Socket.io.
- Working with socket.io for realtime applications.

- Understanding usage ofRooms in Socket.io.
- Understanding Middleware in Socket.io.

15. Working With Caching - Local and Redis 🍄

- What is Caching and How to cache data locally?
- What is Redis?
- Why Use Redis for Caching?
- Implementing Redis Caching in Node.js.
- Advanced Redis Features TTL ,Complex Data Structures , Pub/Sub.

16. Error handling in express 🛑

- Basic Error Handling in Express next() .
- Catching Specific Errorstry &catch .
- Creating Util Class for Error Handling.

17. Testing Tools 🛠️

- Understanding Unit-Testing With Jest.
- Cross Browser Testing and Why Is It Performed?
- What Is Web Testing? and How to Test a Website.

18 Next.js Fundamentals

- What is Next.js and why it exists
- Problems Next.js solves in React applications
- Understanding client-side vs server-side rendering
- Rendering strategies – [CSR, SSR, SSG]
- When to use which rendering method
- Overview of Pages Router vs App Router
- Creating a Next.js application using create-next-app
- Understanding Next.js project structure

---

19 Routing in Next.js

- File-based routing system in Next.js

- Static routes
- Dynamic routes – [[id]]
- Nested routing structure
- Route grouping concepts (overview)
- Navigation using next/link
- Programmatic routing
- Handling 404 and error routes

---

20 Data Fetching in Next.js

- Data fetching strategies in Next.js
- Static data fetching – getStaticProps
- Server-side data fetching – getServerSideProps
- Dynamic static generation – getStaticPaths
- Incremental Static Regeneration (ISR)
- Choosing the right data fetching method
- Data fetching lifecycle in Next.js

---

21 Styling & Assets in Next.js

- Global CSS handling
- Component-level styling with CSS Modules
- Integrating Tailwind CSS with Next.js
- Asset management in Next.js
- Image optimization using next/image
- Font optimization and font loading
- Metadata handling
- SEO optimization in Next.js applications

---

22 API Routes

- Introduction to API routes in Next.js
- Using Next.js as a full stack framework
- Writing backend logic inside Next.js
- Handling HTTP methods in API routes
- Connecting MongoDB with Next.js
- Creating authentication APIs
- Structuring API routes for scalability

---

21 Next.js with TypeScript

- Using TypeScript with Next.js
- Typed pages and components
- Typed layouts
- Typing props and params
- Typed API routes
- Type safety in full stack Next.js apps

---

22 Middleware & Authentication

- Introduction to middleware in Next.js
- Request and response interception
- Protecting routes using middleware
- Authentication flow in Next.js
- Auth using cookies
- JWT-based authentication
- Role-based access control (overview)

---

23 Deployment & Production

- Deploying Next.js applications
- Deployment using Vercel

- Environment variables management
- Production build optimization
- Performance optimization techniques
- Monitoring and debugging production issues

🧩 **Episode 4 – Merge**

---

1. Generative AI and Applications 🤖

🧠 Introduction to Generative AI

- What is **Generative AI** and how it works (ML, DL, and LLM concepts).
- Core models behind Generative AI – Transformers, Diffusion Models, GANs.
- Understanding use-cases and limitations (bias, hallucination, accuracy).

💡 Building Real-World AI Applications

- Building an **Authentication System** with Generative AI.
- Creating **Social Media Automation Tools** (AI post generator, auto-scheduler).
- Developing **Content Generation** Projects (blog writer, idea generator).
- AI-powered Resume Reviewer (using ChatGPT or Gemini API).
- Virtual Interview Assistant using **LLM APIs + Voice/Prompt Engineering**.

🔗 LangChain & Agentic Systems

- Introduction to **LangChain** and its features.
- Working with **LLM Chains**, **Memory**, and **Tools**.
- Building **AI Agents** that can browse, plan, and execute tasks.
- Exploring **Agentic-AI Applications** (multi-step reasoning systems).
- Understanding and implementing **Multi-Agent Systems**.
- Overview of **MCP Server** (Model Context Protocol) and its role in LLM interoperability.

