# MODULE 6: React JS


## Q.1) List and explain different features of react

⇒

React is a popular JavaScript library for building user interfaces, especially for single-page applications. Here are some of its key features:

1. **JSX (JavaScript XML)**:
    - JSX is a syntax extension that allows you to write HTML elements in JavaScript code. It makes it easier to create React components and visualize the UI structure.
    - Example:

```
const element = <h1>Hello, World!</h1>;
```

2. **Components**:
   - React is component-based, meaning the UI is divided into reusable and self-contained pieces called components. Each component can manage its own state and props.
   - Example:

```
function Greeting() {
  return <h1>Hello, World!</h1>;
}
```

3. **Virtual DOM**:
   - React uses a virtual representation of the DOM (Document Object Model). This virtual DOM allows React to update only the parts of the UI that need re-rendering, resulting in improved performance.
   - React compares the virtual DOM with the real DOM and efficiently updates the real DOM based on changes.
4. **One-way Data Binding**:
   - React follows a unidirectional data flow, meaning that data flows from parent components to child components through props. This makes it easier to understand how data changes in the application.
   - Example:

```
function App() {
  const message = "Hello, World!";
  return <Greeting message={message} />;
}
```

5. **Lifecycle Methods**:
    - React components have lifecycle methods that allow developers to hook into different phases of a component's existence (e.g., mounting, updating, unmounting). This is useful for managing side effects, fetching data, etc.
    - Common lifecycle methods include `componentDidMount`, `componentDidUpdate`, and `componentWillUnmount`.
6. **Hooks**:
    - React introduced Hooks (e.g., `useState`, `useEffect`) to allow functional components to manage state and side effects without needing class components. Hooks promote cleaner and more concise code.
    - Example:

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  );
}
```

7. **React Router**:
    - React Router is a powerful routing library for React applications that enables the navigation between different views of the application. It allows developers to create single-page applications with multiple views.
    - Example:

```
import { BrowserRouter as Router, Route, Switch } from 'react-route
r-dom';

function App() {
 return (
  <Router>
   <Switch>
    <Route path="/" exact component={Home} />
    <Route path="/about" component={About} />
   </Switch>
  </Router>
 );
}
```

## Q.2) what are the features of react?

⇒

The features of React can be summarized as follows:

- **Declarative**: React makes it easy to design interactive UIs by using declarative views, which makes your code more predictable and easier to debug.
- **Reusable Components**: You can create reusable UI components that can be easily maintained and reused throughout your application.
- **Performance Optimization**: With the virtual DOM and efficient rendering, React ensures better performance and faster UI updates.
- **Rich Ecosystem**: React has a large ecosystem, including tools and libraries such as Redux for state management, React Router for navigation, and more.
- **Community Support**: Being one of the most popular JavaScript libraries, React has a vast community and plenty of resources, tutorials, and libraries available.

## Q.3) state some features of react JS.

⇒ (same as above 2 questions)

## Q.4) discuss the advantages of react JS.

⇒

- **Performance**: React's virtual DOM improves application performance by minimizing direct manipulation of the actual DOM, resulting in faster rendering and updates.
- **Reusability**: Components are reusable, making it easier to maintain and scale applications. Changes made to a component are reflected across all instances.
- **Strong Community**: A large community and ecosystem offer extensive resources, libraries, and tools, facilitating development and support.
- **Declarative Syntax**: The declarative nature of React makes code more predictable and easier to debug. Developers can focus on what the UI should look like rather than how to achieve it.
- **SEO-Friendly**: React can be rendered on the server side, improving search engine optimization (SEO) for applications, as search engines can crawl the rendered HTML.
- **Flexibility**: React can be used in combination with other libraries and frameworks, allowing developers to choose the best tools for their projects.
- **Mobile Development**: React Native extends React's capabilities to mobile app development, allowing developers to create native applications using the same principles.

## Q.5) What are the features of react JS and write a code for printing "hello world" using react JS? —— [10 MARKS rest above are 5 marks]

⇒

The features of React JS include:

- **JSX Syntax**: A syntactic sugar that allows mixing HTML with JavaScript.
- **Component-Based Architecture**: Encourages the creation of reusable components.
    - **Lifecycle Methods**: Provides hooks to control component behavior at different stages.
- **State Management**: Allows managing the state of components effectively.
- **Rich Ecosystem and Libraries**: Offers a variety of tools and libraries for development.
- **Developer Experience**: With features like hot reloading and developer tools, React provides a better development experience, enabling faster feedback and debugging.

## Code Example: Printing "Hello, World!" using React JS

Here's a simple example of a React application that prints "Hello, World!":

`src/index.js`

This file serves as the entry point for your application.

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App"; // Import the App component

ReactDOM.render(
  <React.StrictMode>
    <App /> {/* Render the App component */}
  </React.StrictMode>,
  document.getElementById("root") // Mount the App to the root div
);
```

