React JS

By:Khushbu Surati

Points....

- Overview
- React JSX
- React Components
- React Props
- React States
- React Events
- Event Handlers
- React Conditional Rendering
- React List

React Introduction:

- ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components.
- Open-source component-based front end library responsible only for the view layer of the application
- Created by **Jordan Walke,** who was a software engineer at **Facebook.**
- It was initially developed and maintained by Facebook and was later used in its products like **WhatsApp** & **Instagram.**
- Facebook developed ReactJS in **2011** in its newsfeed section, but it was released to the public in the month of **May 2013.**

React Introduction:

React Introduction:

- A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code.
- The components are the heart of all React applications.
- These Components can be nested with other components to allow complex applications to be built of simple building blocks.
- ReactJS uses virtual DOM based mechanism to fill data in HTML DOM.
- The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.
- Form component is made up of various input components.

Ways to install ReactJS

Using the npm command

Using the npx command

Install ReactJS (Using the create-react-app)

- Starting a new React project is very complicated
  - with so many build tools
  - Uses many dependencies, configuration files, and other requirements such as Babel, Webpack, ESLint before writing a single line of React code.
- Create React App CLI tool removes all that complexities and makes React app simple.
- Just install the package using NPM, and then run a few simple commands to get a new React project.
- The **create-react-app** is an excellent tool for beginners, which allows you to create and run React project very quickly.
- Does not take any configuration manually.
- This tool is wrapping all of the required dependencies like **Webpack**, **Babel** for React project itself and then you need to focus on writing React code only.

Install ReactJS (Using the create-react-app)

Minimum Requirement:

1. **Node**
2. **NPM**

- Run the following command to check the Node version in the command prompt.

  $ node -v

- Run the following command to check the NPM version in the command prompt.

  $ npm -v

By:Khushbu Surati

Install ReactJS (Using the create-react-app)

- Install React & Create a new React project

  C:\Users\ folder > **npx create-react-app reactproject**

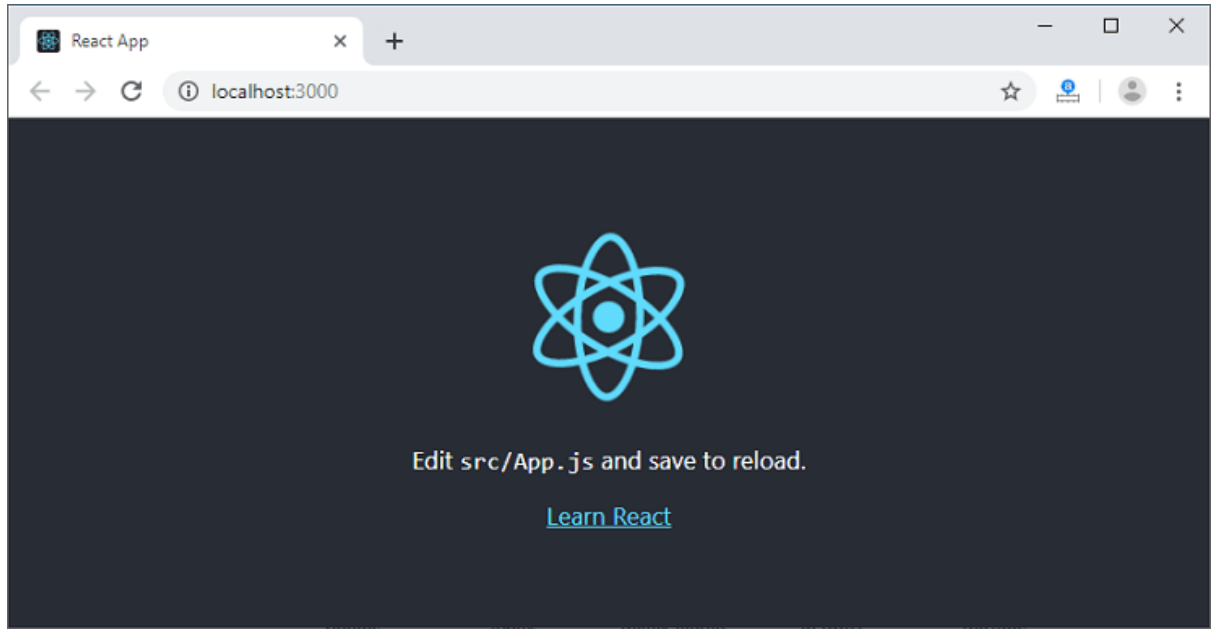
By:Khushbu Surati

Install ReactJS (Using the create-react-app)

- After the React project is created successfully on our system. Now, we need to start the server so that we can access the application on the browser. Type the following command in the terminal window.

