React.js: Building UIs

Presentation by Vicki Wu
MEDP 34100 Midterm
11/13/24

Introduction to React

- React is a JavaScript library for building user interfaces
- It was developed by Facebook in 2013 and has become one of the top libraries for building web applications
- Used by major companies like Facebook and Netflix

Key Features of React

- Uses components to build different parts of the user interface, making code modular and organized
- Users can write code that describes what we want on the screen, and React updates the page as data changes
- React changes only the parts that need updating, rather than reloading the whole page

Benefits of Using React

- Because React only updates parts of the page when needed, apps built with React are usually faster and feel smoother to use
- You can create a component once and use it anywhere in your app. For example, if you create a button component, you can use it on multiple pages with different text or styles
- React has a lot of vast resources and third-party libraries, so finding help and plugins is easy

How React Works: JSX and Components

- JavaScript XML (JSX) is a syntax extension that looks like HTML within JavaScript, making code more readable and allowing HTML-like syntax in React components

- React has functional and class components, with functional components being more common and recommended for simplicity

```jsx
function App() {
  return (
    <div>
      <h1>Hello World</h1>
    </div>
  );
}

export default App;
```

Hello World

States and Props in React

- Props (read-only) are used to pass data from one component to another, keeping components flexible and reusable

- States, on the other hand, allows components to manage their own dynamic data, changing how components render based on user interactions or data changes

```
const myElement = <Car brand="Ford" />;

function Car(props) {
  return <h2>I am a { props.brand }!</h2>;
}
```

props is an object that receives the brand value ("Ford") passed as a property to the Car component, which is then accessed using by “props.brand”

```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }
```

state is an object that contains data specific to this Car component - it's storing properties like the brand ("Ford"), model ("Mustang"), color ("red"), and year (1964)

React Hooks

- React's hooks give components features like "memory" (state) and the ability to handle changes over time
  - In plain JavaScript, we'd need a lot of extra code to track changes and update the screen ourselves
  - With hooks, React does this for us, keeping our code simpler and more predictable.

- Key Hooks
  - "useState" lets us save and update values directly in a component. In JavaScript, we'd need separate variables, and we'd have to update the DOM manually every time these values change
  - "useEffect" lets us do things that don't directly change the UI—like fetching data, setting up a timer, or updating the page title. In JavaScript, this would mean creating event listeners and cleaning them up, which can lead to memory issues if we forget

React Router

- With React Router, users can move between parts of the app instantly, with no page reloads
  - In JavaScript, every link opens a new page and reloads all content, which disrupts the experience
  - React Router, on the other hand, lets us swap out parts of the screen without reloading, which keeps the app fast and responsive

- React Router connects different URLs to parts of the app, so users can save or share URLs and reopen exact views later
  - In regular HTML and JavaScript, each URL would load a whole new page, but React Router gives us the flexibility to load only what changes, keeping data and views consistent while allowing easy navigation

Setting Up a Simple React App

1. Install Node.js and npm
    a. Node.js is a runtime environment for running JavaScript outside a browser, and npm (Node Package Manager) is used to install JavaScript packages
2. Open the terminal and run the command: “npx create-react-app my-app”
    a. npx is a package runner tool that comes with npm. This command creates a new folder called my-app with all the files and dependencies for a React project.
3. Once the project is created, navigate to the folder you just created by running the command “cd my-app” in the terminal
4. Then, start the development server by running “npm start”

Simple Counter

```jsx
import React, { useState } from "react";

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div style={{ textAlign: "center", padding: "20px" }}>
      <h2>Simple React Counter</h2>
      <p>Current Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increase</button>
      <button onClick={() => setCount(count - 1)}>Decrease</button>
    </div>
  );
}

export default Counter;
```

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Simple Counter</title>
    <style>
      body {
        text-align: center;
        padding: 20px;
      }
    </style>
  </head>
  <body>
    <h2>Simple Counter</h2>
    <p id="count">Current Count: 0</p>
    <button id="increase">Increase</button>
    <button id="decrease">Decrease</button>

    <script>
      let count = 0;

      const countDisplay = document.getElementById("count");

      const increaseButton = document.getElementById("increase");
      const decreaseButton = document.getElementById("decrease");

      function updateCount() {
        countDisplay.innerText = `Current Count: ${count}`;
      }

      increaseButton.addEventListener("click", function () {
        count++;
        updateCount();
      });

      decreaseButton.addEventListener("click", function () {
        count--;
        updateCount();
      });
    </script>
  </body>
</html>
```

Breakdown of Differences

1. State Management:
   a. React: The useState hook handles the state (count) and automatically re-renders the component when the state changes
   b. JS: You manually manage the state (count) and explicitly update the DOM when the state changes using innerText
2. DOM Manipulation:
   a. React: React updates the UI efficiently using the virtual DOM. You only specify what the UI should look like
   b. JS: You directly manipulate the DOM, manually selecting elements with document.getElementById() and updating them with .innerText
3. Event Handling:
   a. React: The onClick handler is attached to the button elements directly in the JSX
   b. JS: You use addEventListener to attach click event listeners to the buttons