**2.** `src/App.js`

This is where you define the main application component that will display "Hello, World!".

```jsx
import React from "react";

function App() {
  return (
    <div style={{ textAlign: "center", marginTop: "50px" }}>
      <h1>Hello, World!</h1> {/* Display "Hello, World!" */}
    </div>
  );
}

export default App; // Export the App component
```

Q.) write a code in react JS to display ”hello world”

⇒(same code as above )

## Q.6) what is JSX? write JSX attributes with example?

⇒

### What is JSX?

JSX, or JavaScript XML, is a syntax extension for JavaScript commonly used with React to describe what the UI should look like.

It allows developers to write HTML-like code directly within JavaScript, making it easier to create and visualize the structure of UI components.

JSX is not required to write React applications, but it provides a more intuitive way to structure components and manage UI.

When you use JSX, you essentially define your UI using a declarative syntax that closely resembles HTML.

JSX is transformed into React elements, which are plain JavaScript objects.

## Key Features of JSX

1. **HTML-Like Syntax**: JSX looks similar to HTML, making it easy for developers familiar with HTML to pick up quickly.
2. **JavaScript Expressions**: You can embed JavaScript expressions inside curly braces `{}` within JSX, allowing for dynamic content rendering.
3. **Component Integration**: JSX can be used to create React components, allowing for a modular approach to building UIs.

## Example of JSX

Here's a simple example of how JSX is used in a React component:

```
import React from 'react';

function Greeting(props) {
  return (
    <div>
      <h1>Hello, {props.name}!</h1> {/* Using a JavaScript expression */}
    </div>
  );
}

export default Greeting;
```

In this example:

- The `Greeting` function is a React component that takes `props` as an argument.
- Inside the `return` statement, JSX is used to create a `div` that contains an `h1` element.
- The `{props.name}` is a JavaScript expression that dynamically inserts the value of `name` passed as a prop.

## Attributes in JSX

JSX supports several attributes that can be used similarly to HTML attributes, but with some important differences.

1. **ClassName**: Instead of using `class` as you would in HTML, JSX uses `className` to define CSS classes.

```
<div className="container">Hello World</div>
```

2. **Style**: Inline styles are defined using an object with camelCase properties, rather than a CSS string.

```
<div style={{ backgroundColor: 'blue', color: 'white' }}>Styled Text</di
v>
```

3. **HTML Attributes**: Most HTML attributes can be used in JSX, but some names differ. For example:

   - `tabindex` becomes `tabIndex`
   - `for` becomes `htmlFor`
   - `maxlength` becomes `maxLength`

   Example:

   1. **Event Handling**: Events are named using camelCase, and you pass a function as the event handler, rather than a string.

      Example:

      ```
      <button onClick={() => alert('Button clicked!')}>Click Me!</button>
      ```

```
<label htmlFor="username">Username</label>
<input type="text" id="username" />
```

4. **Conditional Rendering**: You can conditionally render elements using JavaScript expressions.

   Example:

```
const isLoggedIn = true;
return (
  <div>
    {isLoggedIn ? <h1>Welcome back!</h1> : <h1>Please sign in.</h1>}
  </div>
);
```

5. **Children**: JSX can also contain child elements, allowing you to nest components.

   Example:

```
function App() {
  return (
    <div>
      <Greeting name="Alice" />
      <Greeting name="Bob" />
    </div>
  );
}
```

### Summary

JSX is a powerful syntax extension for JavaScript that enhances the way developers create React components by providing a clear and concise way to describe the UI structure. Its attributes allow for seamless integration with CSS and event handling, making it a key feature of modern React development.

JSX allows developers to write more expressive and readable code while leveraging the full power of JavaScript within their UI components. By understanding JSX and its attributes, developers can create dynamic and interactive user interfaces in React efficiently.

## Q.7) What is JSX? explain its attributes with example

⇒ (same as above)