**REACT**

Table of Contents

[**What is react?** 2](#_Toc164719190)

[1. React 2](#_Toc164719191)

[2. Creating React Application 3](#_Toc164719192)

[3. Folder Structure 3](#_Toc164719193)

[4. First Code 3](#_Toc164719194)

[5. JSX 4](#_Toc164719195)

[**React Components** 5](#_Toc164719196)

[1. Functional Components 5](#_Toc164719197)

[2. Components Nesting 6](#_Toc164719198)

[3. Conditionals 6](#_Toc164719199)

[4. Styling 7](#_Toc164719200)

[5. Class Components 7](#_Toc164719201)

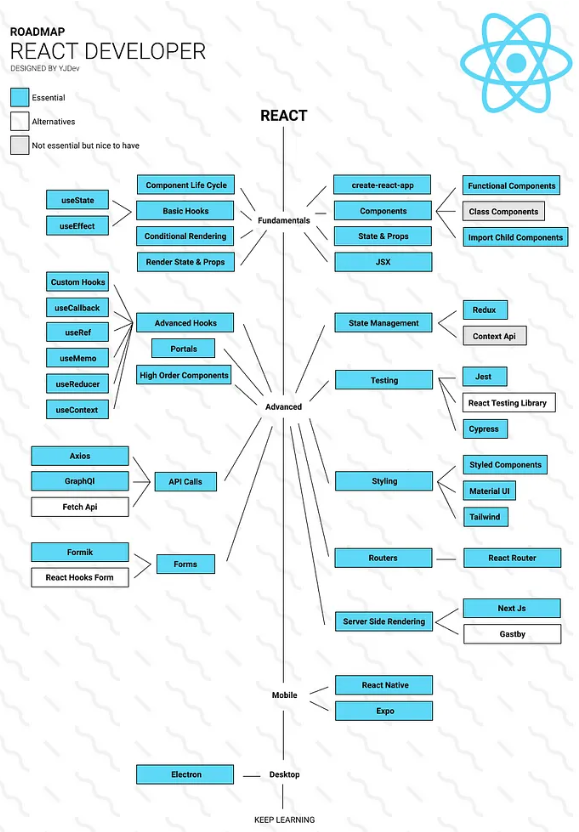
[6. Project 1 8](#_Toc164719202)

# **What is react?**

## React

React is a declarative, efficient and flexible JavaScript library for building user interface. It lets you compose complex UI’s from small and isolated pieces of code called “components”.

It helps you to make SPA(single page application), SPA is a web app implementation that loads only a single web document and then updates the body content of that single document via JavaScript APIs when different content is to be shown.



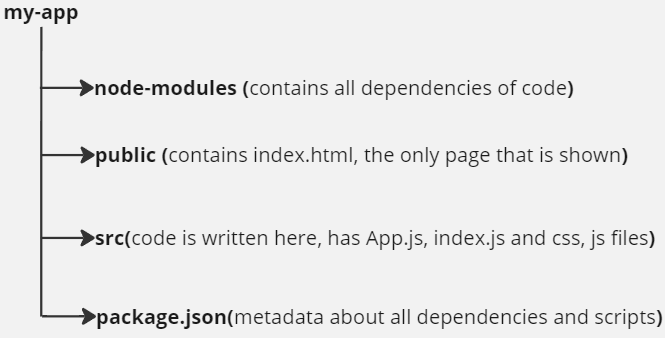
## Creating React Application

npx create-react-app my-app

cd my-app

npm start   #to run the app

## Folder Structure



While sending this code we don’t transfer node-modules folder, so when we use someone else code, to install the node-modules we do :- **npm install**

## First Code

//index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

// Index.js is actual file where our final code is called, and displayed in main screen

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

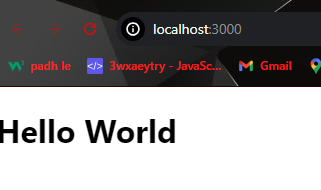
  <React.StrictMode>

    <h1>Hello World</h1>

  </React.StrictMode>

);