---

2. Progressive Web App (PWA) Development 🛜

🌍 Understanding PWAs

- What are **Progressive Web Apps** and why they matter.
- Benefits: Offline access, app-like experience, installability.
- Real-world examples: Twitter Lite, Starbucks PWA.

⚙️ Core Components of a PWA

- **Service Workers:** Role, lifecycle, and registration.
    - Lifecycle stages: Install, Activate, Fetch.
    - Handling caching and background sync.
- **Manifest File:** Purpose and structure.
    - Key properties – name, short_name, icons, start_url, theme_color, background_color.
- **DevTools for Debugging:** Application tab, cache inspection, and audit tools.

🚀 Performance Optimization

- Implementing **Lazy Loading** and **Code Splitting**.
- Testing PWA with **Lighthouse**.
- Advanced caching strategies (e.g., Cache First, Network First, Stale-While-Revalidate).
- Techniques for faster load time and smoother offline experiences.

---

3. DevOps Fundamentals 🐳 (Docker, Kubernetes & Terraform)

🔧 Introduction to DevOps

- What is **DevOps** and why it's essential.
- Stages: Plan → Build → Test → Release → Deploy → Monitor → Feedback.
- Understanding **CI/CD Pipelines** and automation tools (GitHub Actions, Jenkins).

🐳 Docker – Containerization

- What are **containers** and how they differ from **virtual machines**.
- Core Docker concepts: Image, Container, Volume, Network.
- Writing a simple **Dockerfile** for a Node.js app.
- Using **Docker Compose** for multi-container environments.

- Running and managing containers efficiently.

☸️ Kubernetes – Orchestration

- What is **Kubernetes (K8s)** and why it's used for scaling microservices.
- Core components: Pod, Deployment, ReplicaSet, Service, Ingress.
- Managing workloads and ensuring high availability.
- Setting up **Minikube** or **Docker Desktop Kubernetes** locally.
- Deploying a microservice on Kubernetes.
- Load balancing and auto-scaling with Kubernetes.

🌍 Terraform – Infrastructure as Code (IaC)

- What is **Terraform** and how it automates cloud infrastructure.
- Understanding **Providers**, **Resources**, and **State Management**.
- Writing basic .tf files to provision AWS EC2 instances.
- Using Terraform with **Docker** and **Kubernetes** deployments.
- Real-world scenario: Automate deployment of a Node.js + MongoDB app to AWS using Terraform.

---

4. Building Microservices with Node.js 🏘️

🧩 Understanding Microservices

- What are **Microservices**, and why they replace monolithic systems.
- Comparison: **Monolithic vs Microservices**.
- Key benefits: Scalability, modularity, independent deployment.
- Common challenges (data sharing, inter-service communication, debugging).

🏗️ Designing and Building Microservices

- Creating a **Node.js Microservice** (Express-based).
- Structuring microservices with individual package.json.
- Designing microservice architecture for a sample app (e.g., eCommerce system).

🔗 Inter-Service Communication

- Synchronous vs Asynchronous communication.
- REST APIs for service interaction.
- Using **Redis Pub/Sub** or **RabbitMQ** for Event-driven communication.
- Implementing **API Gateway** with Express.js.
    - Proxying requests
    - Rate limiting and authentication
- Dockerizing and deploying microservices.
- Overview of **Kubernetes orchestration** for microservice clusters.

---

5. Deployment ✈️

☁️ **Cloud & Infrastructure Setup**

- Deploying the project on cloud platforms.
- Using **DigitalOcean App Platform** (auto-scaling, containers, built-in load balancer).
- Exploring major providers: **AWS**, **GCP**, **Heroku**, **Azure**.

