React Mastery Revision Guide

🎐 Chapter 01 - Inception

🧠 Theory

- **What is Emmet?**
- **Difference between a Library and Framework?**
- **What is CDN? Why do we use it?**
- **Why is React known as React?**
- **What is `crossorigin` in the script tag?**
- **What is the difference between React and ReactDOM?**
- **What is the difference between `react.development.js` and `react.production.js` files via CDN?**
- **What is `async` and `defer`?**

💻 Coding

1. Set up all the tools in your laptop:
    - **VS Code**
    - **Chrome**
    - **Extensions of Chrome**
2. **Create a new Git repo**
3. Build your first **Hello World** program using:
    - Just **HTML**
    - **JS** to manipulate the DOM
    - **React** using CDN links
4. Create an **Element** and **nested React Elements**
5. Use `root.render`
6. Push code to **Github** (Theory + Code)
7. Learn about **Arrow Functions** before the next class

🔥 Chapter 02 - Igniting our App

🧠 Theory

- **What is NPM?**
- **What is Parcel/Webpack? Why do we need it?**
- **What is `.parcel-cache`?**
- **What is `npx`?**
- **What is the difference between `dependencies` and `devDependencies`?**
- **What is Tree Shaking?**
- **What is Hot Module Replacement?**
- Favorite **5 superpowers of Parcel** and describe 3 of them
- **What is `.gitignore`?** What should be added/not added to it?

- **What is the difference between `package.json` and `package-lock.json`?**
- **Why should I not modify `package-lock.json`?**
- **What is `node_modules`?** Is it a good idea to push that on Git?
- **What is the `dist` folder?**
- **What is browserlist?**
- Read about different bundlers: **Vite**, **Webpack**, **Parcel**
- Read about: `^` - caret and `~` - tilde
- Read about **Script types in HTML** (MDN Docs)

💻 Coding

1. In your existing project:
    - Initialize **npm** in your repo
    - Install **React** and **ReactDOM**
    - Remove **CDN links** of React
    - Install **Parcel**
    - Ignite your app with **Parcel**
    - Add scripts for “start” and “build” with parcel commands
    - Add **.gitignore** file
    - Add **browserlist**
    - Build a production version of your code using `parcel build`

🏗 Chapter 03 - Laying the Foundation

🧠 Theory

- **What is JSX?**
- **Superpowers of JSX**
- **Role of `type` attribute in script tag?** What options can I use there?
- `{TitleComponent}` vs `{<TitleComponent/>}` vs `{<TitleComponent></TitleComponent>}` in JSX

💻 Coding

1. Create a Nested header Element using:
    - `React.createElement(h1, h2, h3 inside a div with class "title")`
2. Create the same element using **JSX**
3. Create a **Functional Component** with **JSX**
4. Pass attributes into the tag in **JSX**
5. **Composition of Components** (Add a component inside another)
6. `{TitleComponent}` vs `{<TitleComponent/>}` vs `{<TitleComponent></TitleComponent>}` in JSX
7. Create a **Header Component** from scratch using **Functional Component with JSX**
    - Add **Logo on the left**, **search bar in the middle**, and **user icon on the right**
    - Add **CSS** to style it

💬 Chapter 04 - Talk is Cheap, Show Me the Code

🧠 Theory

- Is JSX mandatory for React?
- Is ES6 mandatory for React?
- `{TitleComponent}` vs `{<TitleComponent/>}` vs `{<TitleComponent></TitleComponent>}` in JSX
- How can I write **comments in JSX**?
- What is `<React.Fragment></React.Fragment>` and `<> </>`?
- What is **Reconciliation** in React?
- What is **React Fiber**?
- Why do we need **keys** in React?
- Can we use **index** as keys in React?
- What are **props** in React? Ways to use them.
- What is **Config Driven UI**?