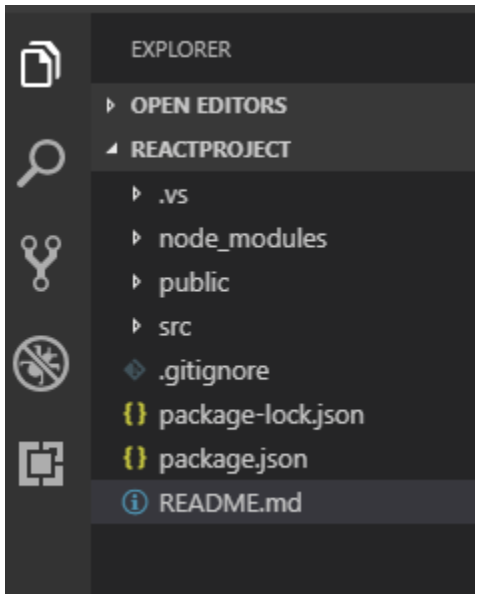
  npm start

- NPM is a package manager which starts the server and access the application at default server http://localhost:3000



By:Khushbu Surati

Install ReactJS (Using the create-react-app)



- **node_modules:** Contains the React library and any other third party libraries
- **public:** holds the public assets of the application. It contains the index.html where React will mount the application by default on the 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.

By:Khushbu Surati

React JSX

- JSX stands for JavaScript XML
- JSX allows us to write HTML in React
- JSX makes it easier to write and add HTML in React
- All of the React components have a **render** function. The render function specifies the HTML output of a React component.
- JSX(JavaScript Extension), is a React extension which allows writing JavaScript code that looks like HTML.
- JSX is an HTML-like syntax used by React that extends ECMAScript so that **HTML-like** syntax can co-exist with JavaScript/React code. The syntax is used by **pre-processors** (i.e., transpilers like babel) to transform HTML-like syntax into standard JavaScript objects that a JavaScript engine will parse.
- JSX provides you to write HTML/XML-like structures in the same file where you write JavaScript code, and then pre-processor will transform these expressions into actual JavaScript code.
- Just like XML/HTML, JSX tags have a tag name, attributes, and children.

By:Khushbu Surati

React JSX

- JSX:

  const myElement = <h1>I Love JSX!</h1>;

- Without JSX:

  const myElement = React.createElement('h1', {}, 'I do not use JSX!');

- With JSX you can write expressions inside curly braces { }.

  const myElement = <h1>React is {5 + 5} times better with JSX</h1>;

- To write HTML on multiple lines, put the HTML inside parentheses
- The HTML code must be wrapped in *ONE* top level element.
- You can use a "fragment" to wrap multiple lines. This will prevent unnecessarily adding extra nodes to the DOM.
- The class attribute is a much used attribute in HTML, but since JSX is rendered as JavaScript, and the class keyword is a reserved word in JavaScript, you are not allowed to use it in JSX.
- Use attribute className instead.