To run this do:- npm start

Output:- 

## JSX

* JSX allows you to put HTML into JavaScript
* It stands for JavaScript XML
* React has Babel dependency which compiles HTML and JavaScript from JSX

JSX Rules:-

1. Return single element from one parent element(Components)

Use <> </> to group different elements

1. camelCase property naming convention

return (

  <div tabIndex={1}>

    <button onClick={myFunction}>click me</button>

    <label htmlFor='name'>Name</label>

    <input readOnly={true} id='name' />

  </div>

)

1. Every tag should be closed, eg can’t have <img> , but do <img/>
2. To call variables inside HTML, use { }

Embedding expression in JSX

//App.js

import './App.css';

function App() {

  const heading = "Hello World";

  return (

    <div className="App">

      <h2>My first code is : {heading} </h2>

    </div>

  );

}

export default App;

This App.js is rendered in index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

# **React Components**

## Functional Components

* Components lets you split the UI into independent, reusable pieces, and think about each piece in isolation.
* Conceptually, components are like JavaScript function. They accept the arbitrary inputs ( called props) and return React elements describing what should appear on the screen.
* Function name should begin with Capital letter (Pascal case)
* Put all components in one component folder and call them in App.js for better arrangements of code.
* To call function name Header, do <Header/>

First Component

//Header.jsx

import React from "react";

function Header(){

    return (

        <>

        <h1>Hello Everyone</h1>

        <p>This is my first component</p>

        </>

    )

}

export default Header;

Component calling

//App.js

import './App.css';

import Header from './components/Header';

function App() {

  return (

    <div className="App">

      {/\* Component calling \*/}

      <Header/>

    </div>

  );

}

export default App;

## Components Nesting

Can call functions within same module, to nest them.

import React from 'react';

function BookList() {

  return (

    <section>

      <Book />

      <Book />

      <Book />

    </section>

  );

}

//Components Nesting

const Book = () => {

  return (

    <article>

      <Image />

      <Title />

      <Author />

    </article>

  );

};

//Different ways to make Components

const Image = () => <h2>image placeholder</h2>;

const Title = () => {

  return <h2>Book Title</h2>;

};

const Author = () => <h4>Author</h4>;

export default BookList;

## Conditionals

We can use If-else conditions in react components

import React from 'react';

const Greetings = () =>{

    const date = new Date();

    var x = date.getHours();

    var header;

    if(x<12)   {

        header = "Morning"

    }else{

        header = "Afternoon"

    }

    return(

        <p>Good {header} Everyone !!</p>

    )

}

export default Greetings;

Can also use operators like && and ||.

Can use ternary operators.

Eg:

{isLoggedIn ? <h1>Hello</h1> : <p> Login Here !</p>}

## Styling

* Can add style for all in style.css and since it is imported in index.js, style applies to all files. Since index.js is the file that does rendering of all components.
* To apply css at each file separately, create a css file and import as:- **import ‘./header.css’**
* All properties names in jsx become camelCase, Eg: class 🡪 className, font-size🡪fontSize . Applicable only when css is written directly in jsx file
* For inline style, first make it as object, and then use that style

Eg: <h1 style = {{ color: “red” }} > </h1>

import React from 'react';

function Header(){

    const paraStyle ={

        fontFamily:"fantasy",

        fontSize:"130px"

    }

    return(

        <div>

        {/\* Two {{, inside{} is for creating object, outer is to call object \*/}

        <h1 style = {{color:"blue", fontWeight:"bold" , fontSize:"30px" }}>Hello Everyone</h1>

        <p style={paraStyle}>I am a stylish text</p>

        </div>

    )

}

export default Header;

## Class Components

Till now, this were functional components, but there is one more way to create components using class. They were used before hooks were introduced to react.

