SWINBURNE UNIVERSITY OF TECHNOLOGY

2025-COS30049-Computing Technology Innovation Project

Workshop Guide

Workshop 08

React env set up and introduction of react.js

Objective:

By the end of this workshop, students will:

1. Set up the React Environment: Install and verify Node.js on macOS/Windows and be ready to start React development.
2. Create a React Project: Use npx create-react-app to build a React project and understand its folder structure.
3. Learn React Rendering: Modify the default app to practice rendering components.
4. Understand State Management: Use state to make components dynamic and responsive to user actions.

1. Recapture lecture slides (20 mins)

- **Node.js Overview:**
  - What is Node.js and its role in the React ecosystem?

- **Introduction to React.js:**
  - What is React?
  - Understanding the component-based architecture of React

- **Structure of a React Project**
  - **node_modules**
    - How npm manages them?
  - **src Folder:**
    - Key files: index.js, App.js
    - The entry point and component structure.
  - **public Folder:**
  - **App.js and App.css:**
    - Role of App.js in a React project as the main component.
    - CSS is applied in React projects through App.css.

2. Environment Setup – Install Node.js (30 mins)

For Mac:

1. **Check for Homebrew Installation:**
    a. First, let's check if you have Homebrew installed. In your terminal, run

```
brew --version
```

    b. If you don't have Homebrew, install it:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/in
stall.sh)"
```

2. **Install Node.js Using Homebrew:**
    a. After installing Homebrew (or if you already had it), use it to install Node.js:

```
brew install node
```

3. **Verify Installation:**
    a. Once the installation is complete, verify that Node.js and npm are installed:

```
node --version
npm --version
```

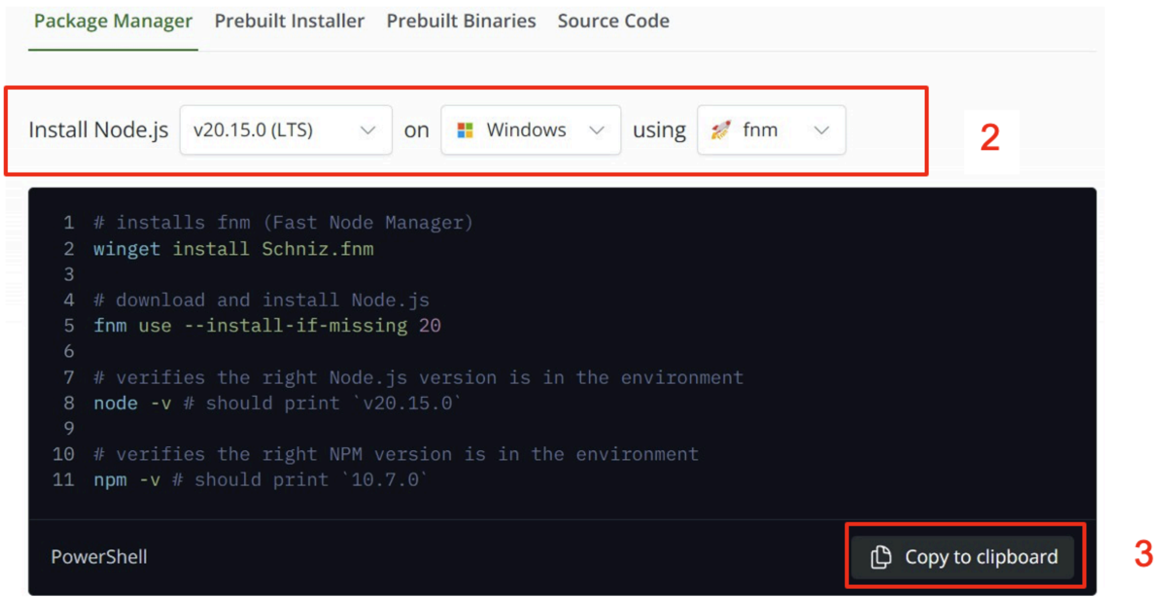
For Windows:

1. **Download Node.js:**
    a. Go to the Node.js download page: https://nodejs.org/en/download/package-manager

2. **Install Node.js:**
    a. Choose the appropriate version based on your needs and follow the installer instructions.
3. **Verify Installation:**

a. After installation, open the Command Prompt or PowerShell and verify that Node.js and npm are installed correctly by running:

```
node --version
npm --version
```



Install Node.js v20.15.0 (LTS) on Windows using fnm