🖥️ **AWS Deep Dive**

- Launching and managing **EC2 Instances**.
- Connecting via **SSH** and setting up the environment.
- Cloning the repository and running the production build.
- Configuring **NGINX** as a reverse proxy.
- Masking the **Domain Name** to the server IP (DNS configuration).

☸️ **Kubernetes + Terraform Integration**

- Deploying microservices using **Kubernetes clusters on AWS (EKS)**.
- Using **Terraform** to automate cluster setup and deployment.
- Managing secrets and environment variables with **Kubernetes Secrets & ConfigMaps**.
- CI/CD pipeline integration: Deploy code → Build Docker image → Push to registry → Deploy via Terraform → Orchestrate with Kubernetes.

---

6. SYSTEM DESIGN 📍

---

1. System Design Basics

- Understanding System Design and why it matters
    - [scalability , performance , reliability , availability , fault tolerance]
- Core Performance Concepts
    - [latency , throughput]
- Traffic Distribution in Systems
    - [load balancing , health checks , layer 4 vs layer 7 routing]
- Caching Fundamentals
    - [client-side , server-side , database cache , cdn cache]
- Scaling Strategies
    - [vertical scaling , horizontal scaling]
- Databases (Foundational Concepts)
    - [data models , indexes , transactions , queries]
- Data Replication
    - [leader-follower , multi-leader , read replicas]
- Data Sharding
    - [shard keys , range-based , hash-based]
- Message Queues
    - [async processing , event-driven , producers , consumers , Kafka]
- Content Delivery Networks (CDN)
    - [edge caching , static assets]
- Stateless vs Stateful Systems
    - [session handling , server independence]
- High Availability Concepts
    - [redundancy , failover , auto-scaling]

- Monitoring & Observability
    - [logging , metrics , tracing]
- Data Migration
    - [schema changes , versioning , online migration , zero-downtime migration]

DSA with JavaScript

1. Conditional Statements

- Understanding Conditional Statements
- Types of Conditional Statements if , if-else , if-else if , switch
- Making decisions in a program based on inputs or variables.
- Validating user data or input forms.
- Creating interactive menus or options in applications.

2. Loops, Nested Loops, Pattern Programming

- Undertsanding the use of Loops.
- for loop.
- while loop.
- do-while loop.
- Understanding the Use of Nested Loops.
- Learning Pattern Programming - Pyramid patterns , right-angled triangles, and inverted triangles.
- Understanding Control Flow statement break and continue
- Learning how to set correct conditions to avoid getting stuck in infinite loops.
- Understand how to optimize nested loops for better performance and reduced time complexity.

3. Array

- Understanding the use of Arrays.
- Basic Manipulations - insertion , deletion , updation
- Accessing Elements in Arrays .

- Traversing Elements in Arrays .
- Array Algorithms - Two Pointer Algorithm, Rotation Algorithms , Kadane's Algorithm , etc

4. Object-Oriented Programming (OOP) in JavaScript

- Understanding Object-Oriented Programming
- Learn how to define a class for creating objects.
- Understand how to instantiate objects from a class
- Learn how the constructor() function initializes an object when it's created.
- Understand how this refers to the current object in the context.
- Use this to access properties and methods within the same object.

5. Strings in JavaScript

- Understanding Strings in JavaScript
- Learning String Manipulation Methods - concat() , slice(), substring() , replace(), replaceAll()
- Learning String Search and Check Operations - indexOf(), lastIndexOf() , includes(), startsWith(), endsWith()
- Learning String Transformations - toUpperCase(), toLowerCase() , trim()
- Learning String Splitting and Joining: - split() , join()
- Embed variables and expressions in strings using backticks ()`
- Learning Escape Characters - \\n , \\t , \\'
- Algorithms on Strings - Reverse a String , Check for Palindrome , Find Longest Common Prefix , Character Frequency Count , Anagram Check