Eg:

class Welcome extends React.Component{

    render(){

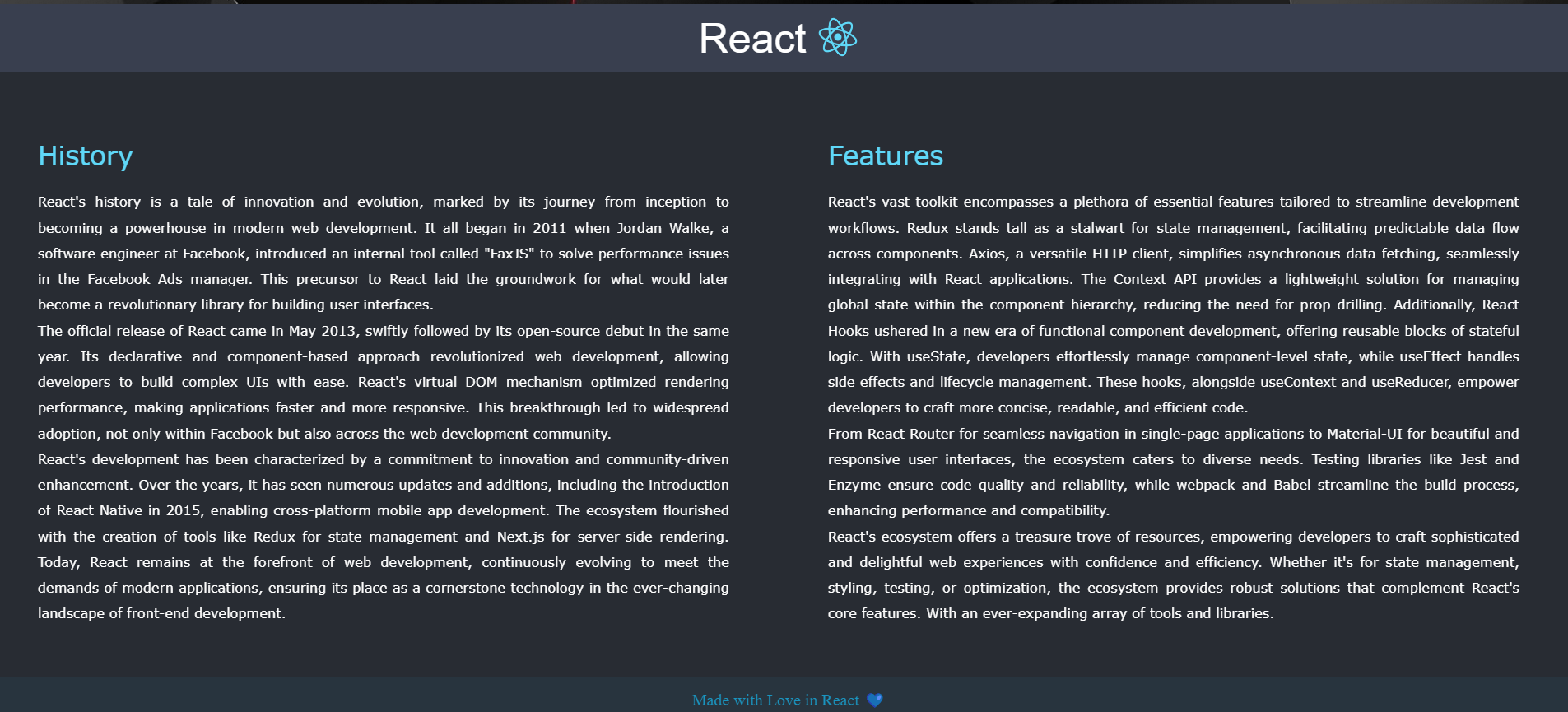
        return <h1>Hello World !!</h1>

    }

}

## Project 1

Create a basic single page website with header, footer and body content showing react logo and history about react.



Solution: