Introduction to ReactJS

A Student-Run Short Course (SRC) conducted by the Student Academic Council in collaboration with the Technical Council.

Instructors:

- Reuben Devanesan
- Ananthu JP
- Kanishk Singhal

Logistics

- 16th Oct to 4th Nov, 10 PM to 11 PM. AB 1/101, Learning Theatre.
- All important updates will be communicated through Google Classroom.
- All codes will be uploaded to the GitHub repository. Link: https://github.com/Reuben27/ReactJS-SRC
- Part I: https://github.com/Reuben27/Web-Development-SRC
- P/F course. Attendance compulsory.

What is ReactJS?

- React.js is an open-source JavaScript library developed by Facebook. It's used for building interactive user interfaces and web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript.
- React is the view layer of an MVC application (Model View Controller). MVC framework is an architectural/design pattern that separates an application into three main logical components Model, View, and Controller. Each architectural component is built to handle specific development aspects of an application.

Why ReactJS?

- ❏ React is *fast*. Apps made in React can handle complex updates and still feel quick and responsive. How? JavaScript is fast, but updating the DOM makes it slow. React minimizes DOM changes. And it has figured out the most efficient and intelligent way to update DOM.
- ❏ React is *modular*. Instead of writing large, dense files of code, you can write many smaller, reusable files. React's modularity can be a beautiful solution to JavaScript's maintainability problems. Maintainable JavaScript.
- ❏ React is *scalable*. Large programs that display a lot of changing data are where React performs best.

Why ReactJS?

- ❏ React is *flexible*. You can use React for interesting projects that have nothing to do with making a web app. Once you have learned it, you can use it on a vast variety of platforms to build quality user interfaces. React is a library, NOT a framework. Its library approach has allowed React to evolve into such a remarkable tool. There's room to explore. Library vs Framework?
- ❏ React is *popular*. While this reason has admittedly little to do with React's quality, the truth is that understanding React will make you more employable. Since 2015, React's popularity has grown steadily. It has a massive active community and its GitHub Repository has over 164k Stars => *broader community support*.

Basics

- Components: Building Blocks of UI
- JSX (JavaScript XML) : Syntax Extension for components.

  E.g. const helloElement = <h1>Hello World!!</h1>
- Rendering: How React displays components on the screen.

Setting Up

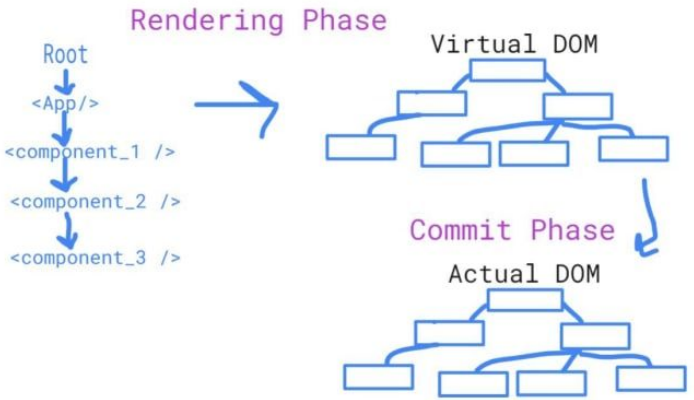
- Prerequisites: Node.js and a code editor (e.g., Visual Studio Code).
- Create a new React app using Create React App (CRA).
- Run
  - npx create-react-app my-app
  - cd my-app/
  - npm start

What is JSX?

- JSX is a syntax extension of JavaScript that combines the JavaScript and HTML-like syntax to provide highly functional, reusable markup. It's used to create DOM elements which are then rendered in the React DOM.
- https://www.codecademy.com/resources/docs/react/jsx

Virtual DOM

- React uses a virtual representation of the actual DOM.
- Reconciles changes and updates the actual DOM efficiently.
- Minimizes unnecessary re-renders.



React Components

- React apps are made out of components. A component is a piece of the UI (user interface) that has its own logic and appearance.
- A component can be as small as a button, or as large as an entire page.
- React component names must always start with a capital letter, while HTML tags must be lowercase.
- The export default keywords specify the main component in the file.

```
function MyButton() {
  return (
    <button>I'm a button</button>
  );
}
```

React Components

Function based: They are **simpler** and more **concise**. They are defined as JavaScript functions and take **props (input data)** as an argument, returning **React elements** to be rendered.

Class based: They are are **JavaScript classes** that extend from **React.Component.** They have a bit more boilerplate but offer additional features, such as **state management and lifecycle methods.**

```
import React from 'react'

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

export default App
```

Function Based Component

```
import React, { Component } from 'react'

export class App extends Component {
  render() {
    return (
      <div>App</div>
    )
  }
}

export default App
```

Class Based Component

React Components

Functional Components	Class Components
Simple, pure JavaScript functions.	ES6 classes extending React.Component.
Accept props as parameters.	Maintain their own state using this.state.
Return JSX to define the UI.	Render method returns the UI structure.

Rendering a Component

Day 4 > todo > src > index.js > ...

```
1   import React from 'react';
2   import ReactDOM from 'react-dom/client';
3   import App from './App';
4
5   const root = ReactDOM.createRoot(document.getElementById('root'));
6   root.render(
7     <React.StrictMode>
8       <App />
9     </React.StrictMode>
10  );
```

React Props

- React components use props to communicate with each other. Every parent component can pass some information to its child components by giving them props.
- Props might remind you of HTML attributes, but you can pass any JavaScript value through them, including objects, arrays, and functions.

```
export default function Profile() {
  return (
    <Avatar
      person={{ name: 'Lin Lanying', imageId: '1bX5QH6' }}
      size={100}
    />
  );
}
```

```
function Avatar({ person, size }) {
  return (
    <img
      className="avatar"
      src={getImageUrl(person)}
      alt={person.name}
      width={size}
      height={size}
    />
  );
}
```

Event Handlers

- You can respond to events by declaring event handler functions inside your components.
- Event handlers are your own functions that will be triggered in response to interactions like clicking, hovering, focusing form inputs, and so on.
- Event handler functions:
  - Are usually defined inside your components.
  - Have names that start with handle, followed by the name of the event.
- By convention, it is common to name event handlers as handle followed by the event name.

Thank you!