

Week 5 Lab Tutorial
React Programming using VSC
CN5006

Prepared by: Dr.Nadeem Qazi

This tutorial introduces you basic of React programming and how to create to functional component to create user interface. At the end of this Lab tutorial you will be able to do following:

- How to create a React Application
- How to change properties:
 - a) Change the text appearing into your React Application
 - b) Change the background color into your React Application
- Creating stateless function component using React
- Creating the Functional Component using properties
- Adding Click Event for Functional Components

Remember to submit the answers of the questions at the end of the this lab exercise Link is provided in feedback and assessment

What is React:

React is an open source UI library created by Facebook, which is used to build interactive web applications (mostly single page applications) made up of components.

How to create React App

1. If you have not done it before, You must installed Node js on your laptop.
2. Create a new directory **React work** in your laptop

3. You can use either Method 1 or Method 2 to run the React Application both works fine

Method 1 use Console to run the npx command

4. Move to this directory cd : C:\CN5006\ReactWork and run npm init
5. Write following two commands on the command prompt :

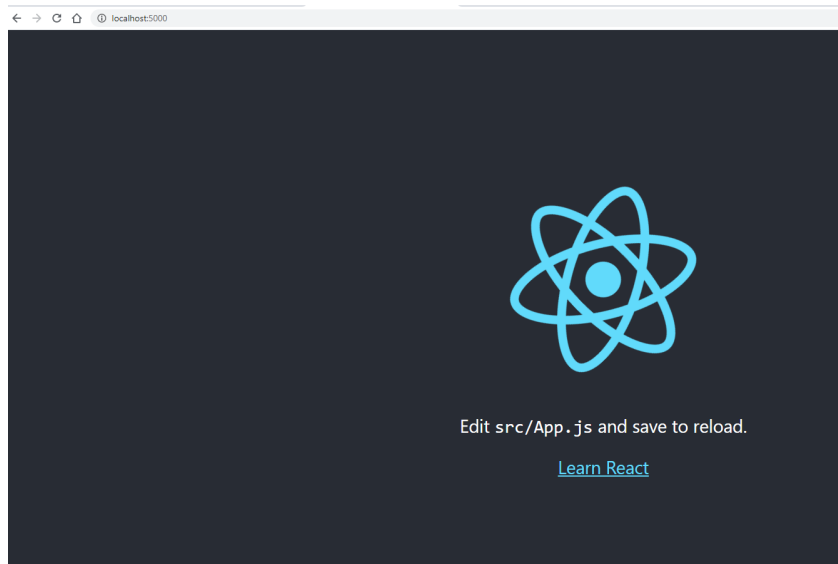
```
6. npx create-react-app myapp
```

Note you can give it any name other than myapp. Wait until react creates your application it may take a few minutes.

7. Open Visual Studio and open the folder to your **myapp** (you now should know how to do this.) make sure your current directory in the terminal video points to the directory when you created the application myapp.
8. Now run your application using npm start in the terminal window.
9. **Cong**, you have created your first React Application and it is running in your browser at localhost:3000
- 10.If you wish to change the port number from 3000 to another port number such as 5000 do following:
 - a. Ctrl C to terminate the programme
 - b. Now in the terminal window write :

c. \$env:PORT=5000

- d. Run the programme again using **npm start** and you will this time your application would be running on different port.



Method 2 :use VSC terminal to run npx command.

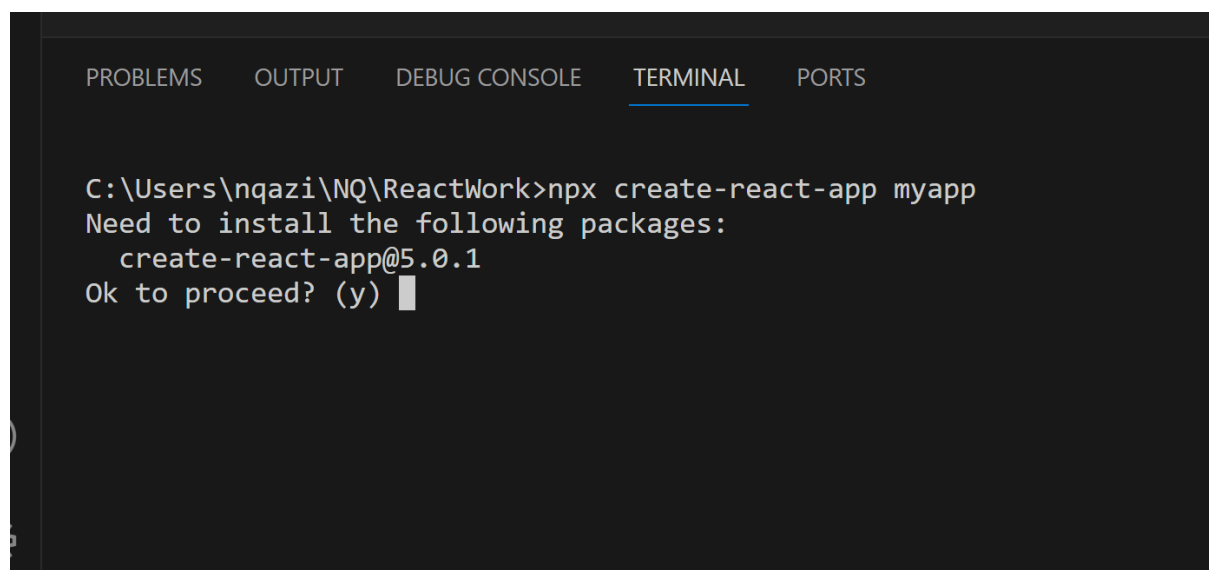
Open Visual Studio and open the folder myReact that you just created by opening the terminal window and using the **cd command** followed by the path.