```
# installs fnm (Fast Node Manager)
winget install Schniz.fnm

# download and install Node.js
fnm use --install-if-missing 20

# verifies the right Node.js version is in the environment
node -v # should print `v20.15.0`

# verifies the right NPM version is in the environment
npm -v # should print `10.7.0`
```

3. Build and Explore a React Project (30 mins)

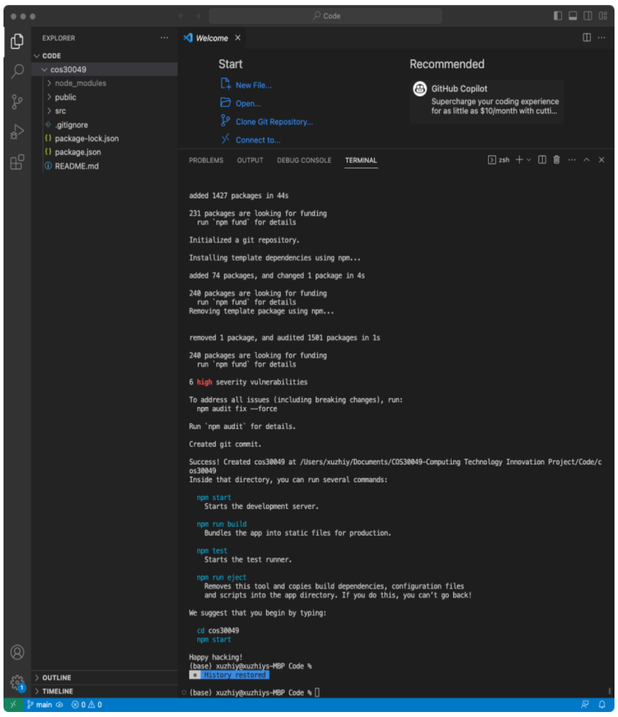
In this section, we will guide you through creating your React project step by step. The entire process will be handled through your terminal, so please follow the commands carefully to successfully set up your React application.

1. **Navigate to Your Desired Directory**
    a. Find the path to your directory where you want to create your project (Use the cd command to change directories if needed.)

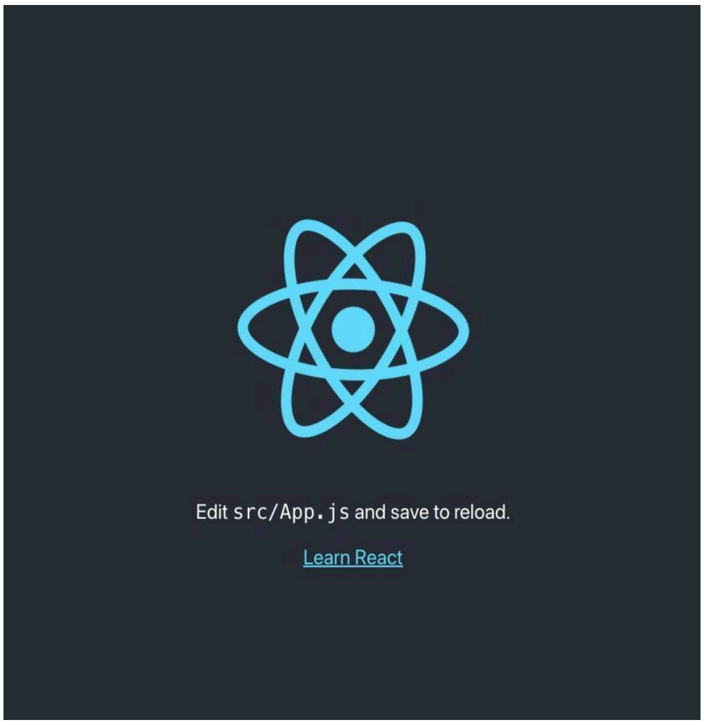
```
cd path/to/your/folder
```

2. **Create Your React Project**
    a. Now, we'll create your React project, Replace "cos30049" with the name of your project. Then you will get a sample code (starter React project) provide by React:

```
npx create-react-app cos30049
```



```
added 1427 packages in 44s

231 packages are looking for funding
  run `npm fund` for details

Initialized a git repository.

Installing template dependencies using npm...

added 74 packages, and changed 1 package in 4s

240 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...


removed 1 package, and audited 1501 packages in 1s

240 packages are looking for funding
  run `npm fund` for details

6 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Created git commit.

Success! Created cos30049 at /Users/xuzhiy/Documents/COS30049-Computing Technology Innovation Project/Code/c
os30049
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd cos30049
  npm start

Happy hacking!
(base) xuzhiy@xuzhiys-MBP Code %
 * History restored

(base) xuzhiy@xuzhiys-MBP Code %
```



Edit `src/App.js` and save to reload.

Learn React

4. Understanding Rendering in React (15 mins)

In React, rendering refers to how the UI is displayed to the user. React components are rendered using the *render( )* method (for class components) or through a return statement (for functional components).

- React intelligently re-renders components when their data or state changes, minimizing unnecessary updates to the DOM.
- **JSX**: React uses JSX, which allows HTML-like syntax within JavaScript. JSX is compiled to JavaScript.

Now, try modifying your code from the default React app and render a custom message. You can replace the default content in App.js with your own text, like this:

```
function App() {
  return (
    <div>
      <h1>Hello, welcome to my React app!</h1>
      <p>This is a custom message replacing the default 
content.</p>
    </div>
  );
}

export default App;
```

Question to Think About:
After making these changes, what happens when you save the file?

Note: **If you see the "No matching export" error**, it's likely because there's a mismatch between how App is exported and imported.

1. **Check** App.jsx:
   a. If you're using export **default** App, the import in index.js should be **without curly braces**:
      `import App from './App';`
   b. If you're using export **function** App, the import in index.js should have **curly braces**:
      `import { App } from './App';`

5. Introduction to State in React (25 mins)

Now that you've seen how React automatically updates the browser when rendering changes, let's explore how React manages data inside a component using **state**.

What is State?
State is a JavaScript object that holds data which may change over time. It allows components to be dynamic and responsive to user actions. When the state of a component changes, React re-renders the component to reflect the updated state in the browser.

Task:
Replace the default content in App.js to use state. Follow the example below:

```
import React, { useState } from 'react';

function App() {
  // Declare a state variable called "count" and a function
to update it
  const [count, setCount] = useState(0);

  return (
    <div className="App">
      <h1>React Counter</h1>
      <p>Current count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increase
Count</button>
      <button onClick={() => setCount(count - 1)}>Decrease
Count</button>
    </div>
  );
}

export default App;
```

Play with the State:

- Try initializing the state with a different value.
- Add another piece of state for something like “name” and create an input field that updates the “name” as the user types.
- Change the styling or layout based on the state values.

Question to Think About:

What do you think happens behind the scenes when the state is updated? Why does React only update certain parts of the page?