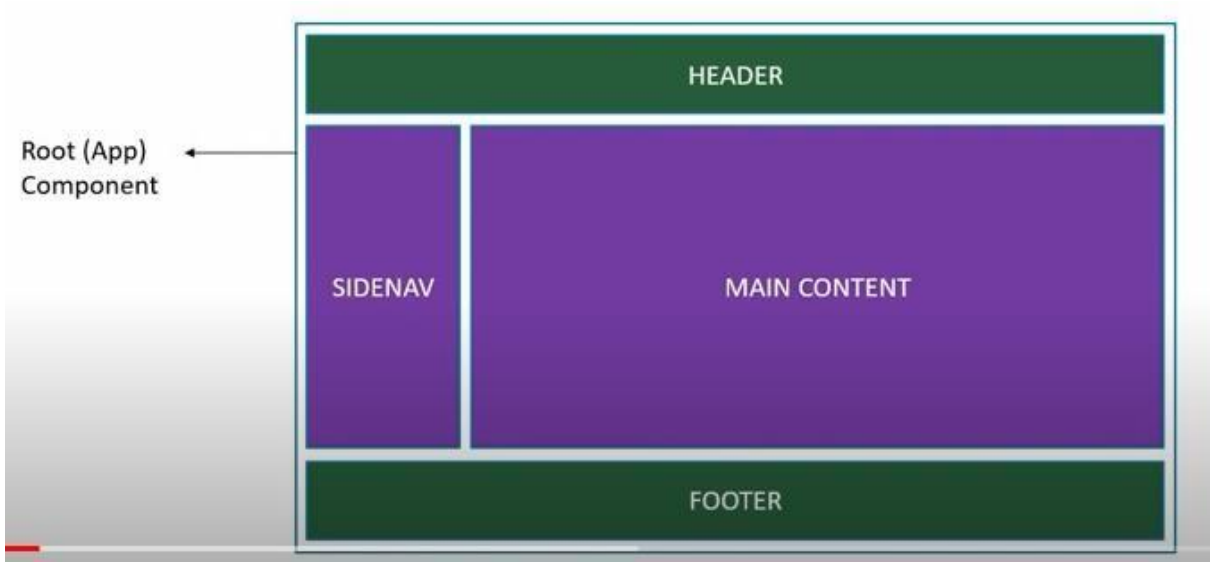
By:Khushbu Surati

React Components

- The component-based approach was introduced to build entire application into a small logical group of code, which is known as components.
- A Component is considered as the core building blocks of a React application.
- It makes the task of building UIs much easier.
- Each component exists in the same space, but they work independently from one another and merge all in a parent component, which will be the final UI of your application.
- Every React component has their own structure, methods as well as APIs. They can be reusable as per your need.
- For better understanding, consider the entire UI as a tree. Here, the root is the starting component, and each of the other pieces becomes branches, which are further divided into sub-branches.
- React is a component-based architecture. In react component represents a part of User Interface.

By:Khushbu Surati

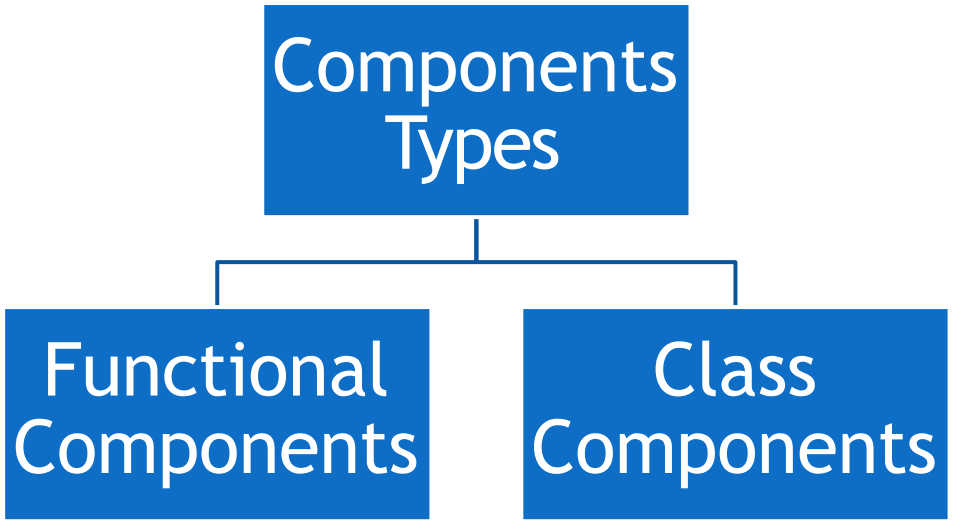
React Components



- Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML. Components can contain other nested components.
- A component code is usually placed inside JavScript file. For example App component is placed inside App.js file. It can have .jsx extension. Component is a code inside .js file.

By:Khushbu Surati

React Components



By:Khushbu Surati

Functional Components

- Function components are a way to write components that only contain a render method and don't have their own state.
- They are simply JavaScript functions that may or may not receive data as parameters.
- We can create a function that takes props(properties) as input and returns what should be rendered.
- The functional component is also known as a stateless component because they do not hold or manage state.

Greet.js

```
import React from 'react'

// function Greet(){
//     return <h1>Good Morning!!!</h1>
// }

const Greet = () => <h1>Goog Morning!!!!!</h1>

export default Greet
```

By:Khushbu Surati

Class Components

- Class components are more complex than functional components. It requires you to extend from React.
- Class component create a render function which returns a React element. You can pass data from one class to other class components.
- You can create a class by defining a class that extends Component and has a render function.
- The class component is also known as a stateful component because they can hold or manage local state.

