1. **JSX**

JSX or JavaScript XML is syntax extension for JavaScript which allows us to write HTML-like code in our JavaScript file.it simplifies the process of defining and structuring UI component in a more declarative manner.

**Syntax similarity to HTML**

JSX looks similar to HTML, making it more readable and familiar for developers who are already comfortable with HTML.

**Integration with JavaScript**

JSX is not a separate language but a syntactic extension of javascript.it get transpile into regular javascript by tools like Babel before it is executed in the browser.

**React.createElement**

JSX is essentially shorthand for the React.createElement function. The function takes three arguments the type of the element, any attribute/properties, and the content of the element.

JSX makes it more convenient to express these elements in a more HTML-like syntax.

**Embedding JavaScript Expressions**

JSX allows embedding JavaScript expressions within curly braces **`{}`**. This makes it easy to include dynamic value or execute JavaScript code within the markup.

**React component**

JSX is commonly used in React to define components. Component are reusable and self-contained piece of UI.

JSX use because

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 tim

It makes easier to create templates.

JSX Attribute

JSX Comments

JSX styling

1. **React Components**

The component is core building block of a react application. It make the task of complex UIs much easier. React components are independent and reusable code.

Component serve the same purpose as javascript functions, but work individually to return JSX as element for our UI.

Component are reusable, self-contained unit of code that encapsulate a specific piece of functionality, ui and behaviour.

**Functional Component**

The functional components are javascript functions that defines reusable piece of user interface. They do not natively have state or lifecycle methods, but they used react hooks for state and other feature of class components.

Functional component are functions that takes in props and return JSX.

Class Component

The class component in react is a javascript class that extends `React.Component`. It allows us to define a reusable, self-contained unit of ui with its own state and lifecycle methods

Class components were the primary way of creating components in React before the introduction of hooks.

Class components typically have a constructor where we initialize the component state.

State is managed internally within the component and can be updated using the setState() method

The render() method is required in class components. It returns JSX, describing the structure of the UI. It is called whenever the component needs to be re-rendered due to state or props changes.

Pure Component

The pure component are special type of class / functional component that do the shallow comparison of state and props. That means pure component only rerenders when its state / props changes.

In class regular React.Component and a React.PureComponent is only difference pure components are do the shallow comparision. Pure component take care of shouldComponentUpdate() by itself. If the previous state / props are the same as the next, the component is not retendered.

Pure component basically use for optimizing performance, because of stop unnecessary rerendering

React component are usually retendered

setState() is called

props values are updated

forceUpdate() is called

Higher-Order Component

Higher-order component is advanced technique in react for reusing component logic.it is not a react component in the API. It is a pattern that emerged from react compositional nature.

The higher-order component is a patter in React where a function takes a component as argument and returns a new component. This allows us to enhance the functionality of existing components by wrapping them with additional logic or behaviour.

HOC promote code reuse by encapsulating common logic or behaviour that can be applied to multiple components.

Common use cases

Authentication

HOC can add authentication logic to components, restricting access based on user authentication status.

Logging

HOC can log lifecycle events or actions performed by components.

Performance optimization

HOC can optimize performance by memorizing components or handling data fetching

1. **React State**

React state is a updatable structure/ object that is used to contain data of a component.it determines how a component renders and behaves over tim. State can change over tim in response to user actions, network responses, other events, and when state changes, react re-renders the component to reflect the updated state.

In class component, to define a state, we have to first declare a default set of value for defining the component initial state. To do this add a class constructor which assigns initial state using this.state. this.state property can be rendered inside render() method.

In functional component, we can use useState() hook to use state into the component. useState() returns a stateful value and a function to update it, allowing us to manage state in functional component.

Changing the State

In class component we can change class using this.setState() method by passing a new State object as the argument.

In functional component, useState hook return a function that is use to update the state of a component.

1. **React props**

Props stands for **“Properties”.** They are read-only. In react props are a way to pass data from parent component to child component. They allow us to customize behaviour and appearance of a component dynamically.

