Week 6: React Overview and the Component-Based Model

Day 1: Introduction to React and the Virtual DOM

What is React?

React is an open-source JavaScript library primarily used for building user interfaces (UI) for single-page applications. It is maintained by Meta (formerly Facebook) and a community of individual developers. Unlike full-scale frameworks, React focuses solely on the "View" layer of the application.

The Motivation for Creating React

Before React, updating the UI required direct manipulation of the Browser DOM (Document Object Model). This process was computationally expensive and often led to performance bottlenecks. React was created to solve these performance issues by introducing the **Virtual DOM**.

The Virtual (Mock) DOM

The Virtual DOM is a lightweight, in-memory representation of the actual DOM. When the state of a component changes, React first updates the Virtual DOM. Then, it performs a "diffing" algorithm to compare the new Virtual DOM with the previous version.

- **Efficiency**: Only the sub-components that actually change are re-rendered in the real DOM.
- **Speed**: Minimizing direct DOM manipulation significantly increases the speed of the application.

Day 2: The React Component Model

Understanding Components

The foundational concept of React is the **Component Model**. A React application is built as a tree of independent, reusable pieces of UI called components.

- **Declarative**: You describe what the UI should look like for a given state, and React handles the updates.
- **Encapsulation**: Each component manages its own logic and rendering.

One-Way Data Flow

React utilizes a **One-Way Data Flow** (unidirectional data flow). This means data is passed from parent components down to child components via "props". This makes the application more predictable and easier to debug than systems with two-way data binding.

Day 3: Functional vs. Class-Based Components

Functional Components

Functional components are basic JavaScript functions that return JSX.

- **Simplicity**: They are easier to read and test.
- **Props**: They receive data through a single "props" object argument.

Class-Based Components

Before Hooks, class-based components were the only way to manage state and lifecycle methods.

- **Structure**: They must extend React.Component and include a render() method.
- **Legacy**: While modern React favors functions, understanding classes is essential for maintaining older code.

Component Names

A critical rule in React is that **Component Names Must Be Capitalized**. This allows JSX to distinguish between HTML elements (like) and custom components (like <MyComponent />).

Day 4: JSX (JavaScript XML)

What is JSX?

JSX is a syntax extension for JavaScript that allows you to write HTML-like code directly within your JavaScript files.

- **Transpilation**: Browsers cannot read JSX directly; it is transpiled into standard React.createElement() calls.
- **Expressions**: You can embed any JavaScript expression inside JSX by wrapping it in curly braces {}.
- **Escaping**: JSX escapes values by default to prevent Cross-Site Scripting (XSS) attacks.

Nesting and Fragments

- **Nesting**: JSX elements can be nested inside one another, similar to HTML.
- **Fragments**: Since a component must return a single root element, <React.Fragment> (or <>...</>) is used to group multiple elements without adding extra nodes to the DOM.

Day 5: React Project Structure and Libraries

Creating React Projects

Modern React projects are typically initialized using tools like create-react-app or Vite.

- **Project Structure**: A standard project includes a src folder for components, a public folder for static assets, and a package.json file for managing dependencies.
- **Project Files**: Key files include App.js (the root component) and index.js (the entry point where the React app is mounted to the DOM).

React Libraries

React is part of a larger ecosystem. To build a full-scale application, you often integrate additional libraries:

- **React-Router**: For navigation and routing.
- **Redux**: For advanced state management.
- **Axios**: For making API calls to your **Gadget API**.