6. Time and Space Complexity

- Understanding Time Complexity
- Understanding the Big-O Notation.
- Constant Time – O(1)
- Logarithmic Time – O(log n)
- Linear Time – O(n)
- Linearithmic Time – O(n log n)

- Quadratic Time – O(n^2)
- Exponential Time – O(2^n)
- Factorial Time – O(n!)
- Key Factors That Affect Complexity - Algorithm Design , Data Structure Choice , Problem Constraints
- Tips to Reduce Time Complexity - Avoid Nested Loops , Efficient Data Structures , Optimize Recursion , Divide and Conquer
- Understanding what is Recursion and its use case

7. Math Problems and Algorithms

- Understanding Mathematical Operations and Their Applications
- Mathematical operations like (pow) (sqrt) and greatest common divisor (HCF) are essential in various problem-solving scenarios.

8. Advanced Problems on Array

- Understanding Advanced Array Concepts
- Learning two-pointer approach ,
- Learning prefix sums
- Solving complex problems efficiently.
- Multi-Dimensional Arrays in JavaScript
- Working with Multi-Dimensional Arrays
- Key Operations on Multi-Dimensional Arrays
- Algorithms Using Multi-Dimensional Arrays
- Multi-Dimensional Arrays in Real-World Scenarios

9. Sorting Algorithms ,Time complexity and their application

- Learning Selection Sort
- Learning Insertion Sort
- Learning Merge Sort
- Learning Quick Sort
- Learning Cyclic Sort

10. Binary Search and Its Algorithms

- Binary Search on Sorted Arrays
- Variations of Binary Search
- Binary Search on Infinite Arrays
- Binary Search in Rotated Sorted Array
- Binary Search on 2D Matrix
- Real-World Use Cases of Binary Search

11. Hashing (Set and Map) in JavaScript

- Understanding Hashing in JavaScript - s**et , map*
- Working with Set in JavaScript
- Methods in Set - add(value) , delete(value) , has(value) , clear() , size
- Working with Map in JavaScript
- Methods in Map - set(key, value) ,get(key) , delete(key) , has(key) , clear() , size
- Learning Algorithms Using Set & map

12. Linked List in JavaScript

- Understanding Linked List - Data **,** Pointer
- Singly Linked List.
- Doubly Linked List.
- Circular Linked List.
- Creating a Node in Linked List:
- Building a Linked List:
- Traversing a Linked List:
- Operations on Linked Lists - Insertion , Deletion , Searching
- Algorithms Using Linked Lists

13. Queue in JavaScript

- Implementation of Queue by Linked List and Array
- Working with Queues - Basic Queue , Circular Queue
- Operations on Queues - Enqueue , Dequeue , Peek , IsEmpty , Size
- Algorithms Using Queues

- Applications of Queues

14. Stack in JavaScript

- Understanding Stacks in javaScript
- Implementation of Stack by Linked List and Array
- Working with Stacks
- Operations on Stacks - Push , Pop , Peek , IsEmpty , Size
- Algorithms Using Stacks
- Applications of Stacks

15. Advanced Problems on Recursion and Backtracking

- Understanding Advanced Recursion and Backtracking
- Key Problems and Algorithms like N-Queens Problem,Sudoku Solver,Subset Sum,Word Search
- Optimizing Recursive Solutions with Backtracking
- Challenges with Recursion and Backtracking
- Applications of Recursion and Backtracking

16. Tree

- Understanding Binary Trees
- Types of Binary Trees - Full Binary Tree , Complete Binary Tree , Perfect Binary Tree
- Key Terminology in Binary Trees - Node , Root , Leaf , Height of a Tree , Depth of a Node , Level of a Node
- Binary Tree Operations - Insertion , Deletion , Traversal , Searching
- Binary Tree Algorithms - Height , Diameter , LCA , Symmetry Check
- Applications of Binary Trees

17. Binary Search Tree (BST):

- Understanding Binary Search Tree
- Properties of Binary Search Tree
- BST Operations -
- Binary Search Tree Algorithms

