**React JS Interview Questions and Answers**

1. **What is React?**

React is a popular JavaScript library for building user interfaces (UIs). It was developed by Facebook and is widely used for developing single-page applications, mobile applications, and large-scale web applications.

1. **What are the major features of React?**

* **JSX**

JSX stands for JavaScript XML. It is a JavaScript syntax extension. Its an XML or HTML like syntax used by React JS. This syntax is processed into JavaScript calls of React Framework. It extends the ES6 so that HTML like text can co-exist with JavaScript react code. It is not necessary to use JSX, but it is recommended to use in React JS.

* **Components**

React JS is all about components. React JS application is made up of multiple components, and each component has its own logic and controls. These components can be reusable which help you to maintain the code when working on larger scale projects.

* **One-way Data Binding**

React JS is designed in such a manner that follows unidirectional data flow or one-way data binding. The benefits of one-way data binding give you better control throughout the application. If the data flow is in another direction, then it requires additional features. It is because components are supposed to be immutable and the data within them cannot be changed.

* **Virtual Dom**

A virtual DOM object is a representation of the original DOM object. It works like a one-way data binding. Whenever any modifications happen in the web application, the entire UI is re-rendered in virtual DOM representation. Then it checks the difference between the previous DOM representation and new DOM. Once it has done, the real DOM will update only the things that have actually changed.

* **Simplicity**

ReactJS uses JSX file which makes the application simple and to code as well as understand. We know that ReactJS is a component-based approach which makes the code reusable as your need. This makes it simple to use and learn.

* **Performance**

ReactJS is known to be a great performer. This feature makes it much better than other frameworks out there today. The reason behind this is that it manages a virtual DOM. The DOM is a cross-platform and programming API which deals with HTML, XML or XHTML. The DOM exists entirely in memory. Due to this, when we create a component, we did not write directly to the DOM.

1. **What is JSX?**

JSX stands for JavaScript XML. It is an extension to the JavaScript language syntax that allows you to write HTML-like code within your JavaScript code.

1. **How to create components in React?**

In React, you can create components using two different approaches: function components and class components.

* **Function Components:**

Function components are simple JavaScript functions that accept props (properties) as arguments and return a React element.

**Example:**

function Greeting(props) {

return <h1>Hello, {props.name}!</h1>;

}

* **Class Components:**

Class components are JavaScript classes that extend the React.Component class. They define a render() method to return a React element.

**Example:**

class Greeting extends React.Component {

render() {

return <h1>Hello, {this.props.name}!</h1>;

}

}

1. **What is the difference between state and props?**

|  |  |
| --- | --- |
| **props** | **state** |
| Props are read-only. | State changes can be asynchronous. |
| Props are immutable. | State is mutable. |
| Props allow you to pass data from one component to other components as an argument. | State holds information about the components. |
| Props can be accessed by the child component. | State cannot be accessed by child components. |
| Props are used to communicate between components. | States can be used for rendering dynamic changes with the component. |
| Stateless component can have Props. | Stateless components cannot have State. |
| Props make components reusable. | components reusable. State cannot make components reusable. |
| Props are external and controlled by whatever renders the component. | The State is internal and controlled by the React Component itself. |

1. **What is Virtual DOM?**

A virtual DOM object is a representation of the original DOM object. It works like a one-way data binding. Whenever any modifications happen in the web application, the entire UI is re-rendered in virtual DOM representation. Then it checks the difference between the previous DOM representation and new DOM. Once it has done, the real DOM will update only the things that have actually changed.

1. **What is React JS? Why we use React JS?**

React is a popular JavaScript library for building user interfaces (UIs). It was developed by Facebook and is widely used for developing single-page applications, mobile applications, and large-scale web applications.

The main objective of React JS is to develop User Interfaces (UI) that improves the speed of the apps. It uses virtual DOM (JavaScript object), which improves the performance of the app. The JavaScript virtual DOM is faster than the regular DOM. We can use React JS on the client and server-side as well as with other frameworks. It uses component and data patterns that improve readability and helps to maintain larger apps.

1. **What is state in React?**

In React, state is a built-in feature that allows components to store and manage their own data. It represents the current state of a component and influences its behavior and rendering. State is mutable and can be updated throughout the component's lifecycle.

**Example:**

class Counter extends React.Component {

constructor(props) {

super(props);

this.state = {

count: 0,

};

}

incrementCount() {

this.setState({ count: this.state.count + 1 });

}

render() {

return (

<div>

<p>Count: {this.state.count}</p>

<button onClick={() => this.incrementCount()}>Increment</button>

</div>

);

}

}

1. **What are props in React?**

Props stand for "Properties." They are read-only components. It is an object which stores the value of attributes of a tag and work similar to the HTML attributes.

Props are immutable so we cannot modify the props from inside the component.

**Example:**

// Parent component

function App() {

const name = "John";

const age = 25;

return <ChildComponent name={name} age={age} />;

}

// Child component

function ChildComponent(props) {

return (

<div>

<p>Name: {props.name}</p>

<p>Age: {props.age}</p>

</div>

);

}

**Output:**

Name: John

Age: 25

1. **How to write comments in React?**

In React, you can write comments using JavaScript's standard comment syntax. There are two ways to add comments in React components:

* **Single-line Comments:**

function MyComponent() {

// This is a single-line comment

return <div>Hello, React!</div>;

}

* **Multi-line Comments:**

function MyComponent() {

/\*

This is a multi-line comment

spanning multiple lines.

