

**The Ultimate  
React Course**

# Lecture 13: Introduction to React

-Vishal Sharma

JS

# Prerequisites:

- Basic Web Development Knowledge
  - HTML
  - CSS
  - Javascript
- Text Editor
  - VS Code

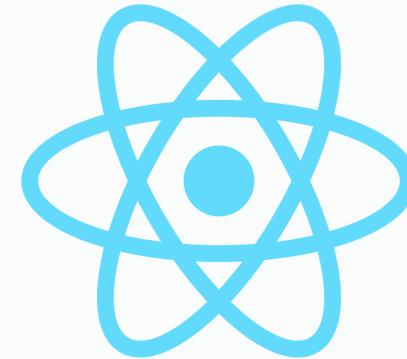


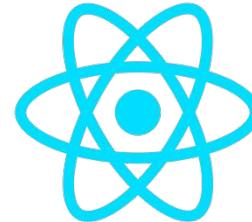
I have been already  
taught these before



# Table of Contents

- What is React? (Overview and History)
- Setting up a react development environment
- Understanding the React Project Structure
- Creating Functional Components
- JSX Syntax: Writing HTML in javascript
- Rendering components in React application





# What is React?

A brief Overview

# Have you ever visited a restaurant?

In a local restaurant, the chef makes each dish from scratch, providing flexibility but it can be slow and labour intensive. In a franchise, standardized recipes and pre-made ingredients make the process faster and more consistent.



Local Restaurant



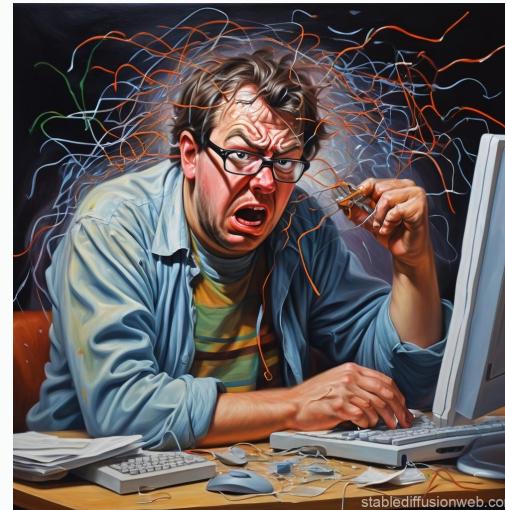
Franchise Restaurant

# JavaScript: Local Restaurant of Web Dev

Just like a local restaurant, JavaScript is flexible but can be slow but difficult to maintain and labor-intensive as the app grows.



Can't make from scratch



Fed up  
rewriting the  
code..

# React: The Franchise Model

React just like a franchise restaurant, it offers standardized processes, reusable components, and faster, more consistent app development.



Cooked this  
perfect dish in few  
moments..



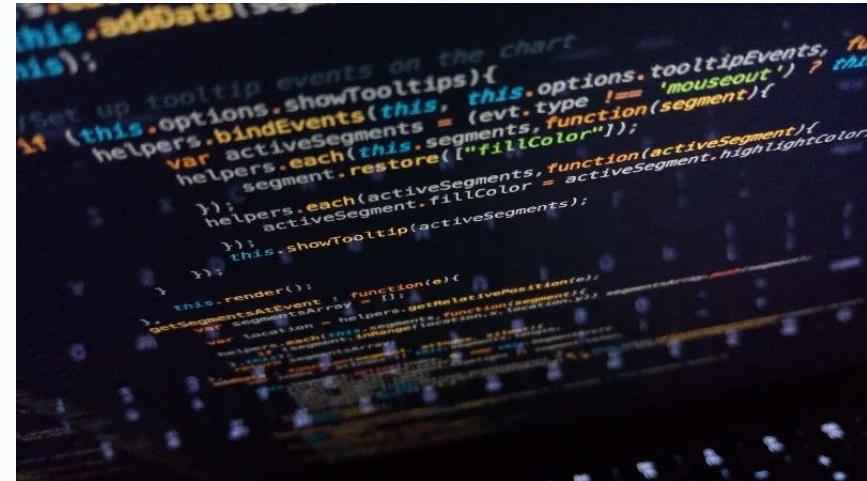
Wrote two day  
code in 2 hrs..

# Struggles of JavaScript Development

As web applications became more dynamic, codebase grew larger javascript struggled with low performance, inefficient updates, and complex code was hard to manage and maintain.



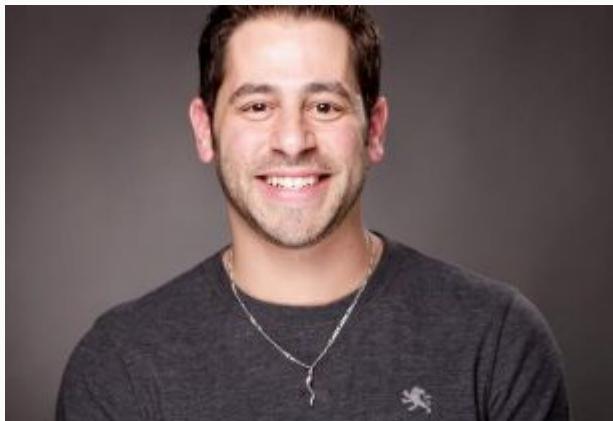
It's Struggle to maintain JavaScript websites