In my case, it is **cd C:\Users\nqazi\NQ\ReactWork**

Then write down the following command

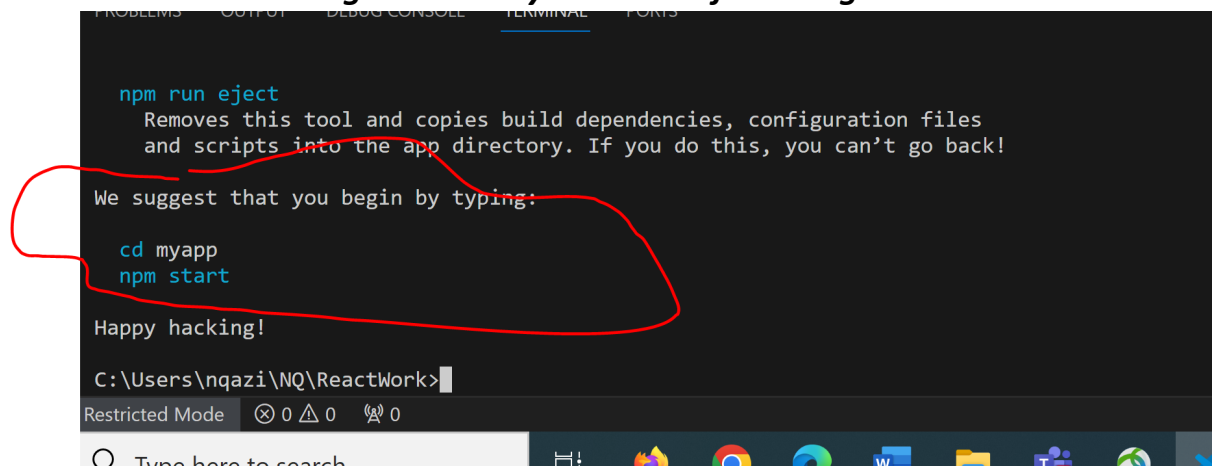
npx create-react-app myapp

See the screenshot below



Press y

You its starts installing wait until you see the following screen



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

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!

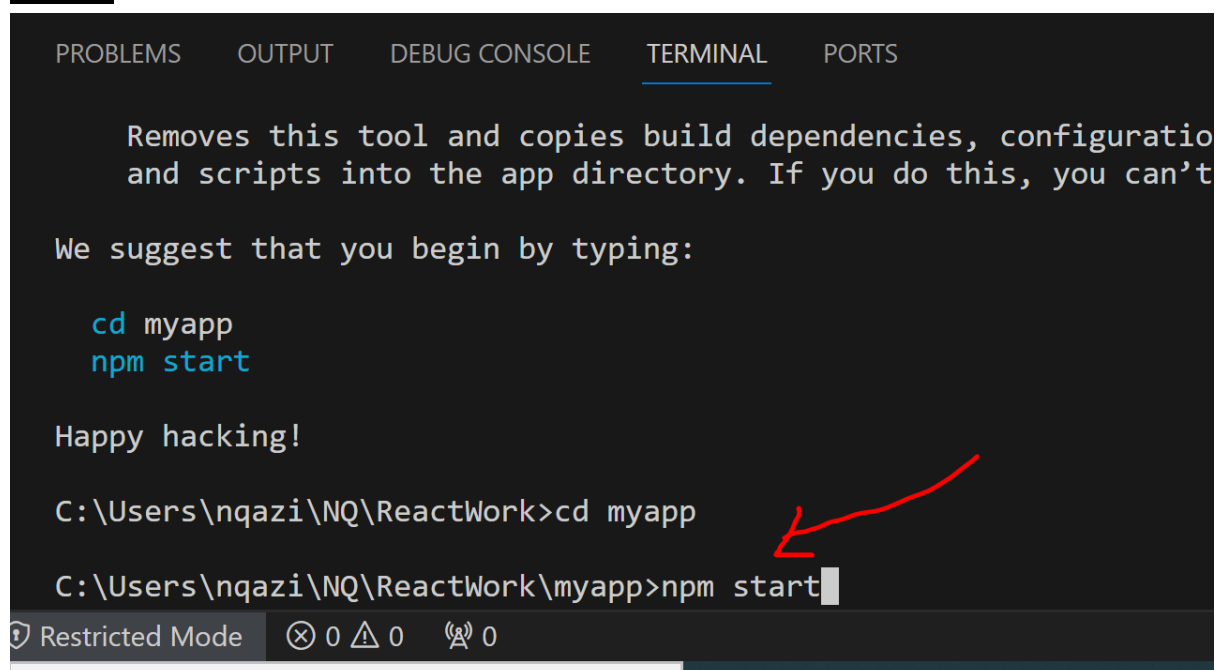
We suggest that you begin by typing:

cd myapp
npm start

Happy hacking!

