Introduction To React

Software Development Bootcamp

What is React

React is a front-end JavaScript library that allows you to create complex web applications out of reusable parts called components. Everything you are about to learn is still JavaScript. It is not a completely separate technology. It only adds features to what you've learned thus far.

Thinking In React

Take 10 minutes to read this article.

Come back and discuss.

https://reactjs.org/docs/thinking-in-react.html

React Basics

- React uses a component-based architecture
- It efficiently updates and renders components
- Uses a virtual DOM for performance optimization
- Allows for reusable UI elements

Why Use React?

Traditional DOM manipulation can be slow and cumbersome for complex web applications. React addresses these issues by:

- Providing a declarative approach to UI development
- Offering efficient updates and rendering
- Enabling creation of reusable components

The Virtual DOM

- React is a tool that allows us to manipulate the DOM more efficiently.
- It does this by manipulating the DOM on our behalf.
- To do that, React has a “virtual DOM” (its own record of what should be visible).
- It checks if the real DOM matches the virtual DOM.
- If not, it will update the real DOM through a process called reconciliation.

React | Reconciliation

Components

Components let you split the UI into independent, reusable pieces, and think about each piece in isolation.

- How is it useful to build apps using them?
- What parts of websites could be components?
- Should everything be a component?

[React | Components and Props]

Component Classes and Functions

- You can make a React component using either a function or the Component class.
- The entire industry is moving away from classes and towards functions.
- For this reason, we will focus on making components using functions.

Components as Functions

- Each component is a function.
- We capitalize the function name to let other developers know it’s a component.
- The argument passed to your component is always an object.
- This object is called props by convention.
- Your component function will return JSX.

What Is JSX?

JSX is a syntax extension for JavaScript that lets you write HTML-like code in your JavaScript files.

- A React Component returns JSX

Key Points About JSX

- Looks like HTML, works like JavaScript
- Allows embedding expressions in curly braces
- JSX attributes use camelCase
- JSX requires a single parent element

Topic

Build Tools: *Vite*

What Are Build Tools?

Build tools are like super-smart assistants for web developers. They automate repetitive tasks and optimize your code for the web.

Key Aspects Of Build Tools

- **Bundling**: Combine multiple files into one to reduce HTTP requests
- **Transpiling**: Convert modern JavaScript into older versions for wider browser support
- **Minification**: Shrink code size by removing unnecessary characters
- **Asset optimization**: Compress images, fonts, and other resources

Why Use Build Tools?

- Make development more efficient by automating tedious tasks
- Improve website performance by optimizing code and assets
- Enable use of the latest web technologies while maintaining compatibility

Build tools help create faster, more efficient websites and improve the development experience

What Is Vite?

Vite is a modern build tool and development server for web projects

- Key features:
  - **Lightning-fast startup**: Vite loads your project almost instantly
  - **Hot Module Replacement** (HMR): See changes in real-time without full page reloads
  - **Smart code splitting**: Only processes the code you're actively working on
  - **Optimized builds**: Creates super-efficient production versions of your site

Why Use Vite?

- Makes development more responsive
- Handles complex modern web apps without slowing down
- Works great with popular frameworks like React, but isn’t limited to them

Topic

Setting Up A React Project Using Vite

Getting Started

1. Set up a new project using Vite
    a. `npm create vite@latest my-react-app -- -- template react`
2. Navigate to the project directory and install dependencies
    a. `cd my-react-app`
    b. `npm install`
3. Start the development server
    a. `npm run dev`

Vite Application Structure

Take a moment to look at what Vite has created

Key Files And Directories

- `node_modules/`: Contains all installed dependencies
- `public/`: Stores static assets that don't need processing
- `src/`: Contains your application source code
  - `assets/`: Stores images, fonts, and other assets used in your app
  - `App.css`: Styles specific to the App component
  - `App.jsx`: The main React component of your application
  - `index.css`: Global styles for your application
  - `main.jsx`: The entry point of your React application

Key Files and Directories

- `.eslintrc.cjs`: Configuration for ESLint (code linting)
- `.gitignore`: Specifies files that Git should ignore
- `index.html`: The main HTML file where React app is rendered
- `package.json`: Lists project dependencies and scripts
- `package-lock.json`: Locks dependency versions for consistency
- `README.md`: Project documentation
- `vite.config.js`: Configuration file for Vite

Understanding The Entry Points

1. `index.html`: The main entry point
    a. Contains a `<div id="root"></div>` where React mounts your app
    b. Links to `src/main.jsx`
2. `src/main.jsx`: The JavaScript entry point
    a. Imports React and ReactDOM
    b. Imports the main `App` component
    c. Renders the `App` component into the DOM
3. `src/App.jsx`: The main React component
    a. Defines the structure of your application
    b. Can import and use other components

Topic

Creating A Component

Simple “Hello World” Component

- Remember:
  - Each component is a function that returns JSX
  - We capitalize the function name to let other developers know it’s a component.
- In your `src` file create a new file called `HelloWorld.jsx`
- Open `HelloWorld.jsx`

HelloWorld.jsx

- `import React from 'react'`: Imports the React library
- `function HelloWorld() {...}`: Declares a functional component named `HelloWorld`
- `return(...)`: Defines what the component will render.
- The JSX inside the return statement creates a `<div>` containing an `<h1>` with "Hello World" text
- `export default HelloWorld`: Makes the **HelloWorld** component available for use in other files

```
import React from 'react'

function HelloWorld() {
 return (
   <div>
       <h1>Hello, World</h1>
   </div>
 )
}

export default HelloWorld
```

App.jsx

- Modify `App.jsx` to use the `HelloWorld` component
- `import HelloWorld from './HelloWorld'`: Imports the `HelloWorld` component
- `function App() {...}`: The main App component
- Inside the return statement we use `<HelloWorld/>` to render our `HelloWorld` component.

```
import HelloWorld from './HelloWorld'

function App() {
 return (
   <>
     <HelloWorld />
   </>
 )
}

export default App
```

Hello World!

- Run your Vite development server with `npm run dev`
- You should see “Hello, World” displayed in your browser

Exercise

Hello React