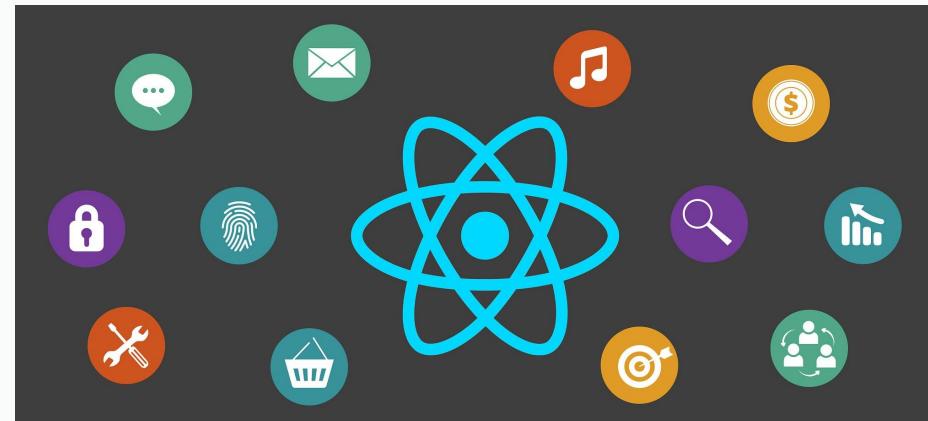
Code getting too Complex

# React: The Savior of Web Development

In 2011, Jordan Walker at Facebook, tired of the ongoing frustrations, set out to find a better way. He envisioned a tool that could make the development process smoother and more efficient.



Jordan Walker



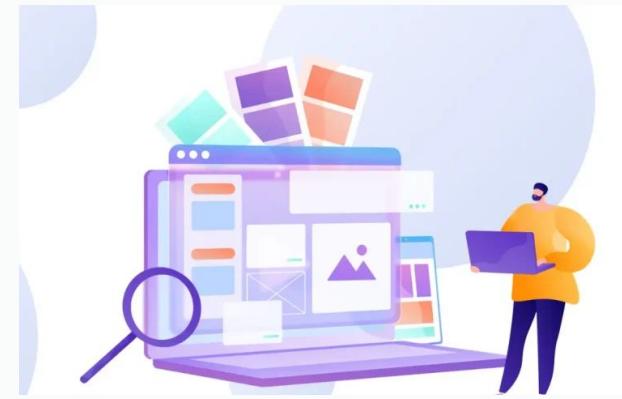
React making life easy for developers

# What is React?

React is a JavaScript library for building user interfaces, maintained by Facebook.



Library for building web interfaces



Build beautiful interfaces with react

# Why use React?

React is most widely library for building single page websites. For the following reasons:-



Reduced Development  
Costs



Performance



Improved UI/UX



Code Reusability



Support Third-Party  
Libraries

# Key Features of React

The following are the key features of react:-



React JS is an  
**Open-source** Javascript  
library under MIT license



**component-based**, and  
high-performance  
platform



It is used for interactive  
and **responsive user  
interfaces**(UIs) of  
websites and web apps



More than **117,667**  
stars on GitHub

Don't worry if you don't  
understand few of the  
terms, we will explain  
each of them throughout  
our course.

# Popular websites built on React

The following are few popular websites which uses React:-

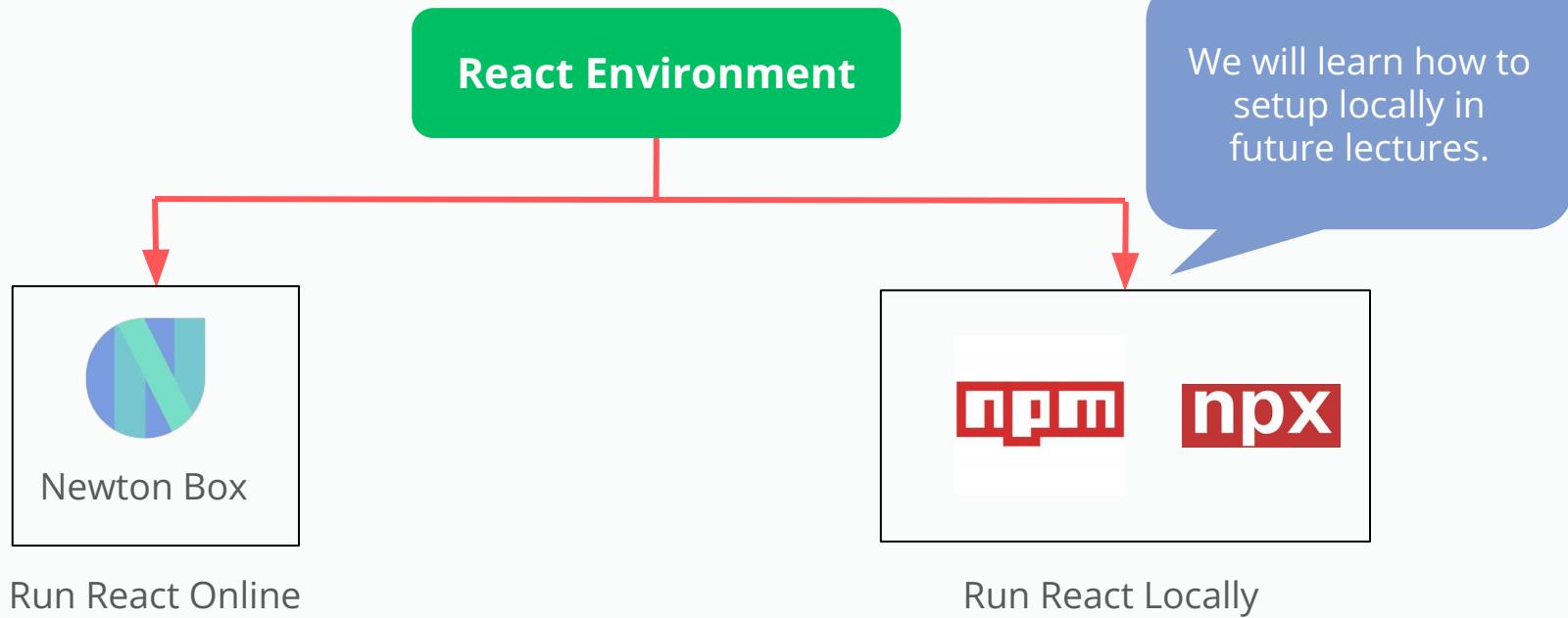


Have you identified  
your dream company  
among these??