C:\Users\nqazi\NQ\ReactWork>
```

Finally change to myapp and run the command npm start



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Removes this tool and copies build dependencies, configuration
and scripts into the app directory. If you do this, you can't

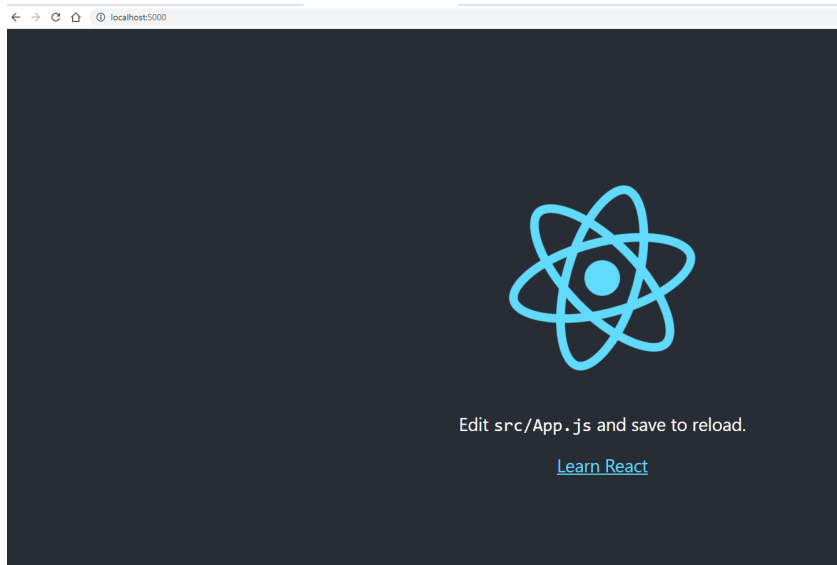
We suggest that you begin by typing:

cd myapp
npm start

Happy hacking!

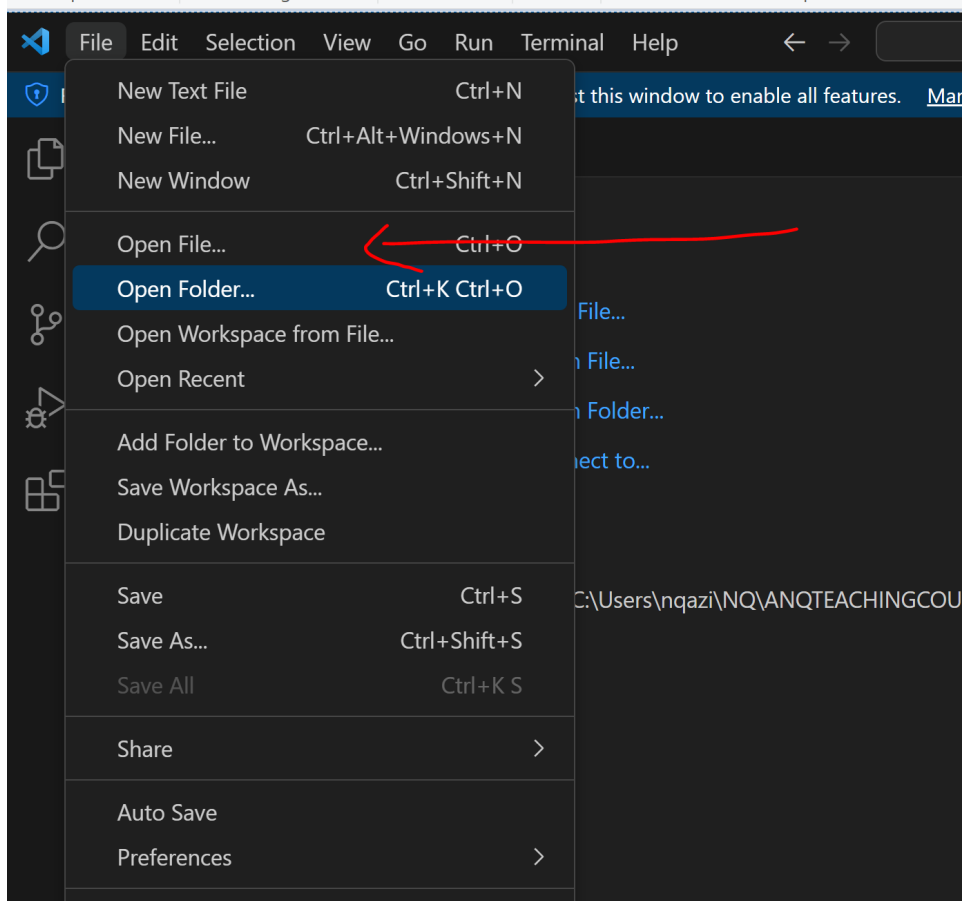
C:\Users\nqazi\NQ\ReactWork>cd myapp
C:\Users\nqazi\NQ\ReactWork\myapp>npm start
```

Your application will start running in the browser window at port 3000

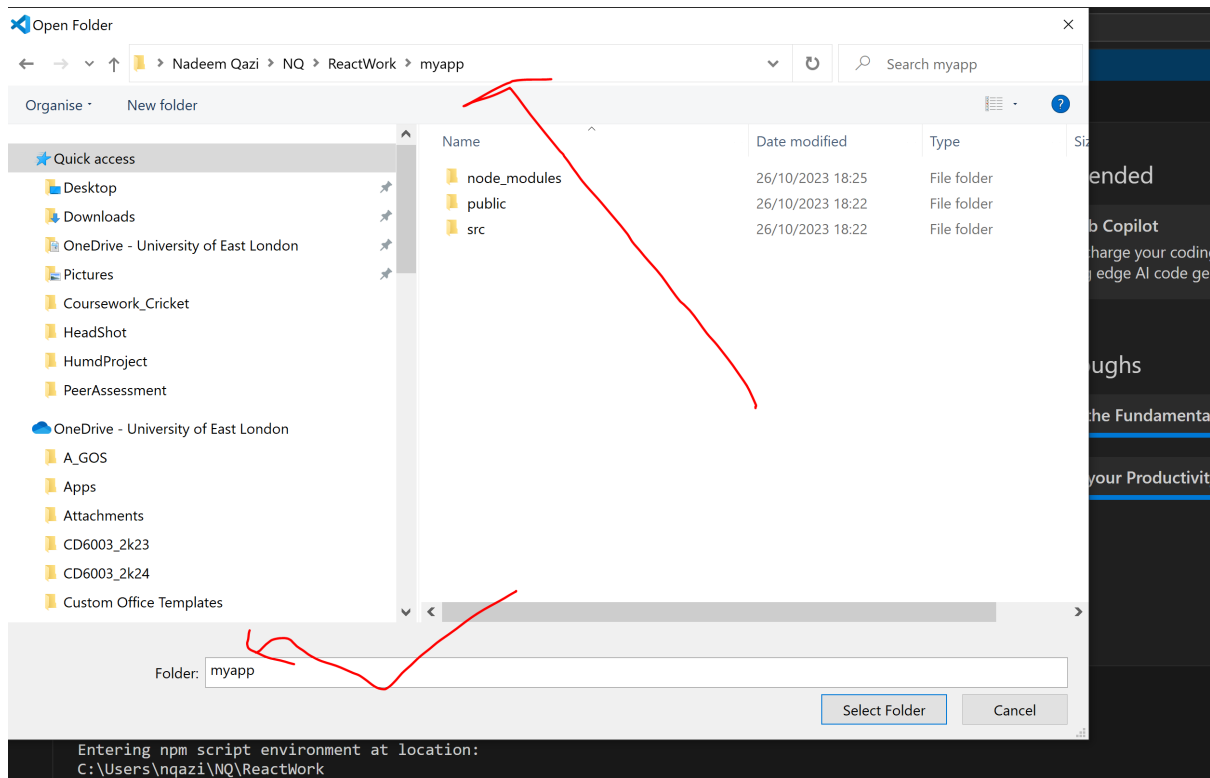


Structure of the React app:

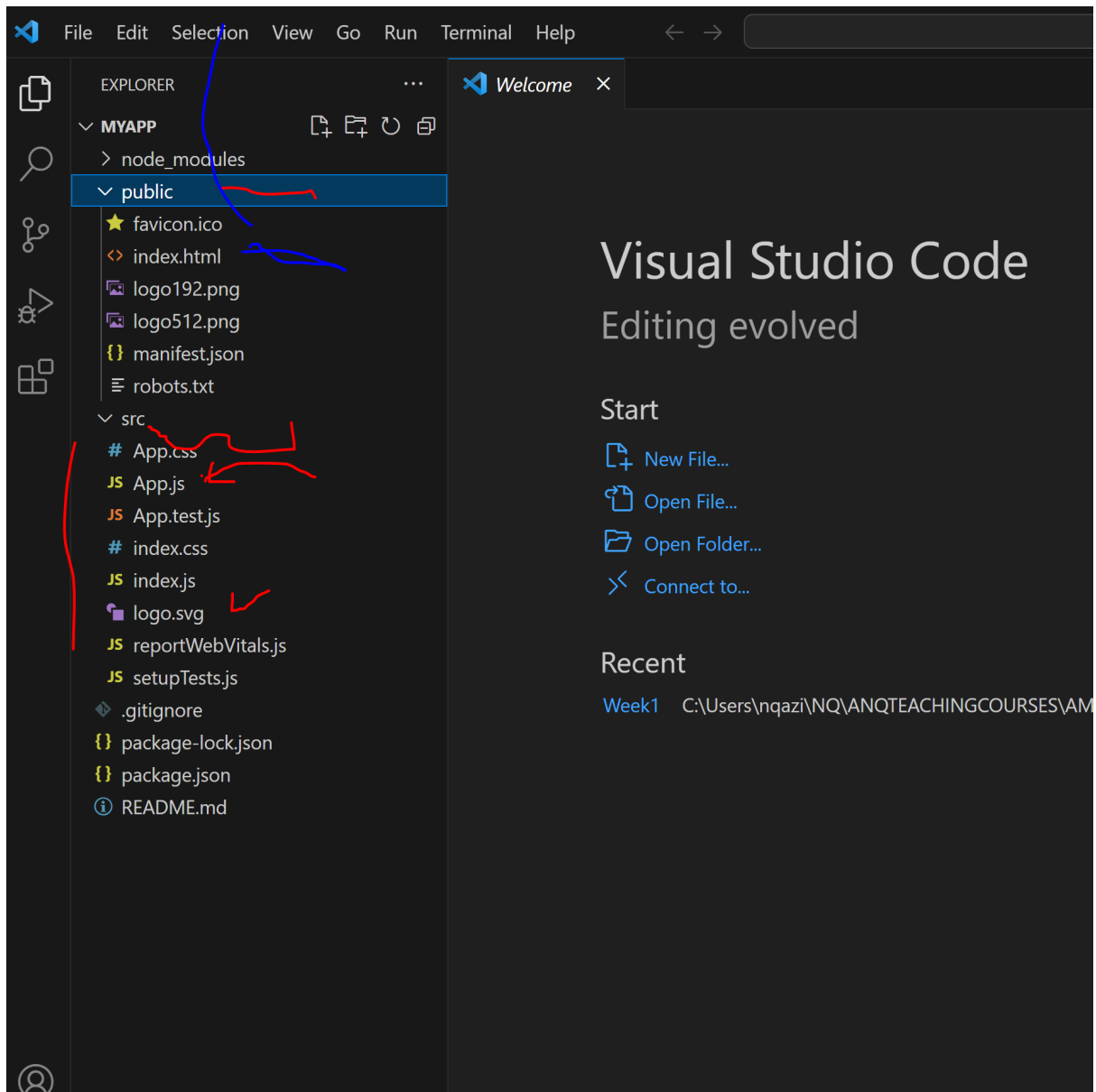
- Open Visual Studio.
- Navigate to the folder where you created the React application.
- Locate the directory named myReact.
- Inside myReact, find the folder named myapp (or the specific name you used on your system).
- Open this folder in Visual Studio to begin working with your React application.



Look at the folder where you created the myapp and open it



Look at the structure of the React application created for you by the React library: You will see the following screen on your Visual studio