- **Note: Once components are created add it into App.js file and run the application.**

By:Khushbu Surati

Class Components

Welcome.js

```
import React,{Component} from 'react'

class Welcome extends Component{
    render(){
        //Without using props
        return <h1>This is a class Component Welcome</h1>

        //Using props
        // return <h1>Welcome {this.props.name} a.k.a.
{this.props.nickName}</h1>

        //after destructuring props
        // const {name, nickName}=this.props
        // return <h1>Welcome {name} a.k.a. {nickName}</h1>
    }
}

export default Welcome
```

By:Khushbu Surati

React Props

- 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.
- It gives a way to pass data from one component to other components.
- It is similar to function arguments.
- Props are passed to the component in the same way as arguments passed in a function.

By:Khushbu Surati

React Props

- Props are **immutable** so we cannot modify the props from inside the component. Inside the components, we can add attributes called props. These attributes are available in the component as **this.props** and can be used to render dynamic data in our render method.
- Props is an optional input that a component may accepts. Its help us to make dynamic components.
- When you need immutable data in the component, you have to add props to the components in the **App.js** file of your ReactJS project and used it inside the component in which you need.
- We can pass Props to functional component as well as class components.

By:Khushbu Surati

Using Props with React components

Greeting.js

```
import React from 'react'

const Greeting= props =>{
    // console.log(props)
    return (
        <div>
            <h1>
                Hello {props.name} a.k.a. {props.nickName}
            </h1>
            {props.children}
        </div>

        )
}

export default Greeting
```

Functional Component

Welcome.js

```
import React,{Component} from 'react'

class Welcome extends Component{
    render(){
        //Without using props
        // return <h1>This is a class Component Welcome</h1>

        //Using props
        return <h1>Welcome {this.props.name} a.k.a. {this.props.nickName}</h1>

        //after destructuring props
        // const {name, nickName}=this.props
        // return <h1>Welcome {name} a.k.a. {nickName}</h1>
    }
}

export default Welcome
```

Class Component

By:Khushbu Surati

React State

- The state is an updatable structure that is used to contain data or information about the component.
- The state in a component can change over time.
- The change in state over time can happen as a response to user action or system event.
- A component with the state is known as stateful components.
- It is the heart of the react component which determines the behavior of the component and how it will render.
- They are also responsible for making a component dynamic and interactive.
- The state is an instance of React Component Class that can be defined as an object of a set of **observable** properties that control the behaviour of the component. In other words, the State of a component is an object that holds some information that may change over the lifetime of the component.

By:Khushbu Surati

React State (Conventions of Using State)

We must first have some initial state that we can define in the constructor of the component's class.

State should never be updated explicitly. Must use setState()

- this.state.attribute = "new-value"; ------ ✖
- this.setState({attribute: "new-value"});

Due to asynchronous processing, this.state.count may produce an undesirable result. A more appropriate approach is to define a function as a parameter