# Setting up a React Project

# How to Run React Code

React applications can be run in various ways, depending on the development setup and requirements.

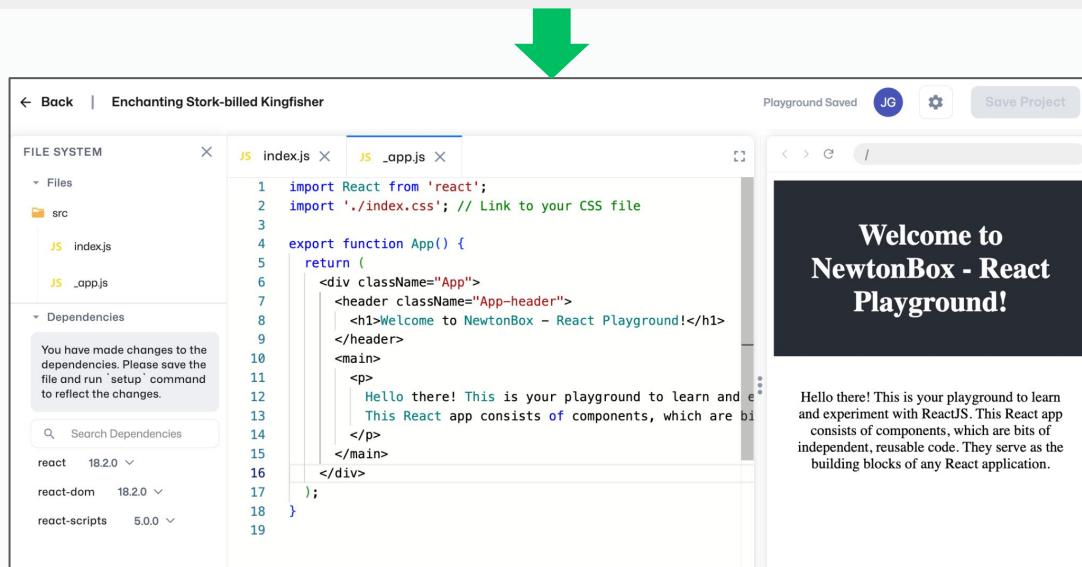


# Run React Online: Newton Box Playground

We can run React code effortlessly without worrying about setup by using remote tools. Let's explore how to run React code using Newton Box.

<https://my.newtonschool.co/playground/newton-box/3mmvu006csv0>

Click on this Link  
to open Newton  
Box



The screenshot shows the Newton Box playground interface. On the left, there is a file system panel with a 'src' folder containing 'index.js' and '\_app.js'. A message in the dependencies section says: 'You have made changes to the dependencies. Please save the file and run 'setup' command to reflect the changes.' Below this, dependency versions are listed: react 18.2.0, react-dom 18.2.0, and react-scripts 5.0.0. In the center, there is an editor window with the following code:

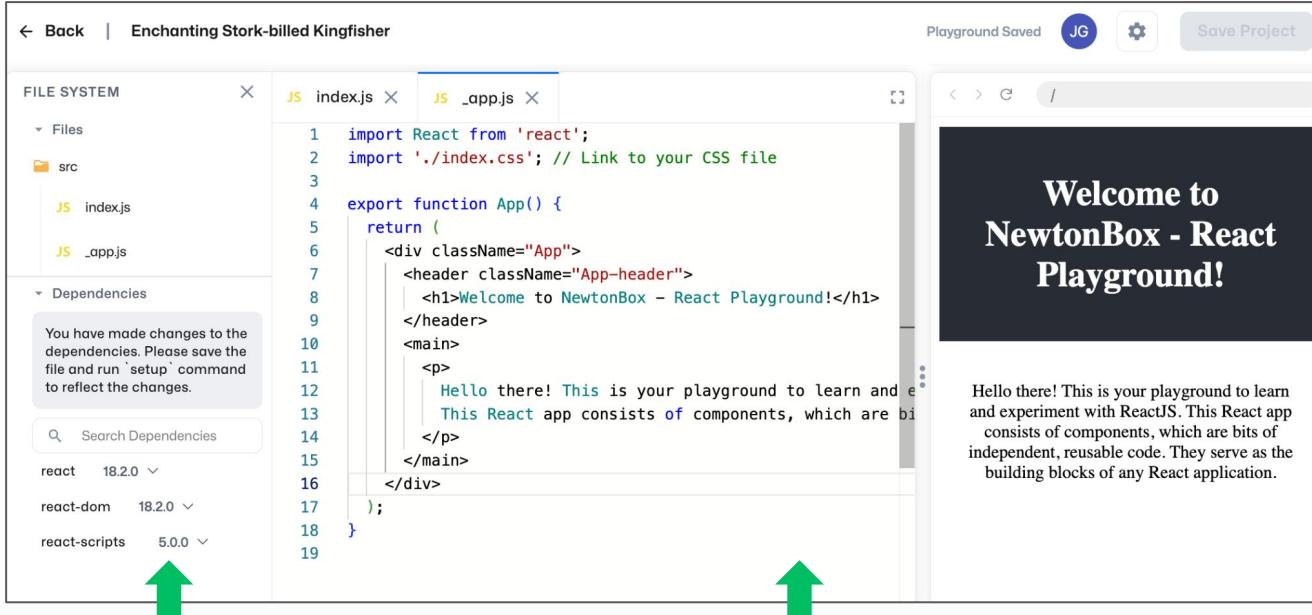
```
1 import React from 'react';
2 import './index.css'; // Link to your CSS file
3
4 export function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <h1>Welcome to NewtonBox - React Playground!</h1>
9       </header>
10      <main>
11        <p>
12          Hello there! This is your playground to learn and e
13          This React app consists of components, which are bi
14        </p>
15      </main>
16    </div>
17  );
18}
19
```