- Applications of Binary Search Tree
- Advantages of Binary Search Tree

18. BINARY HEAP

- Min & Max Heap Concepts
- Priority Queue Implementation
- Comparator & Comparable Usage
- HeapSort Algorithm, Kth Largest Element
- Additional Heap Applications
- 8 problems

19. SLIDING WINDOW

- Efficient technique for solving subarray and substring problems
- Find maximum or minimum values in a sliding window over arrays
- Solve problems like longest substring without repeating characters
- Applied in frequency counting, sum calculations, and more
- Optimizes time complexity by avoiding nested loops
- 8 problems

20. GRAPH

- Explore fundamental graph traversals: BFS & DFS
- Detect cycles in directed & undirected graphs
- Learn Topological Sorting for DAGs
- Shortest path algorithms: Dijkstra & Bellman-Ford
- Minimum Spanning Trees with Prim’s & Kruskal’s algorithms
- Disjoint Set Union-Find for connected components & cycle detection
- Covers both directed and undirected graphs, weighted and unweighted
- 20 problems

EPISODE FIVE: Unleashing the Warrior Within

21. DYNAMIC PROGRAMMING

- Start with basics: Fibonacci, Climbing Stairs

- Classic optimization: 0/1 Knapsack
- Advanced variants: Unbounded Knapsack, Coin Change
- Sequence problems: Longest Common Subsequence (LCS), Longest Increasing Subsequence (LIS)
- Palindromic subsequence challenges
- Complex topics: DP on grids, Bitmask DP, Matrix Chain Multiplication (MCM)
- Builds problem-solving for overlapping subproblems & optimal substructure
- 20 problems

22. TRIE

- Implement Trie data structure (Prefix Tree)
- Perform search, insert, and delete operations efficiently
- Applications in Auto-complete, Spell-check, Word Search problems
- 6 problems

23. SEGMENT TREE

- Handle range queries like sum, min, max
- Optimize updates with Lazy Propagation
- Binary Indexed Tree (Fenwick Tree) for prefix sums and updates
- Efficient for interval queries and dynamic data changes
- 5 problems

Aptitude and Reasoning

Classic Chapters

1. Percentage

- Learn tips and tricks for percentages.
- Solve basic, medium, and advanced questions.
- Practice MCQs to master percentages.

2. Profit and Loss

- Concepts of Profit and loss

- Relationship between cost price, selling price, and mark-up price.
- Solve practical scenarios involving discounts, successive transactions.
- Sharpen your skills with MCQs to prepare for competitive exams.

3. Simple Interest

- Master the formula for calculating simple interest.
- Differentiate between principal, interest rate, and time period.
- Solve case-based problems related to borrowing and lending.
- Practice MCQs for thorough preparation

4. Compound Interest

- Understand the growth of investments and savings.
- Differentiate between simple interest and compound interest.
- Solve problems with annual, semi-annual, and quarterly compounding.
- Practice MCQs for preparation.

5. Ratio and Proportion

- Grasp the basics of ratios.
- Solve problems on proportional relationships.
- Analyze scenarios involving scaling, sharing, and dividing quantities.
- Practice MCQs for preparation.

Number Related Topics

1. Number System

- Understand the classification of natural numbers, whole numbers, integers, rational numbers, and irrational numbers.
- Master divisibility rules, factors, multiples, and place value.
- Practice MCQs to improve understanding and problem-solving speed.

2. HCF and LCM

- Learn techniques to find HCF and LCM.
- Understand their applications in scheduling and resource sharing.

- Solve word problems involving time, distance, and recurring patterns.
- Practice MCQs for competitive exam preparation.

3. Average

- Understand averages and their significance.
- Solve problems on weighted averages, missing numbers, and group data.
- Apply averages in performance analysis and time management.
- Practice MCQs to enhance speed and accuracy.