By:Khushbu Surati

React State

Message.js

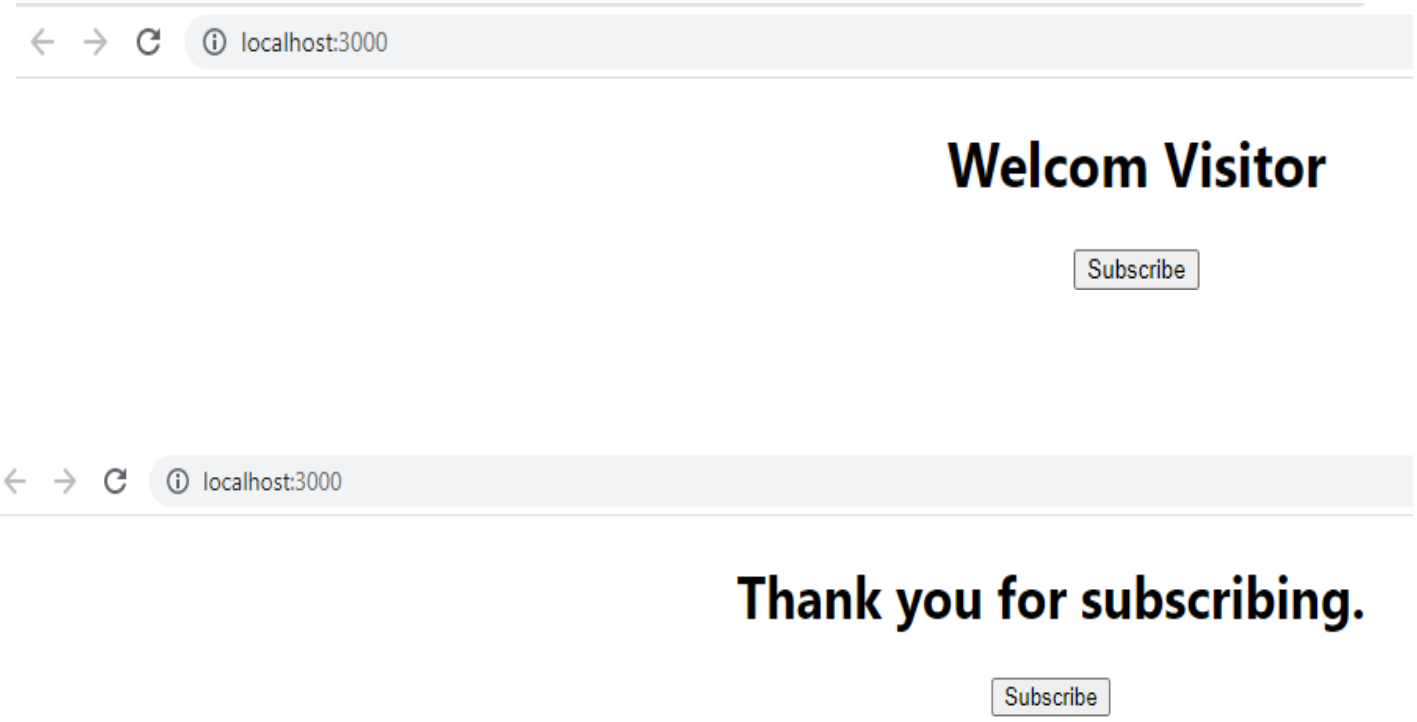
```
import React,{Component} from 'react'

class Message extends Component{

    constructor(){
        super()
        this.state={
            message:'Welcom Visitor'
        }
    }

    changeMessage(){
        this.setState({
            message:'Thank you for subscribing.'
        })
    }

    render(){
        //Using state
        return (
            <div>
                <h1>{this.state.message}</h1>
                <button onClick={()=> this.changeMessage()}>Subscribe</button>
            </div>
        )
    }
}
export default Message
```

App.js

```
import './App.css';
import Message from './components/Message';
function App() {
  return (
    <div className="App">
      <Message/>
    </div>
  );
}
```

By:Khushbu Surati

React State



By:Khushbu Surati

React State

Counter.js

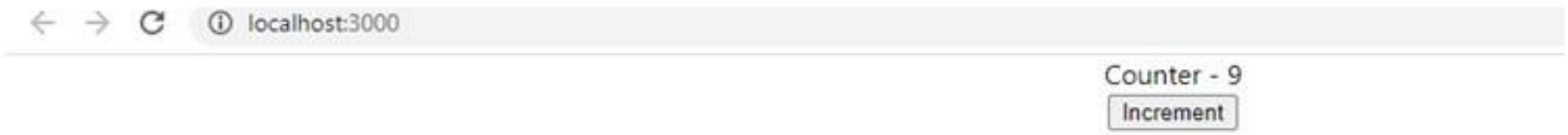
```
import React, { Component } from 'react'

class Counter extends Component {
    constructor(props) {
      super(props)

      this.state = {
         count:0
      }
    }

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

    render() {
        return (
        <div>
            <div>Counter - {this.state.count}</div>
            <button onClick={()=>this.increment()}>Increment</button>
        </div>
        )
    }
}

export default Counter
```

By:Khushbu Surati

React State

App.js

```
import './App.css';
import Counter from './components/Counter';
function App() {
  return (
    <div className="App">
      <Counter/>
    </div>
  );
}
```



By:Khushbu Surati

Props Vs. State

Props	State
Props are read-only.	State changes can be asynchronous.
Props are immutable.	State is mutable.
Props get passed to the component.	State is managed within the component.
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.	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.