To the right of the editor is a preview window showing the rendered output: 'Welcome to NewtonBox - React Playground!'. Below the preview, a descriptive text reads: 'Hello there! This is your playground to learn and experiment with ReactJS. This React app consists of components, which are bits of independent, reusable code. They serve as the building blocks of any React application.'

In Newton Box you can directly write React Code and view its preview in side browser.

# Run React Online: Newton Box

Let's analyse UI of Newton Box:-



The screenshot shows the Newton Box React Playground interface. On the left, there's a file system panel with 'index.js' and '\_app.js' selected. Below it, a 'Dependencies' section lists 'react', 'react-dom', and 'react-scripts'. A message box says: 'You have made changes to the dependencies. Please save the file and run 'setup' command to reflect the changes.' At the bottom, there are two green arrows pointing upwards: one from a blue rounded rectangle to the dependencies section, and another from a green rounded rectangle to the code editor area.

```

import React from 'react';
import './index.css'; // Link to your CSS file

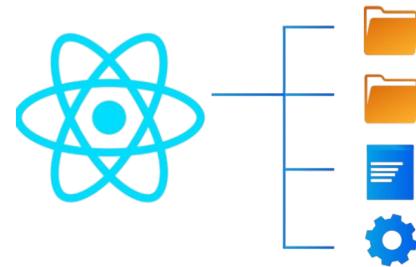
export function App() {
  return (
    <div className="App">
      <header className="App-header">
        <h1>Welcome to NewtonBox - React Playground!</h1>
      </header>
      <main>
        <p>
          Hello there! This is your playground to learn and experiment with ReactJS. This React app consists of components, which are bits of independent, reusable code. They serve as the building blocks of any React application.
        </p>
      </main>
    </div>
  );
}


```

Here you can open Project files/folders and install dependencies

Here you will write your React code

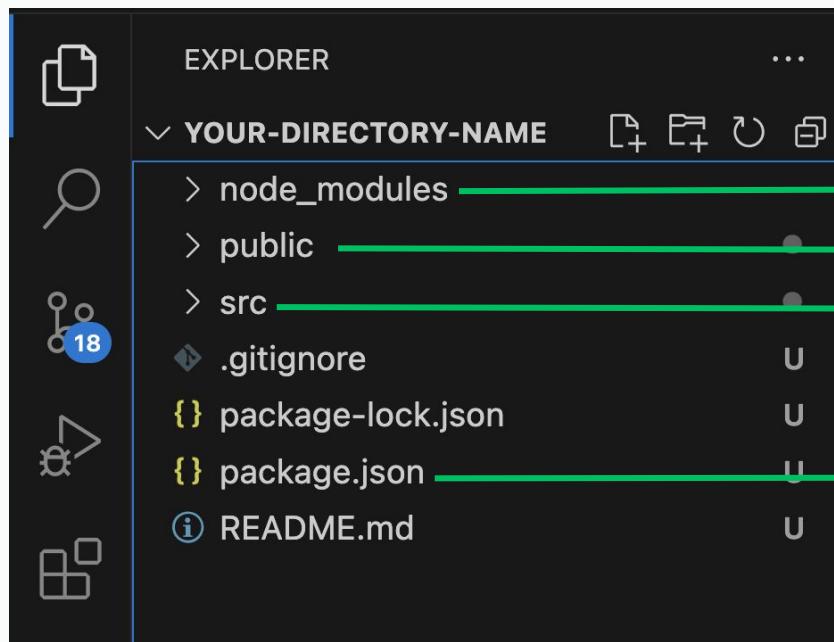
In built browser to see the changes



# Understanding Project Structure

# Project Structure

In file explorer you can see multiple folders and files being created, let's understand each one by one:-



Contains all third-party dependencies installed via npm or yarn.

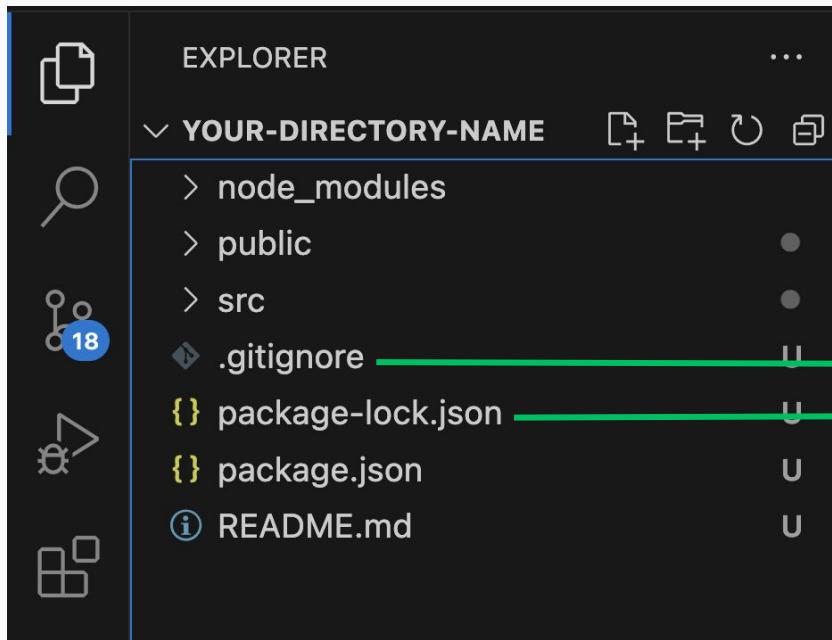
Contains static files directly served by the browser.