\*/

return <div>Hello, React!</div>;

}

1. **What are the advantages of React?**

* **Easy to learn and use**

React JS is much easier to learn and use. It comes with a good supply of documentation, tutorials, and training resources

* **Creating Dynamic Web Applications Becomes Easier**

To create a dynamic web application specifically with HTML strings was tricky because it requires a complex coding, but React JS solved that issue and makes it easier. It provides less coding and gives more functionality. It makes use of the JSX(JavaScript Extension), which is a particular syntax letting HTML quotes.

* **Reusable Components**

A ReactJS web application is made up of multiple components, and each component has its own logic and controls. These components are responsible for outputting a small and reusable code helps to make your apps easier to develop and maintain.

* **Performance Enhancement**

ReactJS improves performance due to virtual DOM. The DOM is a cross-platform and programming API which deals with HTML, XML or XHTML. Most of the developers faced the problem when the DOM was updated, which slowed down the performance of the application. ReactJS solved this problem by introducing virtual DOM.

* **The Support of Handy tools**

React JS has also gained popularity due to the presence of a handy set of tools. These tools make the task of the developers understandable and easier. The React Developer Tools have been designed as Chrome and Firefox dev extension and allow you to inspect the React component hierarchies in the virtual DOM.

* **Known to be SEO Friendly**

Traditional JavaScript frameworks have an issue in dealing with SEO. The search engines generally having trouble in reading JavaScript-heavy applications. Many web developers have often complained about this problem. ReactJS overcomes this problem that helps developers to be easily navigated on various search engines.

* **The Benefit of Having JavaScript Library**

Today, ReactJS is choosing by most of the web developers. It is because it is offering a very rich JavaScript library. The JavaScript library provides more flexibility to the web developers to choose the way they want.

1. **Why React uses className over class attribute?**

In React, the className attribute is used instead of the traditional class attribute for specifying CSS classes on elements. This is done to avoid conflicts with the JavaScript class keyword.

**Example:**

function MyComponent() {

return <div className="my-class">Hello, React!</div>;

}

1. **How to use styles in React?**

In React, there are several ways to apply styles to components.

* **Inline Styles:** You can use inline styles by passing a JavaScript object as a value to the style attribute of an element. The object should contain the CSS properties and their corresponding values.

**Example:**

function MyComponent() {

const styles = {

color: 'red',

fontSize: '16px',

fontWeight: 'bold'

};

return <div style={styles}>Hello, React!</div>;

}

* **CSS Modules:** CSS Modules is a popular approach that allows you to write CSS stylesheets as modules and import them into your React components. CSS Modules automatically generate unique class names for each component, avoiding naming collisions.

**Example:**

import styles from './MyComponent.module.css';

function MyComponent() {

return <div className={styles.myClass}>Hello, React!</div>;

}

* **CSS-in-JS Libraries:** There are several CSS-in-JS libraries available that enable you to write CSS styles directly in your JavaScript code. These libraries provide various approaches and syntaxes for defining styles within your components. Some popular libraries include styled-components, Emotion, and CSS Modules with a CSS-in-JS syntax.

**Example:**

import styled from 'styled-components';

const StyledDiv = styled.div`

color: red;

font-size: 16px;

font-weight: bold;

`;

function MyComponent() {

return <StyledDiv>Hello, React!</StyledDiv>;

}

1. **What is the difference between Shadow DOM and Virtual DOM?**

|  |  |
| --- | --- |
| **DOM** | **Virtual DOM** |
| It represents the UI of your application. | It is only a virtual representation of the DOM |
| It updates slow | It updates faster |
| Creates a new DOM if elements updates | Updates the JSX if element updates |
| It allows us to directly target any specific node(HTML element) | It can produce about 200,000 virtual DOM Nodes/Second |
| Can directly update HTML | Can’t directly update HTML |

1. **What are the lifecycle methods of React?**

In ReactJS, every component creation process involves various lifecycle methods. These lifecycle methods are termed as component's lifecycle. These lifecycle methods are not very complicated and called at various points during a component's life.

1. **What are the different phases of component lifecycle?**

**The lifecycle of the component is divided into four phases.**

* Initial Phase
* Mounting Phase
* Updating Phase
* Unmounting Phase

1. **Initial Phase**

* **getDefaultProps():**

It is used to specify the default value of this.props. It is invoked before the creation of the component or any props from the parent is passed into it.

* **getInitialState():**

It is used to specify the default value of this.state. It is invoked before the creation of the component.

1. **Mounting Phase**

* **componentWillMount():**

This is invoked immediately before a component gets rendered into the DOM. In the case, when you call **setState()** inside this method

* **componentDidMount():**

This is invoked immediately after a component gets rendered and placed on the DOM. Now, you can do any DOM querying operations.

* **render():**

This method is defined in each and every component. It is responsible for returning a single root **HTML node** element.

1. **Updating Phase**

* **componentWillRecieveProps():**

It is invoked when a component receives new props. If you want to update the state in response to prop changes, you should compare this.props and nextProps to perform state transition by using **this.setState()**

* **shouldComponentUpdate():**

It is invoked when a component decides updating occurs.