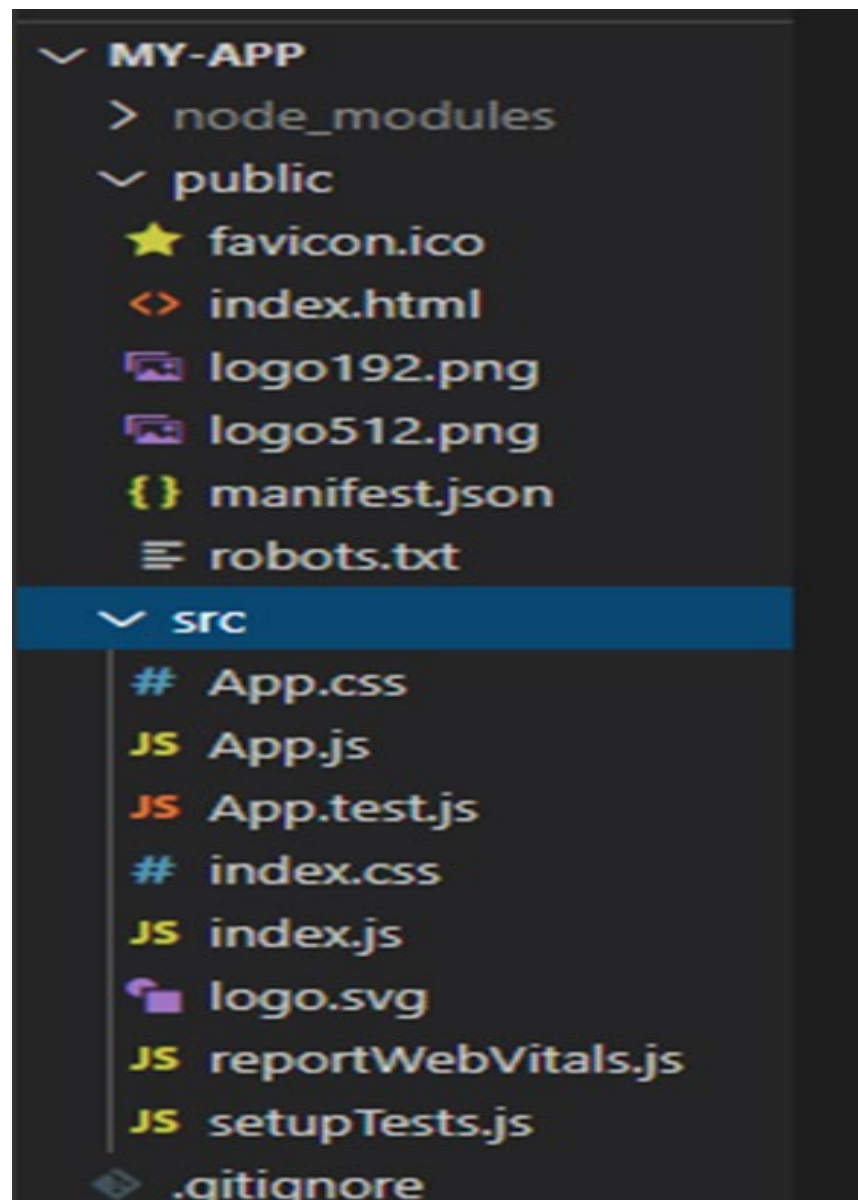


1. Public Directory

- a. Index.html: which is displayed as web page in the browser

2. Src Directory

- a. App.js: is the JS file that contains component
- b. Index.js : is the main file that produce the index.html which runs in browser

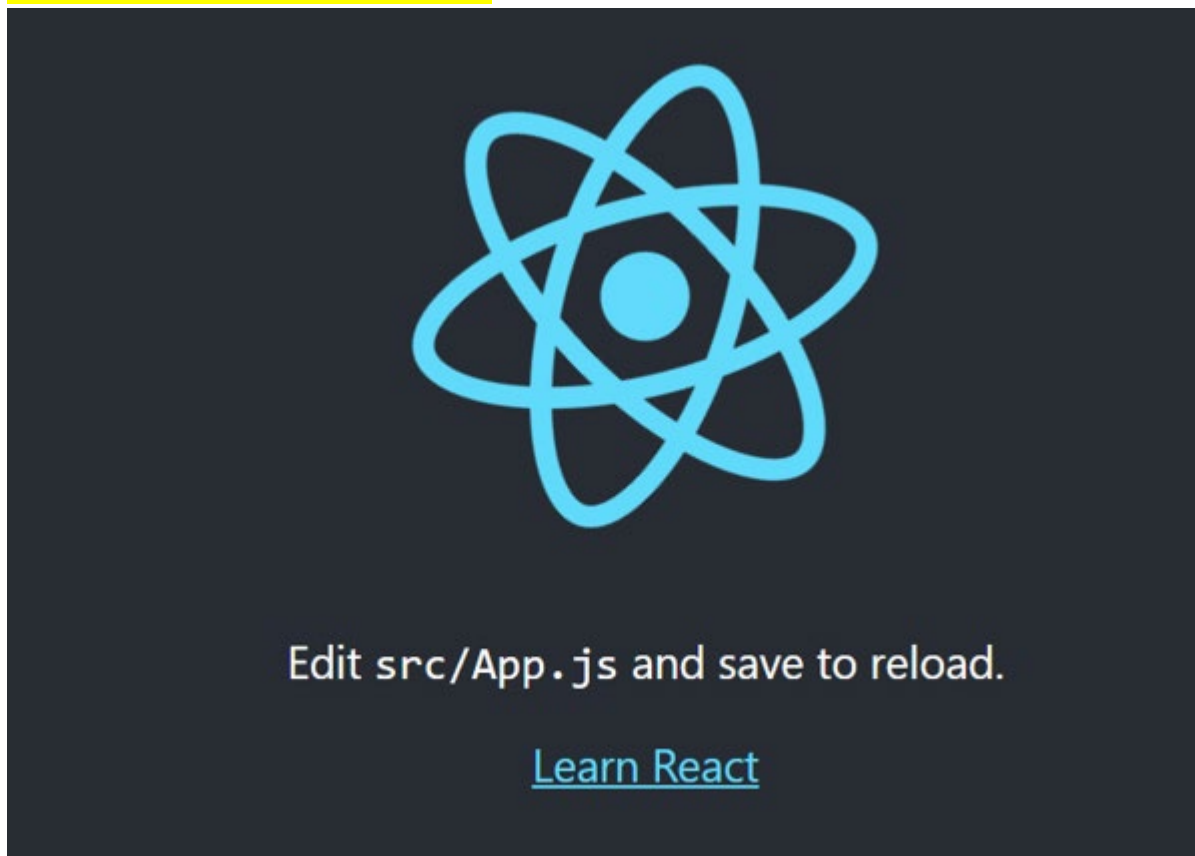


Note: Most of the Time you will be editing or creating **app.js** and using it in **index.js** to code any new feature in your React Application

Also, observe there are two CSS files:

- i) app.css file: for setting css for your app.js
- ii) Index.css : for the overall css of your application

Run the application using the command `npm start` you will see the following :



Task 1: Change the text appearing into your React Application

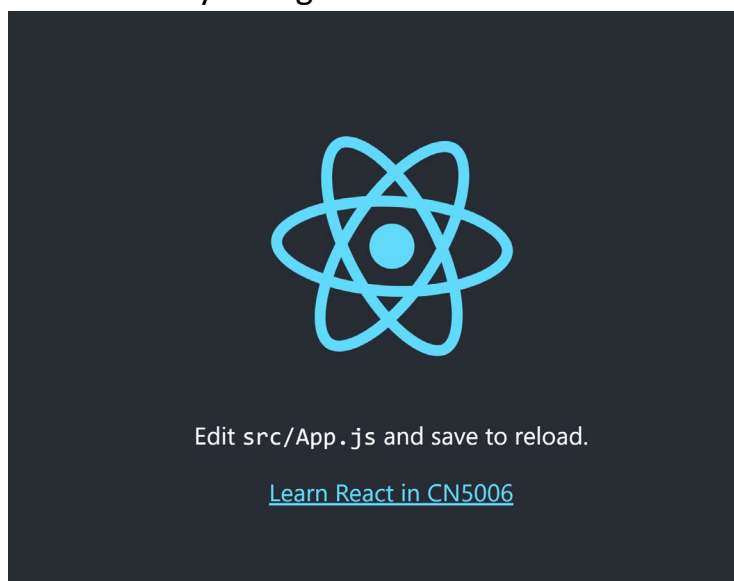
Step 1: click app.js you will see the following code window open in the code window:

```
EXPLORER  ...  {} manifest.json  JS index.js  JS Appcd5108.js  JS AppOld.js  JS Appnhq.js  Release Notes: 1.51.0  JS A

> OPEN EDITORS  2 UNSAVED  src > JS App.js > App
  MY-APP
  > build
  > node_modules
  public
    Capture.PNG
    favicon.ico
    index.html
    logo192.png
    logo512.png
    manifest.json  M
    robots.txt
  src
    # App.css
    JS App.js  M
    JS App.test.js
    JS Appcd5108.js
    JS Appnhq.js  U
    JS AppOld.js
    JS Appold1.js
    # index.css
    JS index.js
    JS LearningReact.js
    logo.svg
    JS MongoDB.js
    JS Myform.js
    JS NodeJspg.js
    JS reportWebVitals.js
    JS setupTests.js
    .gitignore
    JS App.js
    {} package-lock.json
    {} package.json
    @ README.md

1  import logo from './logo.svg';
2  import './App.css';
3
4  function App() {
5    return (
6      <div className="App">
7        <header className="App-header">
8          <img src={logo} className="App-logo" alt="logo" />
9          <p>
10             Edit <code>src/App.js</code> and save to reload.
11          </p>
12          <a
13            className="App-link"
14            href="https://reactjs.org"
15            target="_blank"
16            rel="noopener noreferrer"
17          >
18             Learn React
19          </a>
20        </header>
21      </div>
22    );
23  }
24
25  export default App;
26
```

Note Line 18 (you might have a different line number) that says: **Learn React**, change it to any new text of your choice such as **Learn React in CN5006**. and click save all : You will notice, your running application will automatically change the text to Learn React in C5006.



Note: If your application is not running ,refresh browser or type **ctrl c** in the terminal window type **npm start** and your application start running with new text.

Task2: Change the background color of your application:

Step 1: Click app.js and observe its code as open in the code window:

```
src > # App.css > {} @media (prefers-reduced-motion: no-preference)
1  .App {
2    text-align: center;
3  }
4
5  .App-logo {
6    height: 40vmin;
7    pointer-events: none;
8  }
9
10 @media (prefers-reduced-motion: no-preference) {
11   .App-logo {
12     animation: App-logo-spin infinite 20s linear;
13   }
14 }
15
16 .App-header {
17   background-color: #282c34;
18   min-height: 100vh;
19   display: flex;
20   flex-direction: column;
21   align-items: center;
22   justify-content: center;
23   font-size: calc(10px + 2vmin);
24   color: white;
25 }
26
```

Check the line 17 `background-color: #282234`, (you may have different line number) click it, and change it to any color of your choice such as `#2f3428`, and save all. you will note a background color change in your application.

How does a React application works

1. React application display index.html which is created by index.js.

2. What index.js does:

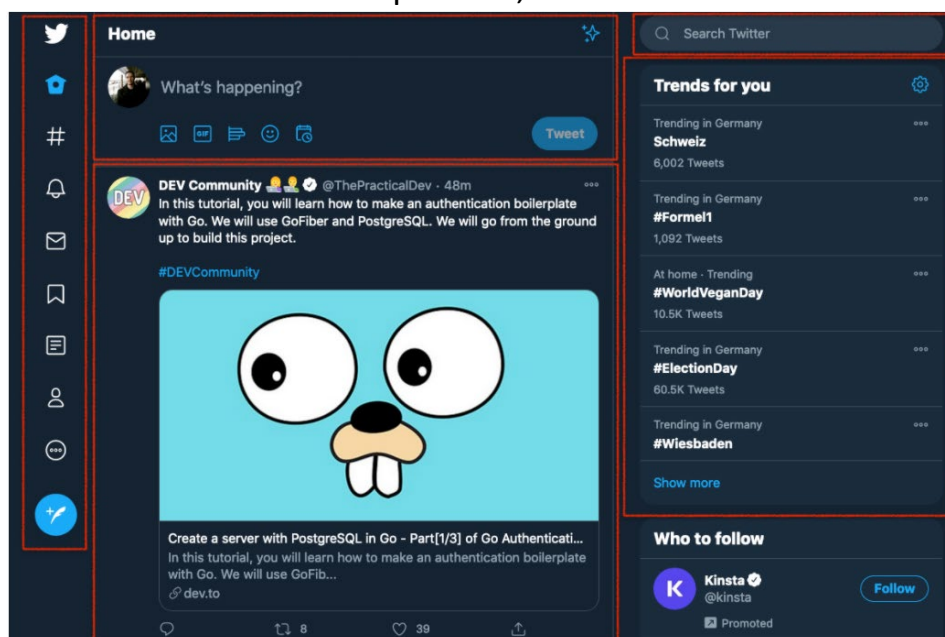
3. The content of the index.html is created by index.js.
4. index.js uses a function called as `render()` to render any element component or **any jsx code**. These elements, components or JSX code, however, may also be defined in some other js files such as app.js.
5. App.js or any other js file that holds the definition of your component must be imported in index.js as shown in the figure below we have imported App in Line 4
6. Line 8 in the figure below: `ReactDOM.render()` This `render()` function uses `<App/>` that generates html content for html page to be displayed in the react app.
7. Note App is a component created in App.js file.

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 |
6 import reportWebVitals from './reportWebVitals';
7
8 ReactDOM.render(
9   <React.StrictMode>
10     <App/>
11   </React.StrictMode>,
12   document.getElementById('root')
13 );
14
15 // If you want to start measuring performance in your app, pass a function
16 // to log results (for example: reportWebVitals(console.log))
17 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
18 reportWebVitals();
19
```

8.

What is Components in React application.

A component is an independent, reusable code block which divides the UI into smaller pieces. React is a huge library to explore. Everything in React is a component, and these usually take the form of JavaScript classes. Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, **but work in isolation and returns HTML via a render function**. An example of react components can be seen in the figure below: All the red boxes are components,



every application we build using React has to deal with

- components,
- props,
- states, and methods.

Components come in two types,

- **Function or stateless components, (deprecated in React 16.0)**
 - A functional component is just a plain JavaScript function and hence are easier to test and read.
 - which accepts props as an argument and returns a React element.
- The input in form of props, shapes the rendered output. These kind of components don't manage state People call them Functional Stateless Components, because they are stateless and expressed by a function. However, React Hooks made it possible to have state in Function Components.
- **Class components**
- A class component requires you to extend from `React.Component` and create a render function which returns a React element.
- Note A component can refer or contain other components multiple times.

What is React props:

- Props are arguments passed into React components.
- Props are passed to components via HTML attributes.

What is JSX:

React uses a separate language called as JSX to create React element. JSX stands for JavaScript XML. JSX allows us to write HTML in React. JSX makes it easier to write and add HTML in React. JSX converts HTML tags into react elements. JSX follows XML rules, and therefore HTML elements must be properly closed.

JSX uses a mix of HTML and JavaScript whereas the JavaScript is used with curly braces within the HTML

JSX Example 1:

```
var myvar=<h1> This is simple JSX</h1>
```

As you can see JSX allows us to write HTML directly within the JavaScript code.. In the above code html tag h1 is assigned to a JavaScript variable using JSX. This element are then rendered in React DOM.

Example 2:

```
const myelement =<h1>React is {5+5}times better with JSX</h1>;
```

Observe the use of { } to include and execute an expression in the html tag i.e h1. This variable will be rendered using the command :

```
ReactDOM.render(myelement, document.getElementById('root'));
```

And it will display on the web page

React is 10 times better than with JSX

Example 3:

```
const Secondelement = <input type="text" />;
```

observe the closing tag with /> if you forget to close it JSX will throw an error if the HTML is not properly closed

```
ReactDOM.render(Secondelement , document.getElementById('root'));
```

Task3 : Creating stateless function component using React

In this task we will be creating a functional component that just displays greeting message. We will create a separate JS file to code our greeting element in JSX and then will use this greeting element in the index.js.

Step 1: Create a file in visual code studio and save it as myGreetingApp.js

A functional component is created through defining a function.

In my greetingApp.js we will create a function that will hold a variable and HTML tag. You can give it any name but in this Task I have given it as GreetingElement. Once we create this function we will also export it so that it can be used outside this file . as shown in the code below: The code is given in black box and explained here.

- Line 1: import './App.css' is included because we want to use app.css in our new file for some decoration.
- Line2: we define a function named as GreetingElement ()
- inside the function GreetingElement, Line 3 Cons greeting = "Hello Function Component" is used to create a Javascript variable named as greeting.

- Line 4 we use return statement to show what this function will return. In this case this function returns a div which contain heading tag `<h1> </h1>`
- In side the return Line 5 is the HTML tag div and it uses the class App as defined in app.css ,Do not worry about this , just use it like this.
- Line 6 `<h1> {greeting}</h1>` , observe the use of curly bracket to include the variable greeting in the heading tage of h1. This is the example of JSX as here we combined Javascript with the HTML using JSX.
- Line 7 outside the return body : we have exported GreetingElement function.

myGreetingApp.js

```
import './App.css';
function GreetingElement() {
  const greeting = 'Hello Lets start learning fun
tion Component...';
  return (
    <div className="App">
      <h1>{greeting}</h1>;
    </div>
  );
}
export default GreetingElement;
```

Step2: Using Created functional Component in index.js file.

Now in step 2 we will use this functional component i.e. **mygreeting** in our index.js.

Open the index.js in your project. Note you do not need to create it. It is already created in your project just click index.js to open it.

Inside the index.js the first step is to import the **function GreetingElement()** from the file **myGreetingApp.js** To do this ,Use the following line

```
import GreetingElement from './myGreetingApp'
```

Next, In index.js clear the tag **<App/>** , you will see it after the line **<React.StrictMode>**.

Step 3. After clearing the **<App/>** , write the line **< GreetingElement/>** . Remember it is case sensitive. Your code should look like this :

Index.JS

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import GreetingElement from './myGreetingApp'
import reportWebVitals from './reportWebVitals';
ReactDOM.render(
  <React.StrictMode>
    <GreetingElement/>
  </React.StrictMode>,
  document.getElementById('root')
);
reportWebVitals();
```

Step 4: Save ALL and you will see updated page in the browser . if you do nt see any thing Refresh the browser or in terminal window type Ctrl C and then re run the programme using npm start.

