# MODULE: 9 ReactJs

* **What is React Js?**

**Ans :** ReactJS is a JavaScript library used for building reusable UI components.

According React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook.

React is a tool for building UI components.

Below are the few features of React. JSX is a combination of HTML and JavaScript. You can embed JavaScript objects inside the HTML elements. JSX is not supported by the browsers, as a result, Babel compiler transcompile the code into JavaScript code. JSX makes codes easy and understandable. It is easy to learn if you know HTML and JavaScript.

History of React :

ReactJS is updated so frequently, that it is quite difficult to navigate the version, which comes with new features every time, each time it comes with new features. The current stable version of ReactJS is 18.2.0 and released on June 14, 2022 and the first release was on May 29, 2013.

**• What is NPM in React Js?**

**Ans :** So now let's talk about what exactly npm is. What is NPM? NPM – or "Node Package Manager" – is the default package manager for JavaScript's runtime Node.js. It's also known as "Ninja Pumpkin Mutants", "Nonprofit Pizza Makers", and a host of other random names that you can explore and probably contribute to over at npm-expansions.

* **What is Role of Node Js in react Js?**

**Ans :** Using Node. js and React together empowers web applications to handle heavy server loads and requests. Therefore, it allows you to work easily throughout the development process.

React.js is a front-end library for building user interfaces (UI). It focuses on creating reusable components that update efficiently based on data changes.

Node.js, on the other hand, is a runtime environment that allows you to run JavaScript code outside of a web browser. This makes it suitable for building server-side applications and APIs.

Backend Server: Node.js is a popular choice for building the backend server for a React application. It can handle tasks like handling user requests, managing databases, and serving static assets. Frameworks like Express.js are often used on top of Node.js to simplify backend development.

API Development: Node.js excels at building APIs (Application Programming Interfaces) that provide data to the React frontend. These APIs can be in JSON format, which both React and Node.js handle efficiently.

* **What is CLI command In React Js?**

**Ans :** React has its own command-line interface (CLI) commands. However, these CLI commands are currently only used to create a passable version of a react application using the command line. This will contain a default template as its design, so all the react application created this way will have great consistency as they all have same structure.

Create React App (CRA): This is the most popular tool for setting up new React projects. It provides a standardized structure, installs essential dependencies, and offers commands to start the development server and build the application for production. You can use npm create-react-app my-app to create a new React project named "my-app".

* **What is Components in React Js?**

**Ans :** Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.

Components come in two types, Class components and Function components, in this tutorial we will concentrate on Function components.

A Component is one of the core building blocks of React. They have the same purpose as JavaScript functions and return HTML. Components make the task of building UI much easier.

A UI is broken down into multiple individual pieces called components. You can work on components independently and then merge them all into a parent component which will be your final UI.

**Functional Component in React :**

Functional components are just like JavaScript functions that accept properties and return a React element.We can create a functional component in React by writing a JavaScript function. These functions may or may not receive data as parameters, we will discuss this later in the tutorial. The below example shows a valid functional component in React:

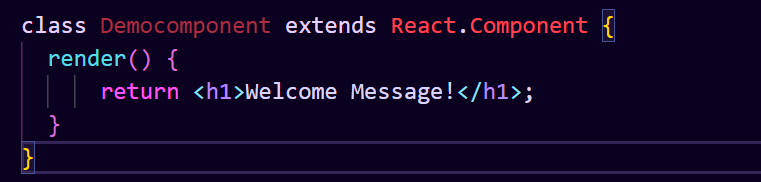
**Syntax for Functional Components:**



**Class Component in React :**

The class components are a little more complex than the functional components. A class component can show inheritance and access data of other components.Class Component must include the line “extends React.Component” to pass data from one class component to another class component. We can use JavaScript ES6 classes to create class-based components in React.

**Syntax for Class Components:**

****

* **What is Header and Content Components in React Js?**

**Ans :**

**Header Component:**

A header component typically represents the top section of a webpage and often includes elements like:

Application logo or title

Navigation bar with links to different sections of the application Search bars or login/logout functionality

Here are some key points about header components:

Provides a consistent visual identity across the application.

Improves user experience by offering easy access to navigation and other essential features.

Can be styled differently to match the application's branding.

You can create a reusable header component that can be imported and used throughout your React application.

**Content Component:**

A content component represents the main content area of a webpage where the core functionality or information is displayed. The content can vary depending on the specific page or section of your application. Here are some examples:

Product listings on an e-commerce site

Blog post content on a blog application

User profile information on a social media platform

Key points about content components:

Responsible for displaying the relevant information or functionality for each page.

Can be dynamic, updating based on user interactions or data changes.

May contain sub-components for further organization (e.g., article components within a blog post content component).

**• How to install React Js on Windows, Linux Operating System? How to Install NPM and How to check version of NPM?Ans :** To publish and install packages to and from the public npm registry or a private npm registry, you must install Node.js and the npm command line interface using either a Node version manager or a Node installer. We strongly recommend using a Node version manager like nvm to install Node.js and npm. We do not recommend using a Node installer, since the Node installation process installs npm in a directory with local permissions and can cause permissions errors when you run npm packages globally.

to download the latest version of npm, on the command line, run the following command:

npm install -g npm

Checking your version of npm and Node.js :

then check version

cmd : node -v

cmd: npm -v

Linux or other operating systems Node installers

If you're using Linux or another operating system, use one of the following installers:

NodeSource installer (recommended)

One of the installers on the Node.js download page

Or see this page to install npm for Linux in the way many Linux developers prefer.

* **How to check version of React Js?**

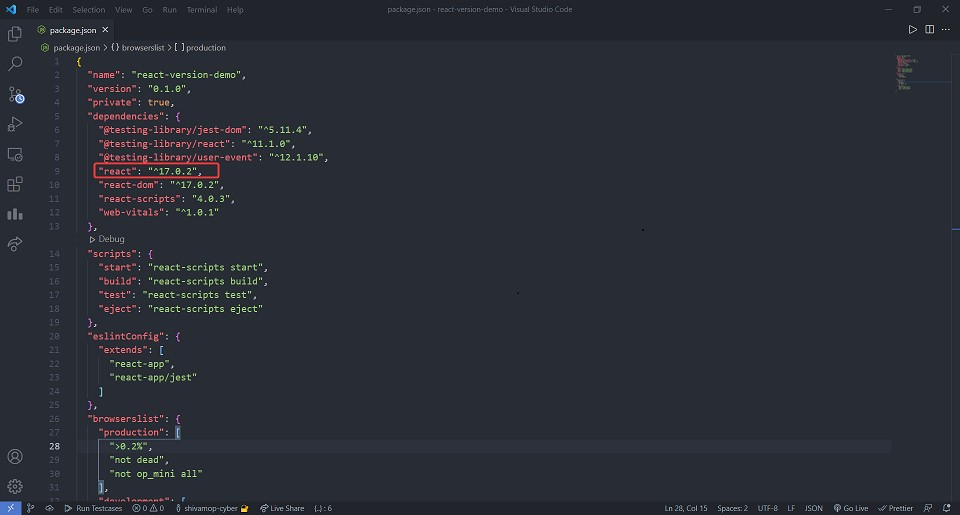
**Ans :**

1: Using the package.json file :

The package.json contains metadata about our project. It is created by default when we create our React project. We can create a react app using the command mentioned below.

**npm create-react-app name\_of\_the\_app**

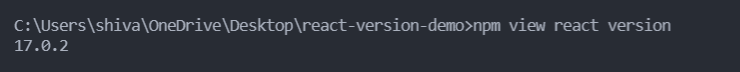
The package.json file contains a lot of information in the name/value pairs in JSON format. We can easily check our React version under the list of dependencies as shown in the image given below.



2: Using the command line :

We can easily check the React version by using the command mentioned below on our command line.

**npm view react version**



3: Using the version property of default import from React :

The default import from React library is an object that has a version property on it. We can use this property inside our JSX elements in our desired manner.

**import React from 'react';**

**let a = React.version**

**import React from "react";**

**const App = () => {**

**return (**

**<h1>**

**We are currently using react version{" "}**

**{React.version}**

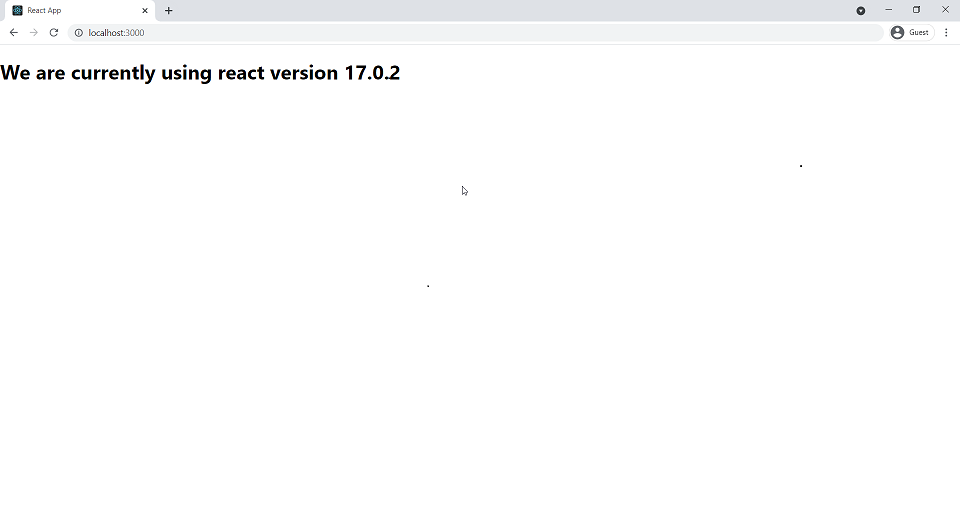
**</h1>**

**);**

**};**

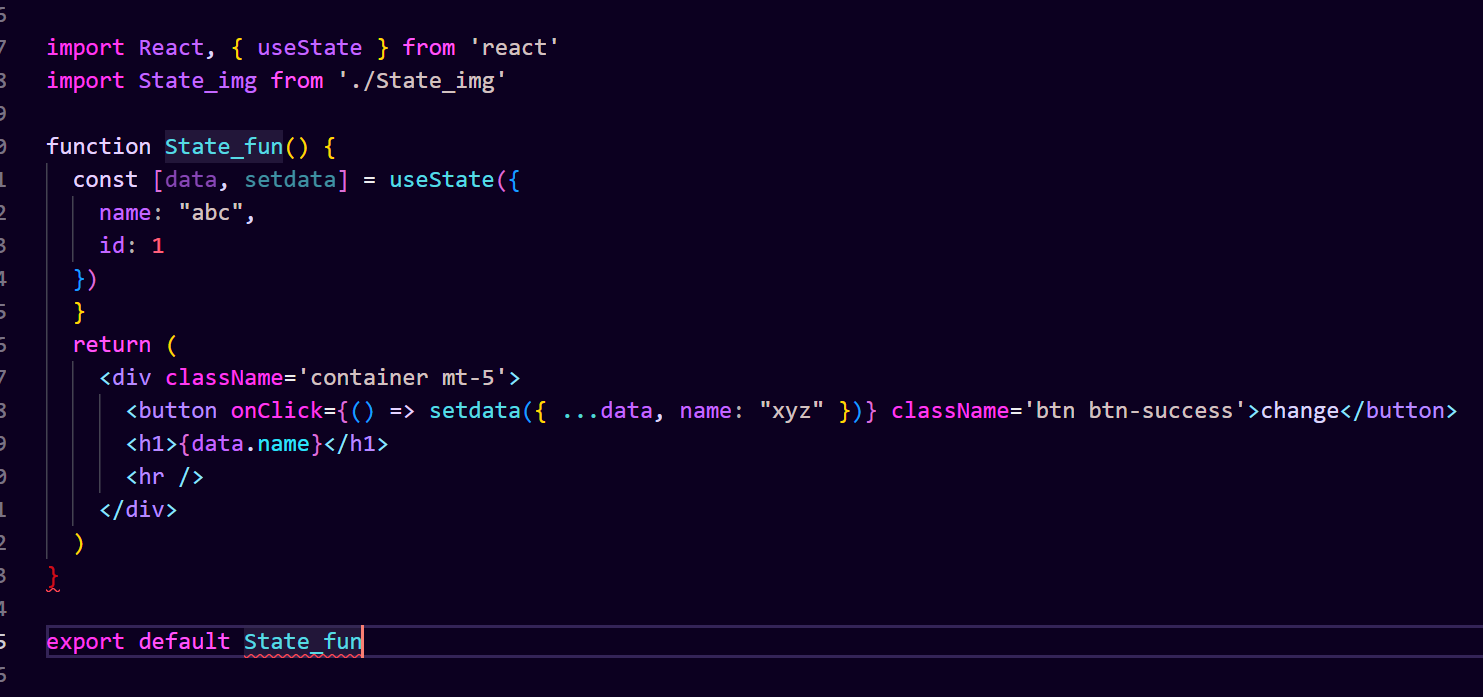
**export default App;**

**npm start**



* **How to change in components of React Js?**

**Ans :** We have to set initial state value inside constructor function and set click event handler of the element upon which click, results in changing state. Then pass the function to the click handler and change the state of the component inside the function using setState**.**

****

* **Explain Life cycle in Class Component and functional component with Hooks**

**Ans :**

React lifecycle method explained :

Ever since React came to be, back in 2013, React developers used class components to take full use of the React library (extending from React.Component) in order to manipulate DOM in a React-based app development. How does the React component class approach work?

So, let’s take a look at how it’s been done traditionally. To do that, we’re going to take a closer look at React components.

As you probably know, each React component instance has a lifecycle. The component’s lifecycle consists of three phases:

* Mounting lifecycle methods, that is inserting elements into the DOM.
* Updating, which involves methods for updating components in the DOM.
* Unmounting, that is removing a component from the DOM.

1. **Mounting in the React component lifecycle :**

As we mentioned, during the mounting phase of the lifecycle, the class component is inserted into the DOM. A good example would be componentDidMount() – a lifecycle method that runs after the component is mounted and rendered to the DOM. It is great when you want to do an interval function or an asynchronous request.

componentDidMount(){} // birth// run when component ready to use

=>Means Component Call means mount / use

1. **Updating in the React component lifecycle :**

The componentDidUpdate() render method is called right after the updating happens. This one is called always except for the initial render. That’s a good place to interact with a non-reactive environment. It’s a good idea to make http requests here.

componentDidUpdate(){} // marriage// run when any update in compo

=>Means Component setState means state update or Update

1. **Unmounting in the React component lifecycle**

componentWillUnmount() is invoked just before the component is removed from the DOM. You should use that to remove event listeners, clear intervals and cancel requests. In other words: all the needed cleanup.

componentWillUnmount(){}// end // run when component remove

=>Means Component Remove from screen means end

**React component lifecycle with hooks :**

You can take advantage of the useEffect hook to achieve the same results as with the componentDidMount, componentDidUpdate and componentWillUnmount methods. useEffect accepts two parameters. The first one is a callback which runs after render, much like in componentDidMount. The second parameter is the effect dependency array. If you want to run it on mount and unmount only, pass an empty array [].