* **componentWillUpdate():**

It is invoked just before the component updating occurs. Here, you can't change the component state by invoking **this.setState()** method.

* **componentDidUpdate():**

It is invoked immediately after the component updating occurs. In this method, you can put any code inside this.

1. **Unmounting Phase**

* **componentWillUnmount():**

This method is invoked immediately before a component is destroyed and unmounted permanently. It performs any necessary **cleanup.**

1. **What is the purpose of render method of react-dom?**

In React, the render method of react-dom is used to render React elements into the DOM (Document Object Model). It is a crucial method for rendering the user interface (UI) components onto the screen

**Example:**

import React from 'react';

import ReactDOM from 'react-dom';

function App() {

return <h1>Hello, React!</h1>;

}

ReactDOM.render(<App />, document.getElementById('root'));

1. **What is the recommended way for naming components?**

In React, naming components follows certain conventions and best practices to promote code readability and maintainability. Here are some recommended ways for naming components:

* **Capitalized CamelCase:** Component names should start with a capital letter and use camel case.

**Example:**

// Good naming convention

function MyComponent() {

// ...

}

* **Descriptive and Intuitive:** Choose names that accurately describe the purpose or functionality of the component.

**Example:**

// Good naming convention

function UserProfile() {

// ...

}

* **Specific and Single Responsibility:** Each component should have a single responsibility or represent a specific part of the user interface.

**Example:**

// Good naming convention

function NavigationBar() {

// ...

}

function UserAvatar() {

// ...

}

1. **How to loop inside JSX?**

In JSX, you can use JavaScript expressions, including loops, to generate dynamic content. Here are a few approaches for looping inside JSX:

* **Using JavaScript map()**

function MyComponent() {

const items = ['apple', 'banana', 'orange'];

return (

<ul>

{items.map((item, index) => (

<li key={index}>{item}</li>

))}

</ul>

);

}

* **Using JavaScript for loop:**

function MyComponent() {

const items = ['apple', 'banana', 'orange'];

const elements = [];

for (let i = 0; i < items.length; i++) {

elements.push(<li key={i}>{items[i]}</li>);

}

return <ul>{elements}</ul>;

}

* **Using JavaScript forEach():**

function MyComponent() {

const items = ['apple', 'banana', 'orange'];

const handleClick = (item) => {

console.log(item);

};

return (

<ul>

{items.forEach((item, index) => (

<li key={index} onClick={() => handleClick(item)}>

{item}

</li>

))}

</ul>

);

}

1. **Why you can't update props in React?**

In React, props (short for properties) are used to pass data from a parent component to a child component. Props are intended to be read-only, meaning that they should not be modified directly within the child component. This is because React follows a one-way data flow, where data flows from parent to child components.

1. **What is create-react-app? What are the several files and folders in the root directory?**

The create-react-app is an excellent tool for beginners, which allows you to create and run React project very quickly. It does not take any configuration manually. This tool is wrapping all of the required dependencies like web pack. Babel for react project itself and then you need to focus on writing code.

**the several files and folders in the root directory**

* **node modules:** It contains the React library and any other third party libraries needed.
* **public:** It holds the public assets of the application. It contains the index.html where React will mount the application by default on the <div id="root"></div> element.
* **src:** It contains the App.css, App.js, App.test.js, index.css, index.js, and serviceWorker.js files. Here, the App.js file always responsible for displaying the output screen in React.
* **package-lock.json:** It is generated automatically for any operations where npm package modifies either the node modules tree or package.json. It cannot be published. It will be ignored if it finds any other place rather than the top-level package.
* **package.json:** It holds various metadata required for the project. It gives information to npm, which allows to identify the project as well as handle the project?s dependencies.
* **README.md:** It provides the documentation to read about React topics.

1. **What are the Disadvantages of React JS?**

* **The high pace of development**

The high pace of development has an advantage and disadvantage both. In case of disadvantage, since the environment continually changes so fast, some of the developers not feeling comfortable to relearn the new ways of doing things regularly.

* **Poor Documentation**

It is another cons which are common for constantly updating technologies. React technologies updating and accelerating so fast that there is no time to make proper documentation.

* **View part**

ReactJS Covers only the UI Layers of the app and nothing else. So you still need to choose some other technologies to get a complete tooling set for development in the project.

* **JSX as a barrier**

ReactJS uses JSX. It's a syntax extension that allows HTML with JavaScript mixed together. This approach has its own benefits, but some members of the development community consider JSX as a barrier, especially for new developers. Developers complain about its complexity in the learning curve.

1. **How to combine multiple inline style objects?**

To combine multiple inline style objects in React, you can use the spread operator (...) to merge the objects together.

**Here's an example:**

import React from 'react';

function MyComponent() {

const style1 = { color: 'red', fontSize: '16px' };

const style2 = { fontWeight: 'bold', textTransform: 'uppercase' };

const combinedStyles = { ...style1, ...style2 };

return <div style={combinedStyles}>Combined Styles</div>;

}

1. **Is it possible to use React without rendering HTML?**

No, it is not possible to use React without rendering HTML. React is a JavaScript library that is specifically designed for building user interfaces (UIs) in web applications. It works by rendering components into the HTML DOM (Document Object Model) of a web page.

1. **What are the Requirements for React JS?**

To work with ReactJS, we need a few requirements:

* **Node.js and npm:** ReactJS relies on Node.js and npm (Node Package Manager) for development and managing dependencies. Make sure you have Node.js installed on your system, which includes npm.
* **Text Editor or IDE:** You'll need a text editor or an integrated development environment (IDE) to write your React code. There are many options available, such as Visual Studio Code, Sublime Text, Atom, or WebStorm.
* **Browser:** You'll need a modern web browser to run and test your React applications. Popular choices include Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge.

1. **Write the difference between class component and Functional component?**

|  |  |
| --- | --- |
| **Functional component** | **class component** |
| Functional Components is a plain JavaScript, you do not have a choice to set the state in functional components | Class components we have a feature to set the set state in component. |
| There is no render function we are using in functional components | In class components, we have a render function which is use to return the react elements |
| Functional components only accept the props as an argument | In class components we have both options use the props and set the state also |
| Functional components are sometimes called stateless components | Class components are sometimes called stateful components |

1. **How do you access props in attribute quotes?**

To access props in attribute quotes in JSX, you can use curly braces **{ }** to embed JavaScript expressions.

**Example:**

// Parent component

function App() {

const name = "John";

const age = 25;

return <ChildComponent name={name} age={age} />;

}

// Child component

function ChildComponent(props) {

return (

<div>

<p>Name: {props.name}</p>

<p>Age: {props.age}</p>

</div>

);

}

**Output:**

Name: John

Age: 25

1. **What are React Form? Explain about different ways of creating form in React JS with Example**

In React, forms are used to collect and handle user input. They provide a way for users to enter data and submit it to the application for processing. React provides several ways to create forms, including controlled components and uncontrolled components.

* **Controlled Components:**

Controlled components are React components that store and manage the form state in the component's state. They rely on event handlers to update the state as the user interacts with the form elements.

**Example:**

import React, { useState } from 'react';

function LoginForm() {

const [username, setUsername] = useState('');

const [password, setPassword] = useState('');

const handleUsernameChange = (event) => {

setUsername(event.target.value);

};

const handlePasswordChange = (event) => {

setPassword(event.target.value);

};

const handleSubmit = (event) => {

event.preventDefault();

console.log('Username:', username);

console.log('Password:', password);

// Perform form submission logic here

};

return (

<form onSubmit={handleSubmit}>

<label>

Username:

<input type="text" value={username} onChange={handleUsernameChange} />

</label>

<br />

<label>

Password:

<input type="password" value={password} onChange={handlePasswordChange} />

</label>

<br />

<button type="submit">Submit</button>

</form>

);

}

* **Uncontrolled Components:**

Uncontrolled components, on the other hand, do not rely on managing form state within React's component state. Instead, they allow the form elements to hold their own state and directly access their values when needed.

**Example:**

import React, { useRef } from 'react';

function UncontrolledForm() {

const usernameRef = useRef();

const passwordRef = useRef();

const handleSubmit = (event) => {

event.preventDefault();

const username = usernameRef.current.value;

const password = passwordRef.current.value;

console.log('Username:', username);

console.log('Password:', password);

// Perform form submission logic here

};

return (

<form onSubmit={handleSubmit}>

<label>

Username:

<input type="text" ref={usernameRef} />

</label>

<br />

<label>

Password:

<input type="password" ref={passwordRef} />

</label>

<br />

<button type="submit">Submit</button>

</form>

);

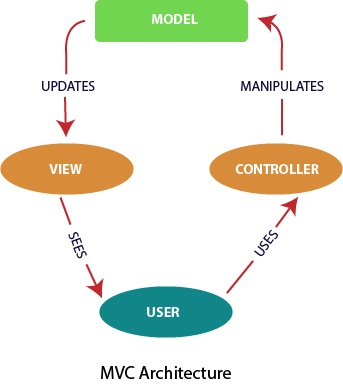
}

1. **Explain about MVC model**

MVC stands for Model View Controller. It is an architectural pattern used for developing the user interface.

**MVC Architecture:**

* **Model:** It is responsible for maintaining the behavior and data of an application.
* **View:**  It is used to display the model in the user interface.
* **Controller:** It acts as an interface between the Model and the View components. It takes user input, manipulates the data(model) and causes the view to update.



1. **What are React Components? Explain the different types of React components with Example.**

In React, components are the building blocks of the user interface. They encapsulate reusable pieces of code that define the structure and behavior of specific parts of the UI.

* **Functional Components:**

Functional components are JavaScript functions that accept props as arguments and return JSX elements.

**Example:**

import React from 'react';

function Greeting(props) {

return <h1>Hello, {props.name}!</h1>;

}

export default Greeting;

* **Class Components:**

Class components are JavaScript classes that extend the React.Component class. They have more features and capabilities than functional components, such as lifecycle methods and the ability to hold and manage their own state.

**Example:**

import React, { Component } from 'react';

class Counter extends Component {

constructor(props) {

super(props);

this.state = {

count: 0,

};

}

incrementCount() {

this.setState((prevState) => ({ count: prevState.count + 1 }));

}

render() {

return (

<div>

<p>Count: {this.state.count}</p>

<button onClick={() => this.incrementCount()}>Increment</button>

</div>

);

}

}

export default Counter;

1. **What is the difference between HTML and React event handling?**

|  |  |
| --- | --- |
| **HTML event handling** | **React event handling** |
| we specify event using “onclick”,”onsubmit”which is the normal convention. | We specify the event using “onClick”,”onSubmit” likewise which is camel case convention. |
| We bind or provide the listener in form of the string. | We bind or provide the listener in form of function name or method name as a variable. |
| The string we pass to the listener should have “( )” at the end in order to work. | We are only suppose to pass the method name and nothing else. React can determine that it has to run this method. |
| We can add event listener any time using external javascript. | We have to specify all the Events at the time of creating the component. |

1. **What are React JSX? Why to use JSX? How to nest the elements in JSX, Explain with Example?**

**React JSX**

JSX provides you to write HTML/XML-like structures (e.g., DOM-like tree structures) in the same file where you write JavaScript code, then pre-processor will transform these expressions into actual JavaScript code. Just like XML/HTML, JSX tags have a tag name, attributes, and children.

**Why use JSX?**

• It is faster than regular JavaScript because it performs optimization while translating the code to JavaScript

• Instead of separating technologies by putting markup and logic in separate files, React uses components that contain both.

• It is type-safe, and most of the errors can be found at compilation time.

• It makes easier to create templates.

**Nested Elements in JSX:**

import React from 'react';

function MyComponent() {

return (

<div>

<h1>Welcome to MyComponent</h1>

<p>This is a paragraph inside the div.</p>

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

</div>

);

}

export default MyComponent;

1. **What are Higher-Order components?**

Higher-Order Components (HOCs) are a design pattern in React that allow for code reuse and component composition. A Higher-Order Component is a function that takes a component and returns a new enhanced component.

1. **What is children prop?**

In React, the children prop is a special prop that is used to pass components or elements as the content between the opening and closing tags of a component. It allows you to create components that can wrap and render their child components or elements.

**Example:**

import React from 'react';

const MyComponent = (props) => {

return (

<div>

<h1>{props.title}</h1>

<p>{props.description}</p>

<div>{props.children}</div>

</div>

);

};

const App = () => {

return (

<MyComponent title="Hello" description="This is a component">

<p>This is the child content.</p>

</MyComponent>

);

};

export default App;

1. **What is the purpose of getDerivedStateFromProps() lifecycle method?**

The getDerivedStateFromProps() is a lifecycle method in React that is invoked before rendering and updating a component. Its purpose is to allow a component to update its internal state based on changes in props.

**Example:**

import React from 'react';

class MyComponent extends React.Component {

static getDerivedStateFromProps(props, state) {

if (props.value !== state.value) {

return { value: props.value };

}

return null;

}

constructor(props) {

super(props);

this.state = {

value: props.value,

};

}

render() {

return <div>{this.state.value}</div>;

}

}

export default MyComponent;

1. **What is the purpose of getSnapshotBeforeUpdate() lifecycle method?**

The getSnapshotBeforeUpdate() is a lifecycle method in React that is invoked immediately before a component's updates are applied to the DOM. Its purpose is to capture information from the DOM before it potentially changes due to the component's update.

**Example:**

import React from 'react';

class MyComponent extends React.Component {

constructor(props) {

super(props);

this.state = {

scrollPosition: 0,

};

this.myRef = React.createRef();

}

componentDidMount() {

this.myRef.current.addEventListener('scroll', this.handleScroll);

}

componentWillUnmount() {

this.myRef.current.removeEventListener('scroll', this.handleScroll);

}

handleScroll = () => {

this.setState({ scrollPosition: this.myRef.current.scrollTop });

};

getSnapshotBeforeUpdate(prevProps, prevState) {

// Capture the scroll position of the component

return this.myRef.current.scrollTop;

}

componentDidUpdate(prevProps, prevState, snapshot) {

// Compare the previous and current scroll positions

if (snapshot !== null && snapshot < this.myRef.current.scrollTop) {

// Scrolled down

// Perform some actions based on the scroll behavior

} else if (snapshot !== null && snapshot > this.myRef.current.scrollTop) {

// Scrolled up

// Perform some actions based on the scroll behavior

}

}

render() {

return (

<div ref={this.myRef} style={{ height: '200px', overflow: 'scroll' }}>

{/\* Component content \*/}

</div>

);

}

}

export default MyComponent;

1. **How to conditionally apply class attributes?**

To conditionally apply class attributes in React, you can use the JavaScript ternary operator or logical operators within the JSX code.

**Example:**

import React from 'react';

const MyComponent = (props) => {

const isSpecial = props.isSpecial;

return (

<div className={`my-component ${isSpecial ? 'special' : ''}`}>

{props.content}

</div>

);

};

export default MyComponent;

1. **What is react CSS? What are the different ways to styling React components elements?**

CSS in React is used to style the React App or Component. The style attribute is the most used attribute for styling in React applications, which adds dynamically-computed styles at render time. It accepts a JavaScript object in camelCased properties rather than a CSS string.

* **Inline Styling:**

The inline styles are specified with a JavaScript object in camelCase version of the style name. Its value is the style’s value, which we usually take in a string.

**Example:**

const MyComponent = () => {

const style = {

backgroundColor: 'red',

fontSize: '20px',

};

return <div style={style}>Hello, React!</div>;

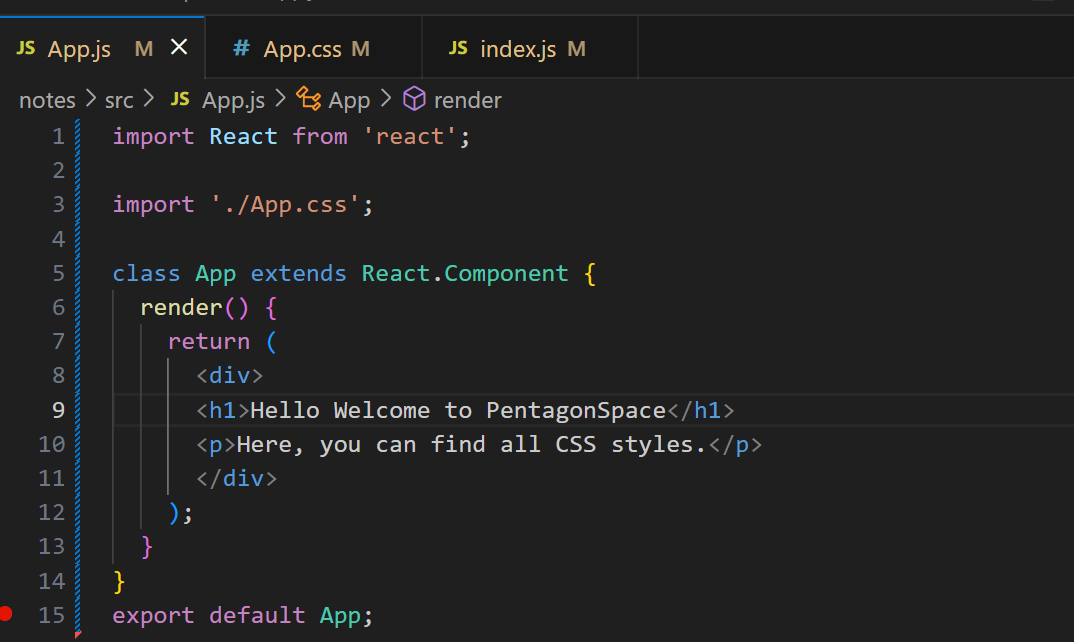
};

* **CSS Stylesheet:**

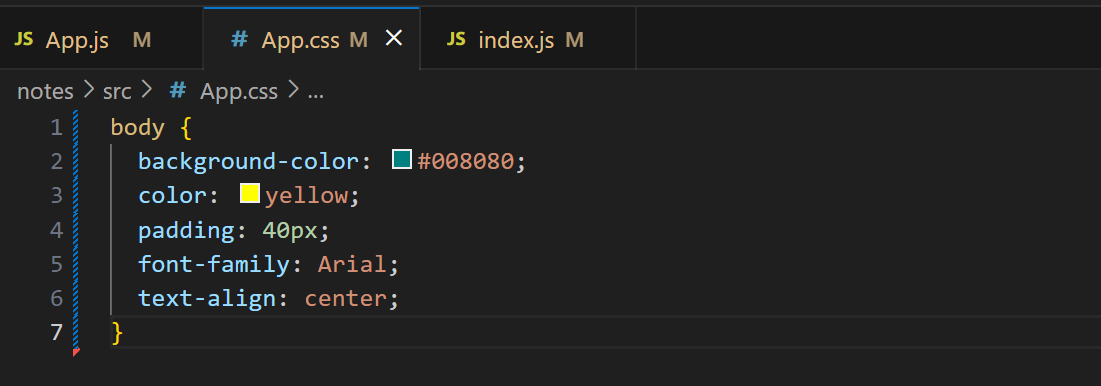
You can write styling in a separate file for your React application, and save the file with a .css extension. Now, you can import this file in your application.

**Example:**

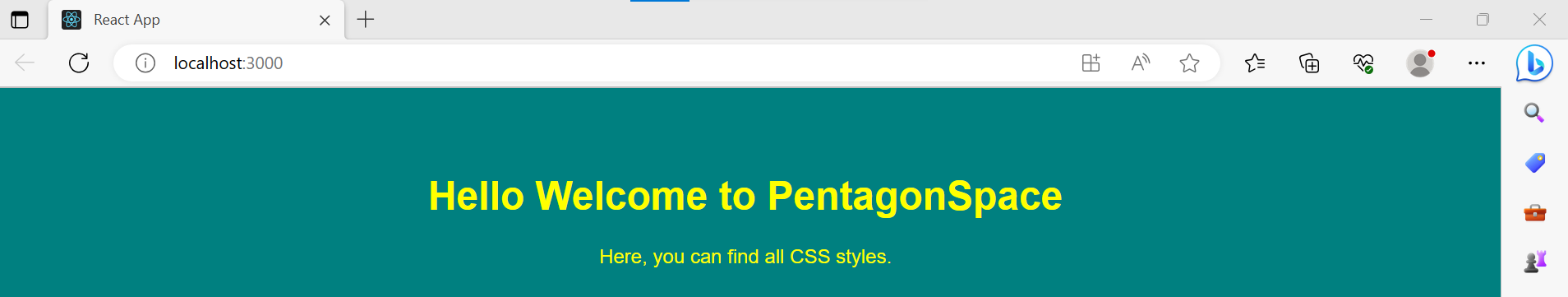
App.js



App.css



Output:

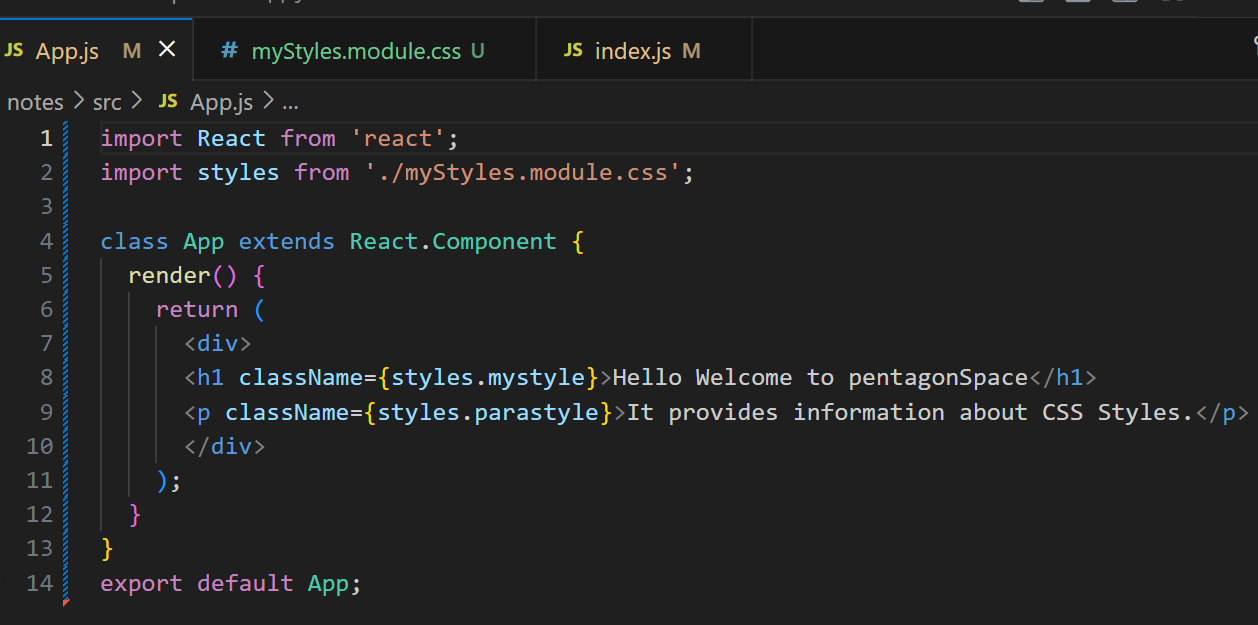
****

* **CSS module:**

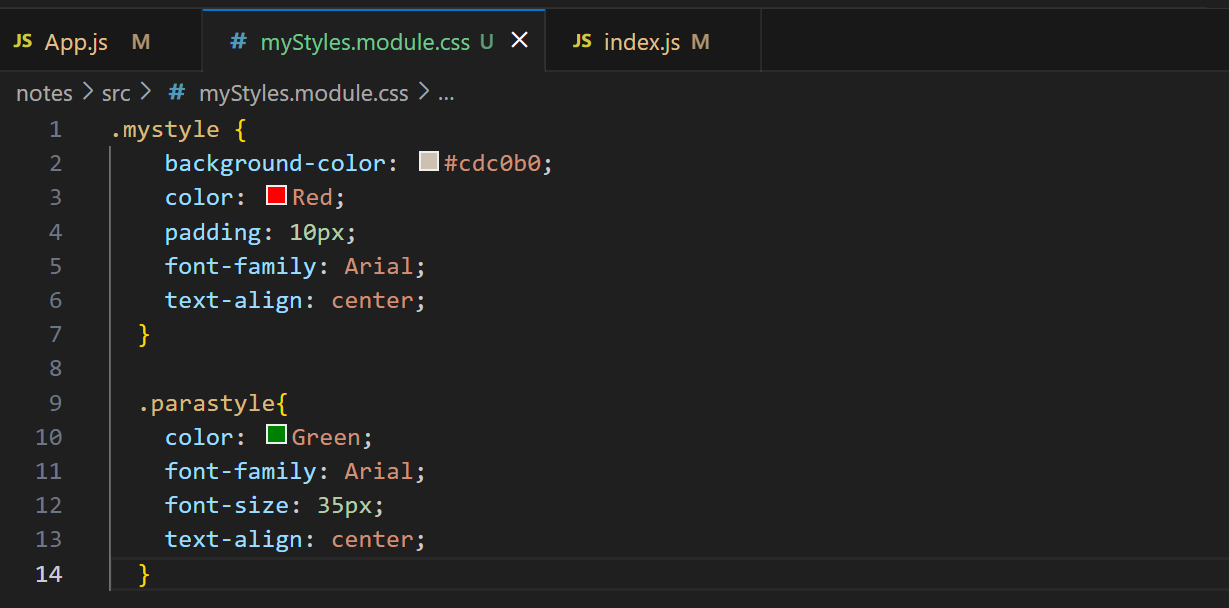
CSS Module is another way of adding styles to your application. It is a CSS file where all class names and animation names are scoped locally by default. It is available only for the component which imports it, means any styling you add can never be applied to other components without your permission, and you never need to worry about name conflicts.

**Example:**

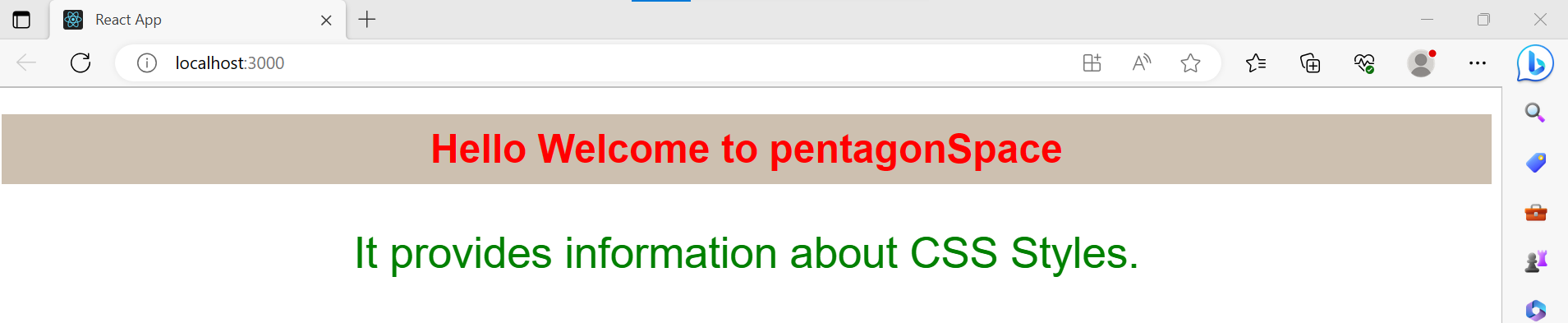
App.js



myStyles.module.css



Output:



1. **What is React Map? Explain about Traversing the list elements with example.**

In React, the map() method is used to traverse and render lists of elements or data. It allows you to iterate over an array and return a new array with transformed or modified elements.

**Example:**

import React from 'react';

const MyComponent = () => {

const names = ['Alice', 'Bob', 'Charlie'];

const nameList = names.map((name, index) => (

<li key={index}>{name}</li>

));

return <ul>{nameList}</ul>;

};

export default MyComponent;

1. **What are React list? Explain about different ways of creating list in React JS with Example.**

Lists are used to display data in an ordered format and mainly used to display menus on websites. The map() function is used for traversing the lists. Include this new list inside <ul> </ul> elements and render it to the DOM.

* **Using map() method:**

The map() method is commonly used to iterate over an array and generate a list of elements. Each element in the array is transformed into a React component or JSX element and returned as part of a new array.

**Example:**

import React from 'react';

const MyComponent = () => {

const items = ['Item 1', 'Item 2', 'Item 3'];

const itemList = items.map((item, index) => (

<li key={index}>{item}</li>

));

return <ul>{itemList}</ul>;

};

export default MyComponent;

* **Using a for loop:**

Although it is less common in React, you can also use a traditional for loop to iterate over an array and generate a list of elements.

**Example:**

import React from 'react';

const MyComponent = () => {

const items = ['Item 1', 'Item 2', 'Item 3'];

const itemList = [];

for (let i = 0; i < items.length; i++) {

itemList.push(<li key={i}>{items[i]}</li>);

}

return <ul>{itemList}</ul>;

};

export default MyComponent;

1. **What are React table? Explain about different ways of creating table in React JS with Ex.**

A table is an arrangement which organizes information into rows and columns. It is used to store and display data in a structured format.

The react-table is a lightweight, fast, fully customizable (JSX, templates, state, styles, callbacks), and extendable Datagrid built for React. It is fully controllable via optional props and callbacks.

* **Using plain HTML:**

we can create a table using plain HTML tags in JSX. This approach is suitable for simple tables with static data.

**Example:**

import React from 'react';

const MyComponent = () => {

return (

<table>

<thead>

<tr>

<th>Name</th>

<th>Age</th>

</tr>

</thead>

<tbody>

<tr>

<td>John</td>

<td>25</td>

</tr>

<tr>

<td>Jane</td>

<td>30</td>

</tr>

</tbody>

</table>

);

};

export default MyComponent;

* **Using Array Mapping:**

If you have dynamic data, you can use array mapping to generate table rows dynamically based on the data array.

**Example:**

import React from 'react';

const MyComponent = () => {

const data = [

{ name: 'John', age: 25 },

{ name: 'Jane', age: 30 },

];

const tableRows = data.map((item, index) => (

<tr key={index}>

<td>{item.name}</td>

<td>{item.age}</td>

</tr>

));

return (

<table>

<thead>

<tr>

<th>Name</th>

<th>Age</th>

</tr>

</thead>

<tbody>

{tableRows}

</tbody>

</table>

);

};

export default MyComponent;