ReactJS:
Single Page Application (SPA) development

Rohith Raj Srinivasan

MS CS student at UC Davis
Ex-Cisco | Ex-Samsung Research
BE in CSE from RVCE (class of 2020)

rohith-raj-s

rohith.june6@gmail.com

What is ReactJS?

- ReactJS is an open-source JavaScript library used for building **user interfaces (UIs).**
- Developed and maintained by **Meta**, ReactJS allows developers to create *dynamic* and *interactive* web applications with ease.
- Its primary role is to efficiently manage the rendering and updating of UI components in response to changes in application state.
- Widely used for developing **Single Page Applications (SPAs)**

Single Page Application?

- Single Page Applications (SPAs) are web applications that operate within a **single web page**.
- They offer a more *seamless and interactive user experience* compared to traditional multi-page applications.
- In SPAs, the content is dynamically updated and loaded on the same page without requiring a full page reload, making them feel more like desktop applications.

Single Page Application?

Multi-page application

Traditional

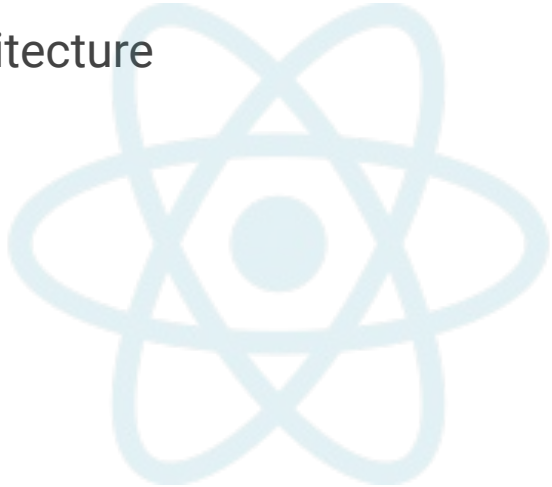
Every request for new information gives you a new version of the whole page.

Single Page Application

You request just the pieces you need.

Key features?

- Component-Based Architecture
- Virtual DOM
- Declarative Syntax
- React Hooks
- One-Way Data Binding



React Component

- Components are the fundamental & reusable **building blocks** in React applications.
- They encapsulate logic and UI elements.
- Allow developers to divide the UI into smaller and manageable parts.
- Components can be either **functional** or **class-based**.

Functional components

- Returns JSX to describe the UI. It is a simpler and more concise way to create components.
- With the introduction of React Hooks in version 16.8, functional components can now manage state and handle lifecycle events using hooks like useState, useEffect, and more.
- Generally **more performant** than class-based components since they don't involve the overhead of creating an instance of a class.

```
import React from 'react';

function FunctionalComponent() {
  return <div>Hello, I am a functional component!</div>;
};
```

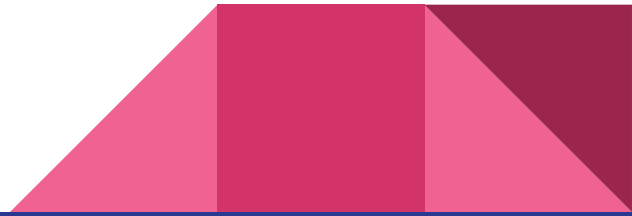
Class-based components (traditional way)

- A class-based component is a JavaScript class that extends the **React.Component** class.
- Class-based components have access to React's lifecycle methods (e.g., componentDidMount, componentDidUpdate, componentWillUnmount).
- They can also manage state using this.state and setState() methods.
- Class-based components have a **slightly higher performance overhead** due to the nature of JavaScript classes and instances.

```
class ClassBasedComponent extends Component {
  render() {
    return <div>Hello, I am a class-based component!</div>;
  }
}
```

Virtual DOM

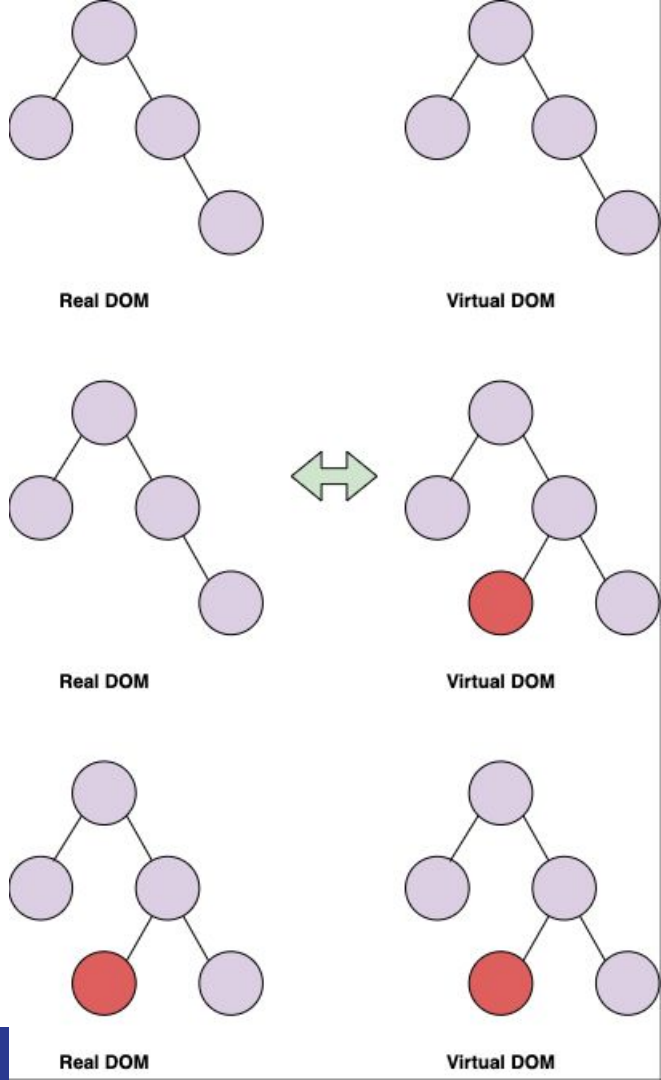
- Document Object Model (DOM) is the abstraction of the HTML of a web page.
- The virtual DOM is the abstraction of the real DOM.
- A virtual DOM object is the same as a real DOM object, except that it is a **lightweight copy.**
- Upon any change of a property, it only updates the corresponding nodes and not the entire tree. That makes it a quick and efficient alternative.



Virtual DOM Reconciliation

Purple node = Elements rendered on the first load.

Orange node = Updated element due to user interaction.



Virtual DOM Advantages

- Performance
- Abstraction
- Efficiency

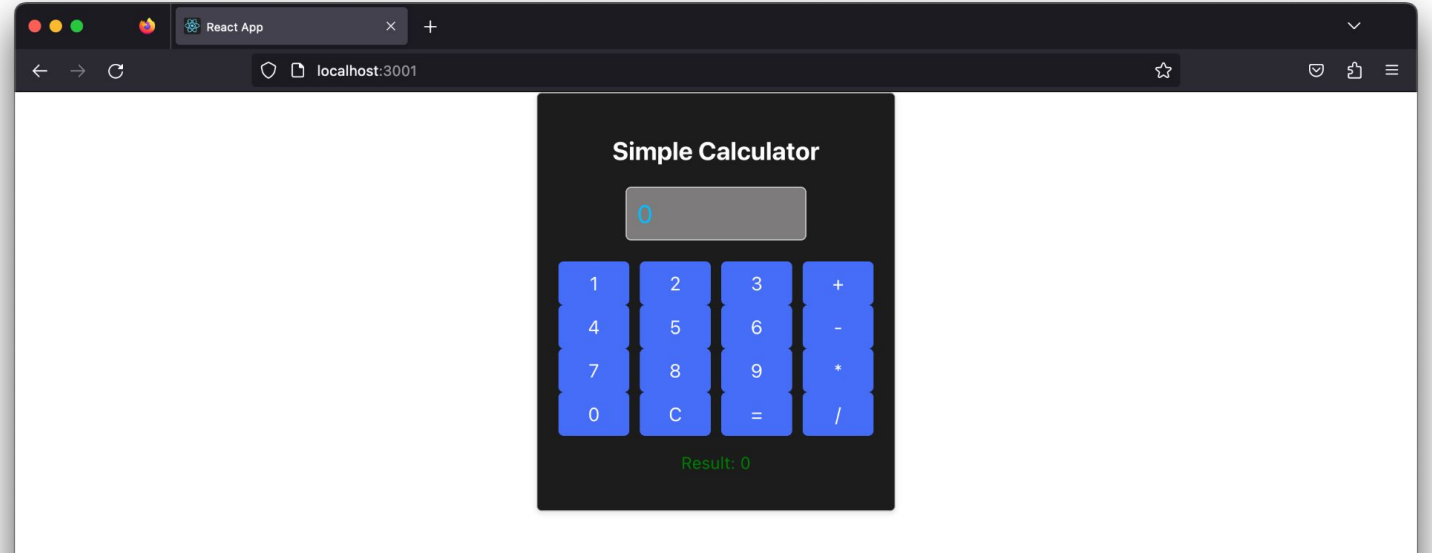
JavaScript XML (JSX)

- JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.
- JSX is an extension of the JavaScript language based on ES6, and is translated into regular JavaScript at runtime.

```
const myElementJSX = <h1>This is JSX</h1>;
const myElement = React.createElement('h1', {}, 'This isn\'t JSX!');

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElementJSX); //with JSX
root.render(myElement); //without JSX
```

Building a Simple Calculator App using ReactJS



https://github.com/rohis06/SimpleCalc

Setting up the Dev environment

Install Node.js:

- Download the latest version of Node.js from https://nodejs.org for your operating system (Windows, macOS, or Linux).
- Run the installer and follow the on-screen instructions to complete the installation process.
- After installation, you can check if Node.js and npm are correctly installed by running the following commands:

```
node -v
```

```
npm -v
```

Setting up the Dev environment

Install a Text Editor:

- Visual Studio Code (https://code.visualstudio.com)
- Sublime Text (https://www.sublimetext.com)
- Atom (https://atom.io)
- Notepad++ (https://notepad-plus-plus.org)

Creating a new React project

Create React App: https://github.com/facebook/create-react-app

npm install -g create-react-app

npx create-react-app my-calc-app

cd my-calc-app

npm start

Extra slides

React CSS Styling

- Inline styling
- CSS stylesheets
- CSS modules

Material UI

- Popular open-source UI framework for React applications.
- Material-UI offers a wide range of customizable and reusable UI components, such as buttons, forms, cards, navigation bars, dialog boxes, and more.

  Key features:
  - Pre-Designed Components
  - Customizable Themes
  - Responsive Design
  - CSS-in-JS
  - Support for Icons
  - Integration with React Ecosystem

https://mui.com/material-ui/getting-started/usage/

React Hooks

- Hooks allow function components to have access to state and other React features. Because of this, class components are generally no longer needed.

- Frequently used hooks:
  - useState
  - useEffect
  - useContext
  - useRef
  - useMemo

Further reading

1. Official React documentation: https://react.dev/
2. React Getting Started Guide on MDN Web Docs: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started
3. React - The Complete Guide (incl. Hooks, React Router, Redux) on Udemy: https://www.udemy.com/course/react-the-complete-guide-incl-redux/
4. YouTube tutorial: https://youtu.be/SqcY0GlETPk