Contains all the application code

Lists dependencies, scripts, and project metadata

# Project Structure

Here we have two more files:-

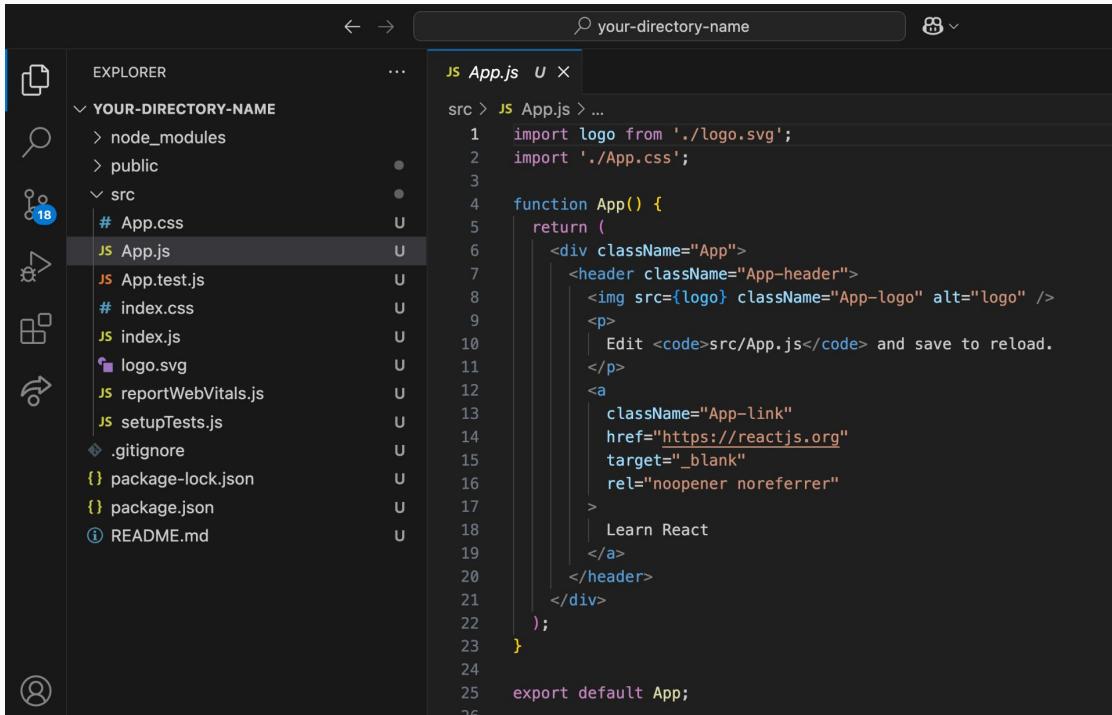


Here you can specify which files needs to be ignored by git.

Its primary purpose is to ensure consistent installations across different environments. You would be rarely updating this file manually.

# Project Structure: src folder

Let's open the src folder and see its contents:-



The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Shows the project directory structure:
  - YOUR-DIRECTORY-NAME
  - node\_modules
  - public
  - src
    - # App.css
    - JS App.js** (selected)
    - JS App.test.js
    - # index.css
    - JS index.js
    - logo.svg
    - JS reportWebVitals.js
    - JS setupTests.js
  - .gitignore
  - { package-lock.json
  - { package.json
  - README.md
- App.js** file content (highlighted in the code editor):

```
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;
```

Here, you'll find multiple files. App development begins with App.js, the main component of the application. The App.css file is used to style and enhance the visual appearance of the content in App.js.

# Project Structure: package.json

📦 `package.json` in a React project is a file that manages project settings, dependencies, and scripts to run, build, and test the app.

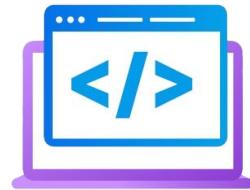
```
1  {
2    "name": "your-directory-name",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "cra-template": "1.2.0",
7      "react": "^19.0.0",
8      "react-dom": "^19.0.0",
9      "react-scripts": "5.0.1"
10    },
11    "scripts": {
12      "start": "react-scripts start",
13      "build": "react-scripts build",
14      "test": "react-scripts test",
15      "eject": "react-scripts eject"
16    },
17    "comment": {
18      "comment": "rest of the json data"
19    }
20 }
```

Dependencies installed automatically with React

Script commands with an alias for npm

We can execute these scripts with npm:-

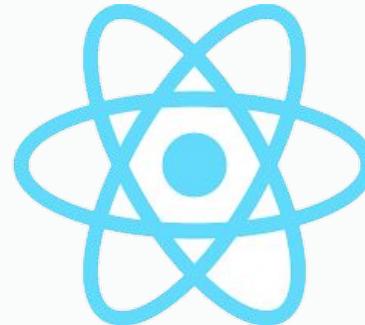
*npm start  
npm build  
.....*



# Functional Components

# React Functional Components

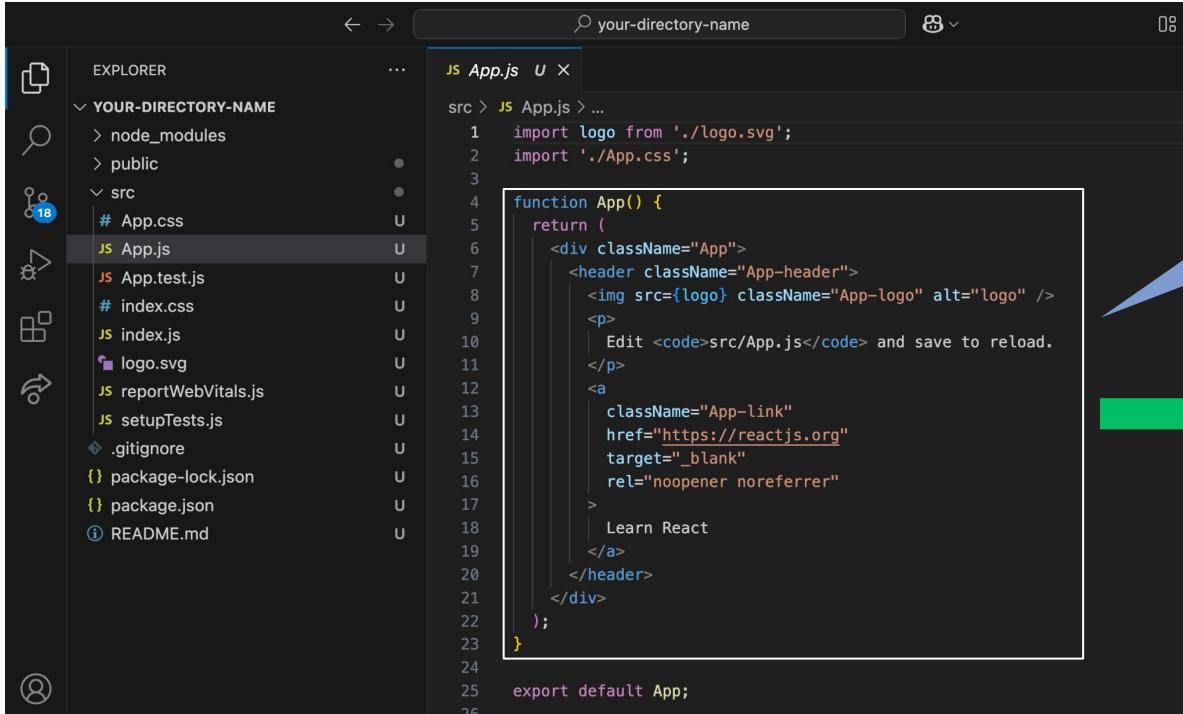
React functional components are JavaScript functions that return JSX to define and render UI elements in a React application.

$$f(x)$$


Just like the math functions, React functional components execute logic and return UI elements

# Creating Functional Components

The default code inside `App.js` is actually a functional component:-



A screenshot of the VS Code interface showing the file structure on the left and the code editor on the right. The code editor displays the following content:

```
your-directory-name
JS App.js U X
src > JS App.js > ...
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;
```

We need to clean up a bit to include our custom code.

Functional component

# Cleaning up Template Code

We would like to create our own functional components so remove template code and start including our own code instead of react template provided by

```
JS App.js U ●
src > JS App.js > ...
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   export default App;
25
```

Let's redact our template code

And add  
*Hello, World!*

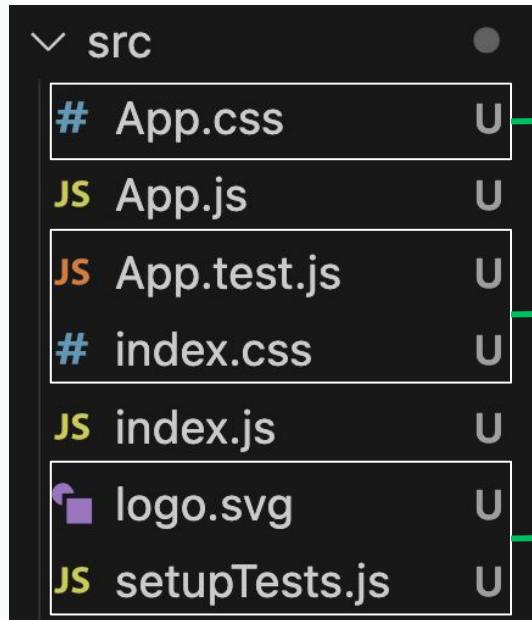
```
function App() {
  return (
    <div>
      Hello, World!
    </div>
  );
}

export default App;
```

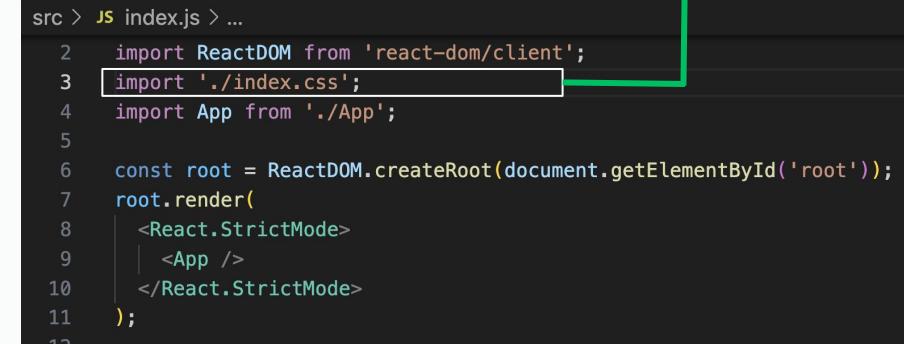
Hello, World!

# Cleaning Up template Code

Let's also delete unnecessary files from the src folder:-



Removed



```
src > JS index.js > ...
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5
6 const root = ReactDOM.createRoot(document.getElementById('root'));
7 root.render(
8   <React.StrictMode>
9     | <App />
10    </React.StrictMode>
11 );
```

The code for the 'index.js' file is shown. It imports 'ReactDOM' and 'App' from their respective files. It then creates a root element using 'ReactDOM.createRoot' and renders the 'App' component in 'StrictMode'. A red box highlights the line 'import './index.css';'. An arrow points from this box to a callout bubble.

index.js file

Also remove index.css from index.js file as we have deleted this file

# Cleaning Up Template Code

Inside *index.js* *reportWebVitals.js* file was throwing error, so we removed it from import statement as well as at the bottom of the file where it was being invoked:-

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    | <App />
  </React.StrictMode>
);

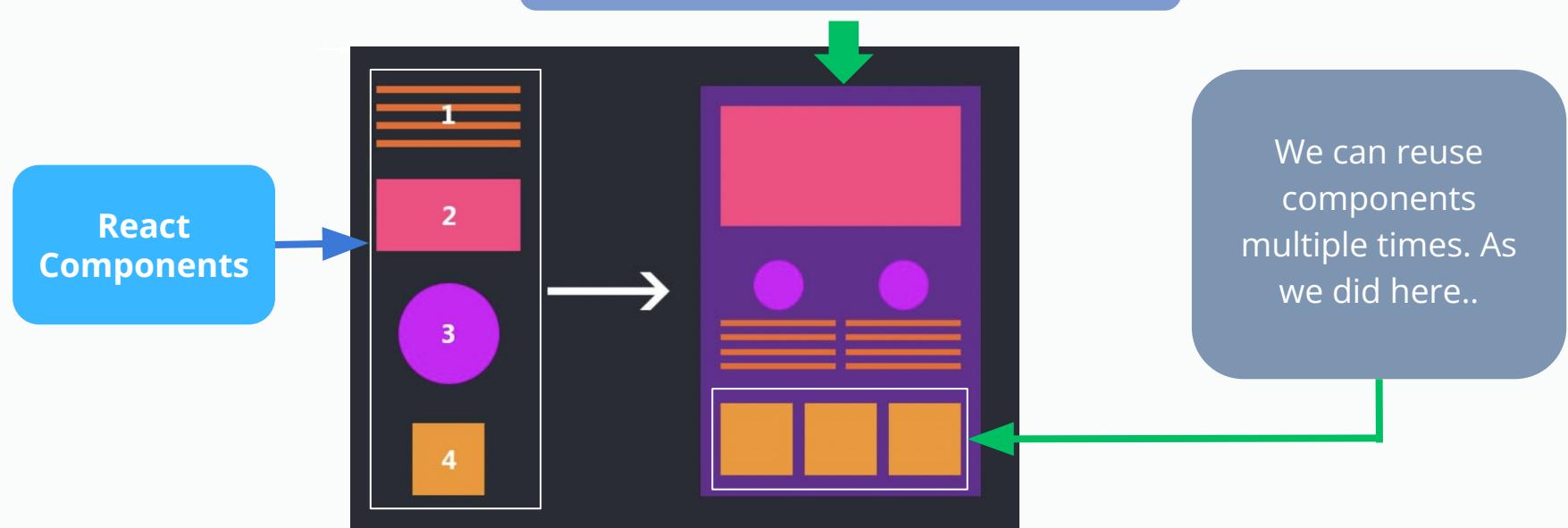
// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Removed  
reportWebVitals

# Component based structure in React

React follows a component based structure, where you built the components once and use them in your code.

React Web page we created from React components



# Let's create some new components

Just like App.js which is the root component, we can build other components. For now we will build functional components.

*We will build these components:-*



# Creating Component: Header.jsx

For now we will create components in the same file i.e. App.js:-

```
const Header = () => {
  return <h1>Welcome to My React App</h1>;
};
```

Header Component

Here we have created a simple header component with a simple message.

# Including Header in App.js

We can simply use Header component previously written in App.js file and use it like this:-

```
4
5 const App = () => {
6   return (
7     <div>
8       <Header /> |
9       <p>This is the main content of the app.</p>
10    </div>
11  );
12};
13
14 export default App;
```



We can include our components using custom JSX tags, similar to HTML elements.

# Let's see how it appears in the browser

Let's have a look at side browser:-

```
4
5 const App = () => {
6   return (
7     <div>
8       <Header />
9       <p>This is the main content of the app.</p>
10      </div>
11    );
12  };
13
14 export default App;
```



## Welcome to My React App

This is the main content of the app.

# Creating Footer Component

Similarly we can create Footer component and use it in inside App like this:-

```
const Footer = () => {
  return (
    <p>
      © 2025 My React App.
      All rights reserved.
    </p>
  );
};
```

```
6  const App = () => {
7    return (
8      <div>
9        <Header />
10       <p>This is the main content of the app.</p>
11       <Footer />
12     </div>
13   );
14 };
15
16 export default App;
```



# Let's have a look of it in the browser

Let's open the browser and see how it looks:-

## Welcome to My React App

This is the main content of the app.