Speed Work and Time Related Topics

1. Work and Time

- Understand the relationship between work, time, and efficiency.
- Solve problems involving individuals or groups working together.
- Analyze scenarios like alternating work schedules and work completion rates.
- Practice MCQs problems.

2. Pipes and Cisterns

- Understand the analogy between pipes and work-time.
- Solve problems with multiple pipes working together or alternately.
- Address challenges like leaks or partial closure.
- Practice MCQs to improve your skills.

3. Speed, Distance, and Time

- Master the formula: Speed = Distance / Time.
- Solve problems on relative speed, average speed, and varying speeds.
- Practice MCQs questions.

4. Problems on Trains

- Calculate the time for a train to cross poles, platforms, or other trains.
- Apply relative speed in train-related problems.
- Solve problems with trains of different lengths and speeds.
- Practice MCQs questions.

5. Boats and Streams

- Understand the impact of stream direction (upstream, downstream) on speed.
- Solve problems on relative speed and effective speed in flowing water.
- Analyze scenarios like rowing competitions or river crossings.
- Practice MCQs to test your understanding.

Probability and Combinations

1. Permutations and Combinations

- Understand the difference between permutations (arrangement) and combinations (selection).
- Learn key formulas and techniques for calculating arrangements and selections.
- Solve problems with factorials, repetition, and circular permutations.
- Practice MCQs to improve problem-solving skills.

2. Probability

- Understand probability as a measure of likelihood.
- Learn formulas for calculating probability in events.
- Practice MCQs to improve proficiency.

Progressions

1. Arithmetic Progression (AP)

- Understand Arithmetic Progression with a constant difference.
- Derive formulas for general term (an) and sum of n terms (Sn).
- Apply AP in real-life problem solving.
- Solve problems on missing terms, specific terms, and sum of series.
- Practice MCQs and concept-based questions.

2. Geometric Progression (GP)

- Understand Geometric Progression with a constant ratio.
- Solve problems on missing terms, specific terms, and sum of series.

Miscellaneous Topics

1. Calendar

- Understand days, months, leap years, and century years.
- Learn Odd Days concept and calculation for day of the week.
- Use key formulas to find the day for any given date.
- Solve problems on repeating calendar years and calendar-based tricks.
- Practice MCQs and scenario-based questions.

2. Clocks

- Understand clock structure, minute hand, hour hand, and their movements.
- Solve angle problems between clock hands.
- Solve problems on overlaps, right angles, and opposite directions.
- Practice clock puzzles and time calculation problems.
- Practice MCQs and puzzle-based questions.

Logical Reasoning

1. Direction Sense

- Understand directions (North, South, East, West) and final direction after movements.
- Track movements and turns (right/left) to find final position.
- Solve problems with multiple directions and movement patterns.
- Practice MCQs for speed and accuracy.

2. Blood Relation

- Identify relationships like father, mother, brother, sister.
- Analyze clues to trace family connections.
- Solve problems with family trees and complex relationships.
- Practice MCQs to improve deduction skills.

3. Syllogism

- Understand logical reasoning and conclusion deduction.
- Break down premises to check conclusions.
- Work with All, Some, No premises.
- Solve MCQs to identify valid/invalid conclusions.

4. Arrangements

- Learn to arrange people or objects based on conditions.
- Apply constraints like sitting together or specific positions.
- Solve problems with multiple arrangement conditions.
- Practice MCQs to strengthen understanding.

5. Series

- Understand number sequences and identify next terms.
- Recognize patterns like arithmetic progressions, geometric progressions.
- Solve problems with varying series types and difficulty.
- Practice MCQs to improve pattern recognition.

Verbal Reasoning

1. Sentence Ordering

- Practice MCQs to improve sentence ordering skills.

2. Error Identification

- Practice MCQs to sharpen error spotting and correction.

3. Sentence Improvement

- Practice MCQs to improve sentence quality.