By:Khushbu Surati

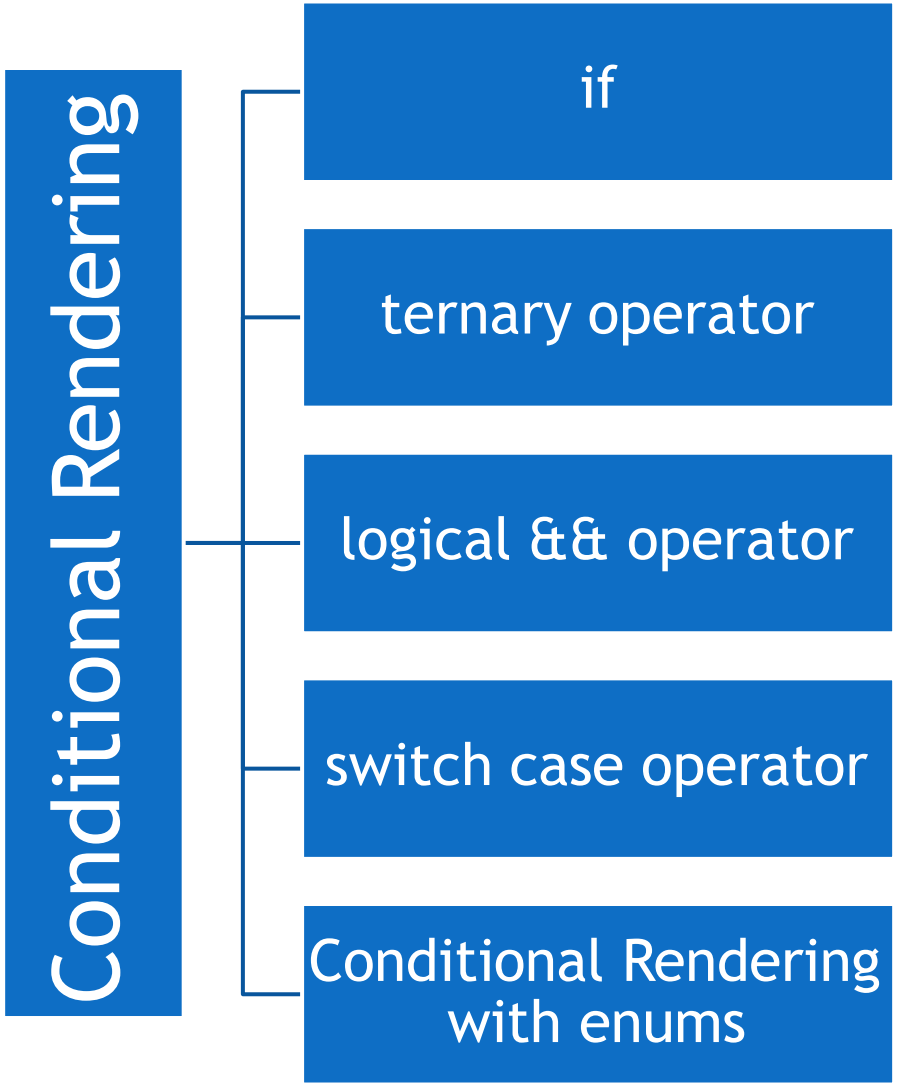
React Events

- An event is an action that could be triggered as a result of the user action or system generated event. For example, a mouse click, loading of a web page, pressing a key, window resizes, and other interactions are called events.
- React has its own event handling system which is very similar to handling events on DOM elements. The react event handling system is known as Synthetic Events. The synthetic event is a cross-browser wrapper of the browser's native event.
- We can add events in the application either using functional component or class components.
- Handling events with react have some syntactic differences from handling events on DOM.

1. React events are named as **camelCase** instead of **lowercase**.
2. With JSX, a function is passed as the **event handler** instead of a **string**.
3. We cannot return **false** to prevent the **default** behavior. We must call **preventDefault** event explicitly to prevent the default behavior.

By:Khushbu Surati

React Conditional Rendering



By:Khushbu Surati

React Conditional Rendering

if

- It is the easiest way to have a conditional rendering in React in the render method. It is restricted to the total block of the component. If the condition is **true**, it will return the element to be rendered.

Logical && operator

- Syntax

```
{
condition &&
// whatever written after && will be a part of output.
}
```

Ternary operator

- The ternary operator is used in cases where two blocks alternate given a certain condition. This operator makes your if-else statement more concise. It takes **three** operands and used as a shortcut for the if statement.
- Syntax

```
condition ? true : false
```

By:Khushbu Surati

React Conditional Rendering

- In React, we can create multiple components which encapsulate behavior that we need. After that, we can render them depending on some conditions or the state of our application.
- In other words, based on one or several conditions, a component decides which elements it will return.
- Conditional rendering works the same way as the conditions work in JavaScript. We use JavaScript operators to create elements representing the current state, and then React Component update the UI to match them.
- Consider an example of handling a **login/logout** button. The login and logout buttons will be separate components. If a user logged in, render the **logout component** to display the logout button. If a user not logged in, render the **login component** to display the login button.
- This situation is called as **conditional rendering**.

By:Khushbu Surati