© 2025 My React App. All rights reserved.

Similarly we can create as many components needed and use them throughout the project

# Styling our React Components

We only have introduced how to write plain JSX, but it looks a bit off. To make it more appealing we can style it using CSS.

## Welcome to My React App

This is the main content of the app.

© 2025 My React App. All rights reserved.



So boring!!

## Welcome to My React App

This is the main content of the app.

© 2025 My React App. All rights reserved.



Wow! It looks way better with **CSS**

# How to Style our Component - Inline?

To write inline styles we use style attribute and write our style inside two curly braces:-

```
3
4  const Header = () => {
5      return (
6          <h1 style={{ }}>
7              Welcome to My React App</h1>
8      );
9  };
10
11 export default Header;
```

Style tag is used to include our styles and unlike html, we include styles inside double curly braces {{ }}



Why two curly braces {{...}}}

# Styling Component: { } vs {{ }}

Use single curly braces when you are using javascript variable to feed styles and double when you are writing styles directly.

```

3
4 const Header = () => {
5   const headerStyle = {
6     backgroundColor: "#4CAF50",
7     color: "white",
8     textAlign: "center",
9     padding: "20px",
10    fontSize: "28px",
11    fontWeight: "bold",
12    borderRadius: "10px"
13  }
14  return (
15    <h1 style={headerStyle}>
16      Welcome to My React App
17    </h1>
18  );
19};

```

Just like any  
JSX variable,  
we enclose  
headerStyle in  
curly braces {}

```

3
4 const Header = () => {
5   return (
6     <h1 style={{ 
7       background: "#4CAF50",
8       color: "#fff",
9       textAlign: "center",
10      padding: "20px",
11      fontSize: "28px",
12      fontWeight: "bold",
13      borderRadius: "10px",
14    }}>
15      Welcome to My React App
16    </h1>
17  );
18};
19
20 export default Header;

```

We first create a style object {} to store our styles. Then, in JSX, we wrap it inside {} again to apply it, just like using any JavaScript variable.

# How to Style our Component - External?

Using inline styles everywhere can clutter the code, so we often use an external CSS file. We define styles with class names, import the file, and apply the classes in components using `className`.

```

2  import "./Header.css"; ←
3
4  const Header = () => {
5      return (
6          <h1 className="header">
7              Welcome to My React App
8          </h1>
9      );
10 }
11

```

Header.jsx

Using `.header` class

**Note:** Instead of using class we use `className` for *classes*

```

1 .header {
2     background: #4CAF50;
3     color: #fff;
4     text-align: center;
5     padding: 20px;
6     font-size: 28px;
7     font-weight: bold;
8     border-radius: 10px;
9 }

```

Header.css

Importing `Header.css` file in our file.

# Have you observed Camel Casing?

In our component which is JSX, style properties use **camelCase** instead of kebab-case.

```
const headerStyle = {  
  backgroundColor: "#4CAF50",  
  color: "white",  
  textAlign: "center",  
  padding: "20px",  
  fontSize: "28px",  
  fontWeight: "bold",  
  borderRadius: "10px"  
}
```

uses

camelCase	kebab-case
backgroundColor	background-color
textAlign	text-align
fontSize	font-size
borderRadius	border-radius

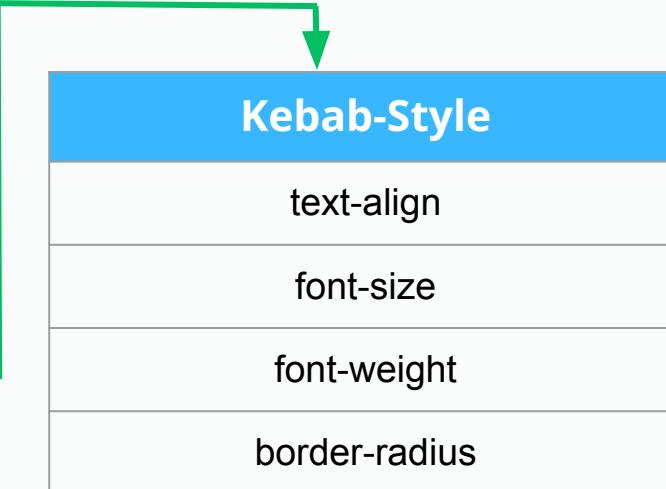
Don't worry, We will learn JSX in next lecture.

# Kebab-Style works outside Components

However you can use kebab-style outside the component files, like .css files.



```
1 .header {  
2   background: #4CAF50;  
3   color: #fff;  
4   text-align: center;  
5   padding: 20px;  
6   font-size: 28px;  
7   font-weight: bold;  
8   border-radius: 10px;  
9 }
```



# Why it is called camelCase?

It's called **camelCase** because the capital letters in the middle create "humps" like a camel! 🐫



Look closely... see any humps? 😊

Were you able to guess?



# In Class Questions

# References

1. **React Docs:** <https://react.dev>
2. **MDN React Guide:** Comprehensive and beginner-friendly documentation for React  
[https://developer.mozilla.org/en-US/docs/Learn/Tools\\_and\\_testing/Client-side\\_JavaScript\\_frameworks/React\\_getting\\_started](https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started)
3. **Websites and Tutorials:**
  - a.  w3schools: <https://www.w3schools.com/REACT/DEFAULT.ASP>
  - b.  codecademy: <https://www.codecademy.com/learn/react-101>
4. **Other Useful Resources**
  - a.  React GitHub Repository: <https://github.com/facebook/react>
  - b.  React Patterns & Best Practices: <https://reactpatterns.com/>

**Thanks  
for  
watching!**