The props are immutable so we cannot modify the props from inside the component. When we need immutable data in the component, we have to add props to reactDom.render() method in the main.js file for react js.

Passing props

Props are passed from parent components to child components as attribute.in the parent component, we specify props when rendering the child component by providing attribute values.

Accessing Props

In the child component, we access props using the **“props”** object.

Props validation

PropTypes is a utility in React used to validate the types of props passed to a component.

By specifying PropTypes for a component, we can ensure that the props passed to it meet certain criteria, such as type and presence. PropTypes are defined as static properties of the component.

Props validation with PropTypes helps catch errors early in the development process and provides documentation for the expected props of a component.it promotes code reliability and maintainability in react application.

**Default props**

It is not necessary to always add props in the the reactDom.render() element. We can also set default props directly on the component constructor.

State and Props both are plain js object, both can contain default values and both are reacd-only when they are using by this.

`

|  |  |
| --- | --- |
| **Props** | **State** |
| Props are read only | State changes can be asynchronous |
| Props are immutable | State is mutable |
| Props allow us to pass data from one component to other component as a argument | State holds information about the components |
| Props can be accessed by the child component | State can not 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 component can not have state. |
| Props make component reusable | State can not 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. **React constructor**

Constructor is a special type of function called to create object. It is a method used to initialize objects state in a class. It automatically called during the creation of object in a class.

The concept of a constructor is the same in React. The constructor in React component is called before the components is mounted. When we implement constructor for a React component, we need to call super(props) method before any other statement. If we don’t call super(props), method this.state will be undefined in the constructor.

In react, constructor are mainly used for two purpose

It used for initializing the local state of the component by assigning a object to this.state.

It used for binding event handler methods that occur in our component.

If we define a function as arrow we not need to bind any event to **this**. Here the scope of this is global and not limited to any calling function.

1. **React component API**

Reactjs component is a top-level API. It makes the code completely individual and reusable in the application. Most important methods available in the React component API.

setState()

forceUpdate()

findDOMNode()

setState()

This method is used to update the state of the component. This method does not always replace the state immediately. Instead, it only adds changes to the original state. It is primary method that is used to update the user interface (UI) in response to event handlers and server responses.

1. **React component life cycle**

In ReactJS, every component creation process involves various lifecycle methods.the lifecycle of the component is divided into four phases.

**Mounting phase**

**Constructor**

**Static getDerivedStateFromProps()**

**render()**

**componentDidMount()**

**Updating phase**

**Static getDerivedStateFromProps()**

**shouldComponentUpdate()**

**render()**

**getSnapshotBeforeUpdate()**

**componentDidUpdate()**

**Unmounting phase**

**componentWillUnmount()**

**Error Handling**

**componentDidCatch()**

**getDerivedStateFromError()**

**componentWillReceiveProps() deprecated since react 16.3**

it was a lifecycle used in older version of react to handle changes to props before a component re-renders.it is deprecated due to its behaviour being problematic and leading to bugs and inconsistencies.

The main issue with **componentWillReceiveProps()** was that it would be called even when props haven’t actually changed, but only when the parent component re-renders.

To address these issues, react introduced **static getDerivedStateFromProps()** as a replacement in react 16.3. It allows components to update their internal state based on changes in props, without causing unnecessary re-renders.

Mounting Phases

Mounting phase occurs when instance of a component is being created and inserted into the DOM.During this, phase React goes through several steps to set up the component, initialize its state, and prepare it for rendering.

constructor()

The first step in the mounting phase is the constructor method called when the component is initialized. It is used for initializing state and binding event handlers.

static getDerivedStateFromProps()

static getDerivedStateFromProps() method is invoked right before rendering, both on the initial mount and on subsequent updates. It exists for rare cases when the state of the component depends on changes in props over time. Primary purpose of method is to derived state based on the initial props.