💻 Coding

1. Build a **Food Ordering App**:
    - Choose a cool name for your app
    - Build an **AppLayout**
    - Build a **Header Component** with **Logo** & **Nav Items** & **Cart**
    - Build a **Body Component**
    - Build a **RestaurantList Component**
    - Build a **RestaurantCard Component**
    - Use **static data** initially
    - Make your card **dynamic** (pass in props)
    - **Props** - passing arguments to a function with **Destructuring** & **Spread operator**
    - Render your cards with **dynamic data** of restaurants
    - Use **Array.map** to render all the restaurants

🪝 Chapter 05 - Let's Get Hooked!

🧠 Theory

- What is the difference between **Named export**, **Default export**, and `* as export`?
- What is the importance of **config.js** file?
- What are **React Hooks**?
- Why do we need the **useState Hook**?

💻 Coding

1. Clean up your code
2. Create a **Folder Structure** for your app
3. Make different files for each **Component**
4. Create a **config file**
5. Use all types of **import** and **export**
6. Create a **Search Box** in your App
7. Use **useState** to create a variable and bind it to the input box
8. Try to make your **search bar** work

🌎 Chapter 06 - Exploring the World

🧠 Theory

- What is **Microservice**?
- What is **Monolith architecture**?
- What is the difference between **Monolith** and **Microservice**?
- Why do we need the **useEffect Hook**?
- What is **Optional Chaining**?
- What is **Shimmer UI**?
- What is the difference between **JS expression** and **JS statement**?
- What is **Conditional Rendering**? Explain with a code example.
- What is **CORS**?
- What is **async and await**?
- What is the use of `const json = await data.json();` in `getRestaurants()`?

💻 Coding

1. Play with the **useEffect Hook** to see when it is called (before or after render)
2. Play with the **dependency array** in **useEffect Hook**
3. Play with the developer console by putting a **debugger** in render and useEffect
4. Call an actual **API call**
5. Handle **Error** in your API call
6. Build **Shimmer UI** when data is not loaded
7. Render your UI with actual **API data**
8. Make **Search functionality** work
9. Make a **Login/Logout button** that toggles with a state

🚀 Chapter 07 - Finding the Path

🧠 Theory

- What are various ways to add **images** into our App? Explain with code examples.
- What would happen if we do **console.log(useState())**?
- How will **useEffect** behave if we don't add a dependency array?
- What is **SPA**?
- What is the difference between **Client Side Routing** and **Server Side Routing**?

💻 Coding

1. Add **Shimmer Effect** without installing a library
2. Install **react-router-dom**
3. Create an **appRouter** and provide it to the app
4. Create **Home**, **About**, and **Contact Page** with **Link** (use child routes)
5. Make an **Error page** for routing errors
6. Create a **Restaurant Page** with dynamic restaurant ID
7. (Extra) Create a **login Page** using **Formik Library**

Chapter 08 - Let's Get Classy

Theory

- How do you create **Nested Routes** in **react-router-dom** configuration?
- Read about **createHashRouter**, **createMemoryRouter** from React Router docs.
- What is the order of **life cycle method** calls in Class-Based Components?
- Why do we use **componentDidMount**?
- Why do we use **componentWillUnmount**? Show with example.
- (Research) Why do we use `super(props)` in the constructor?
- (Research) Why can't we have the callback function of **useEffect** async?

Coding

1. Create a **Class-Based Component**
2. Create **2 class-based child components**
3. Pass **props** from Parent to child
4. Create a **constructor**
5. Create a **state variable** inside the child
6. Use `this.setState` to update it
7. What if there are multiple state variables?
8. Write a **console.log** for each lifecycle method
9. Play with the **console logs** to find out the correct order of their execution
10. Create an **interval** inside `componentDidMount`
11. Use `clearInterval` to fix the issue caused by the interval

Chapter 09 - Optimizing our App

Theory

- When and why do we need **lazy()**?
- What is **suspense**?
- Why do we get this error: "A component was suspended while responding to synchronous input"? How does suspense fix this error?
- Advantages and Disadvantages of using **this code splitting pattern**?
- When do we and why do we need **suspense**?

Coding

1. Create your **custom hooks**
2. Try out **lazy** and **suspense**
3. Make your code **clean**