Task 4 Creating the Functional Component using properties

In this task we will create a functional component **with properties**, which means we can set the behaviour of the component by setting properties. The basic methodology is still the same as we adopted in the Tasks 3 i.e. we will create another js file to define our functional component and then use it in index.js. however this time we will pass properties as argument to the function.

Step 1: Create a separate file mygreetingprop.js. use the code as given in the window below

Note we have defined a function **GreetingElementwithProp(props)**, where props is the parameter that will be set when we use this function component in index.js. Also note we have used props.msg in curly bracket. Where msg is the name of the property. Do not forget to export this function using the line **export default GreetingElementwithProp;**

Mygreetingprop.js

```
import './App.css';
function GreetingElementwithProp(props) {
  const greeting = 'Hello Lets start learning function Component...';
  console.log("prop is",props.msg)
  return (
    <div className="App">
      <h1>{props.msg}</h1>;
    </div>
  );
}
export default GreetingElementwithProp;
```

Step 2:

Now in index file import the file Mygreetingprops.js using the line

import GreetingElementwithProp from './myGreetingProp';

nrxt, clear the line `<GreetingElement/>` that we used in Task3 and in place of this create a new element that we have defined in myGreetingProp.js. use following line to do it:

`<GreetingElementwithProp msg="Hi its Monday"/`

Note that the GreetingElementwithProp is actually the name of the function.

Note the property being passed to this function is msg and it contains the string "Hi it is Monday ". remember we used {props.msg} to display whatever string is being passed to this function, in this case it is Hi it is Monday.

index.js

```
import ReactDOM from 'react-dom';
import './index.css';
import GreetingElement from './myGreetingApp'
import reportWebVitals from './reportWebVitals';
import GreetingElementwithProp from './myGreetingProp';
ReactDOM.render(
  <React.StrictMode>
    <GreetingElementwithProp msg="Hi its Monday"/>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Step 3

Step 4: Save ALL and you will see updated page in the browser . if you do nt see any thing Refresh the browser or in terminal window type Ctrl C and then re run the programme using npm start.

Task 5 .

Use the same GreetingElementwithProp to display seven days of week greeting message. Hint use <GreetingElementwithProp tag multiple times with different strings values in msg property.

Task6 : Functional Components using properties and HTML elements such as buttons

In this task we will be using html tag button ,text and labels in our functional component. The idea is that we want to create a React component that can change the background color on the click of the button. The methodology is still the same. We create this component in separate js file, name it as AppbackgroundColor.js . Add the code as given in the black window

AppbackgroundColor.js

```
import './App.css';

function AppColor(props) {
  function greetUser(e) {
    document.body.style.background = e.target.value;
    alert("Welcome Mr"+ document.getElementById(props.
color).value)
  }
  return (
    <body style={{backgroundColor:'powderblue',color:'black'}}>
      <div className="App">
        <h1>{props.heading} </h1>
        <p style={{color:'blue',font: '30px Arial'}}> How to
create function comoponent.</p>
        <label class="label" id="lbl">{props.lbl}</label>
        <input id={props.color} type="text" />
        <button value={props.color} onClick={greetUser} >Col
our me {props.color}</button>
```

```

    </div>
  </body>
);
}
export default AppColor;

```

Note in this code we have used button and added an event onClick. Note the name of the function greetUser is defined in curly bracket . So that when the button is pressed the greetUser function is activated and change the color of the back ground using the command

```
document.body.style.background = e.target.value;
```

Note the value set in the button is through **props.color** which is the property value will be given in index.js file when we use this element. In your index.js add following line:

```
import AppColor from './AppbackgroundColor'
```

Now create three elements of AppColor using the following lines of code :

1. <AppColor heading="This is first element" lbl="Name :"
color="green"/>
2. <AppColor heading="This is second element" lbl="Name :"
color="blue"/>
3. <AppColor heading="This is third third element" lbl="Name :"
color="Yellow"/>

Note we just created three elements of AppColor each time with different set of properties. the names of properties are headings, lbl and color.

```

import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import GreetingElement from './myGreetingApp'
import AppColor from './AppbackgroundColor'
import reportWebVitals from './reportWebVitals';

ReactDOM.render(
  <React.StrictMode>
    <AppColor heading="This is first element" lbl=
"Name :" color="green"/>
    <AppColor heading="This is second element" lbl=
"Name :" color="blue"/>
    <AppColor heading="This is third third element
" lbl="Name :" color="Yellow"/>
  </React.StrictMode>,
  document.getElementById('root')
);

reportWebVitals();

```

Step 4: Save ALL and you will see updated page in the browser . if you do nt see any thing Refresh the browser or in terminal window type Ctrl C and then re run the programme using npm start. Write in textbox and Press button one by one to see the change in background color.

For todays Lab submission, After you complete the lab write down a word document answering following questions for your portfolio:

- 1.** What is React

2. What do you understand by React component and what command do you use to create a React component with or without property
3. What command will you use to render the the newly created component named as MyReact
4. Suppose the MyReact Component has a property heading, write down the code that could be used to render the MYReact Component, and pass the message to the property heading as "this is my first element"
5. Observe this code and answer the questions below

```
6. <AppColor heading="This is first element" lbl
    ="Name :" color="green"/>
```

- a. What is the name of the React Component
 - b. How many properties this component uses
7. Look at the following Code:
- ```
function GreetingElementwithProp(props) {
```

```
 return (
 <div className="App">
 <h1>Wellcome , {props.studentname}</h1>;
 </div>
);
 }
 export default ??????
```

**what will you write to make this export this function correctly?**  
**Hint you need to replace ?????? with the correct word.**

**Add a function that takes two properties as numbers ,add these numbers on the click event of the button and display the sum.**

**Hint you will be using in jsx**

```
<button value={props.color} onClick={Namdofyourfunc
tion}
```