React JS

ReactJS Concepts

Agenda

01 Virtual DOM

02 JSX

03 Props

04 State

05 Lifecycle Methods

06 Components

07 Class Components

08 Function Components

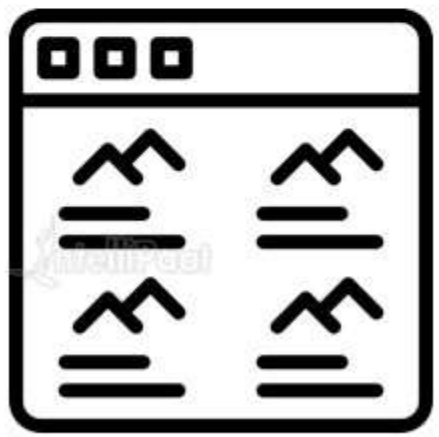
09 Class vs Functional Components

10 PropTypes

Virtual DOM

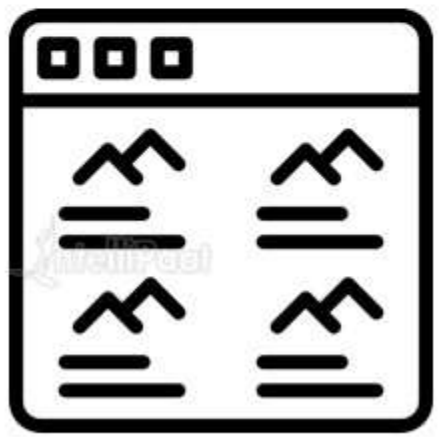
Virtual DOM

In ReactJS, virtual DOM is an ideal, or 'virtual,' representation of a UI that is kept in memory and synced with the 'real' DOM



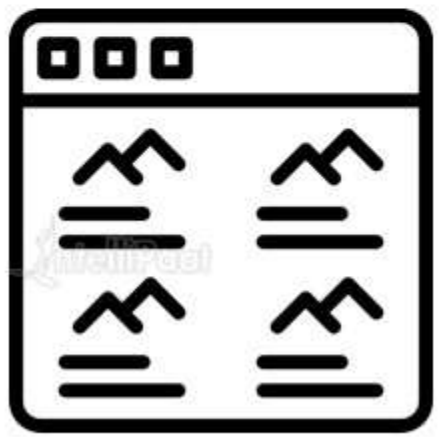
Virtual DOM

A virtual DOM is encapsulated in ReactDOM packages so that it can be used on the web



Virtual DOM

React uses a virtual DOM to improve the performance so that the real DOM is updated only when needed. This process of updating the real DOM from the virtual DOM is called reconciliation

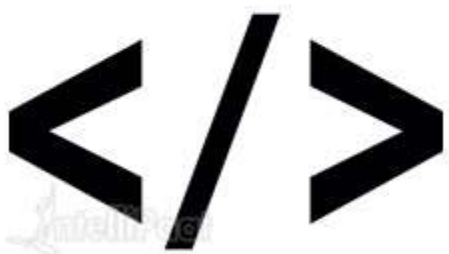


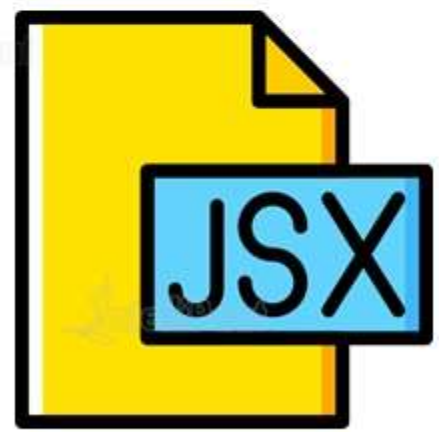
Hands-on: Creating a React Component

JSX

JSX

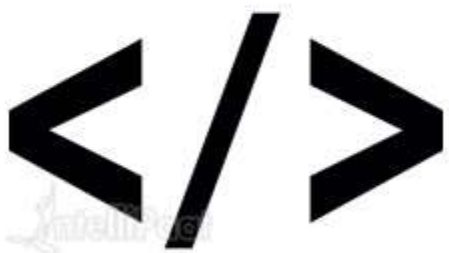
Under the hood, React uses the createElement method to create new DOM nodes to be attached to a node

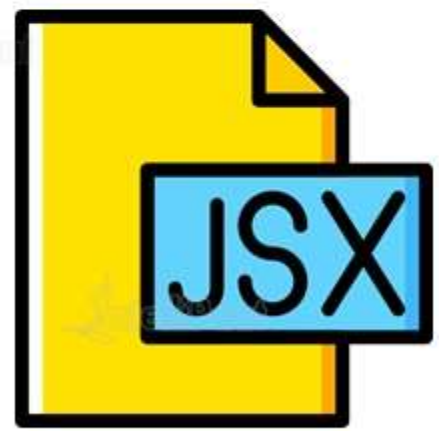




JSX

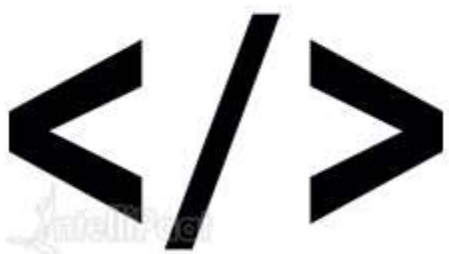
But, it is very difficult to use, especially when we have components nested into each other. This can make our code unreadable

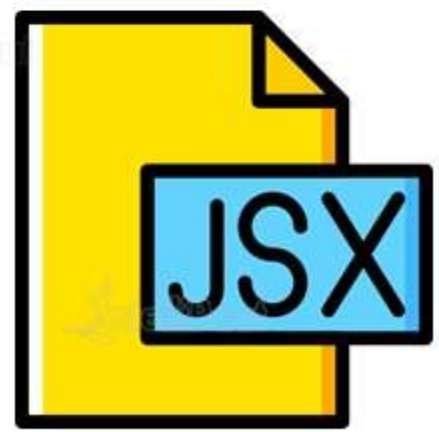




JSX

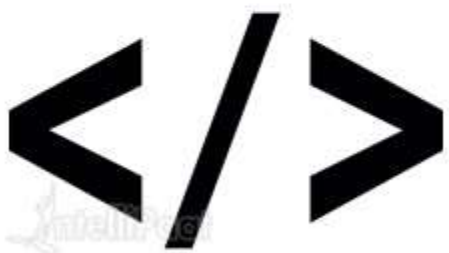
To solve this, in React, we use a special syntax called JSX, which is ultimately compiled to createElement calls, but it makes our code easier to read

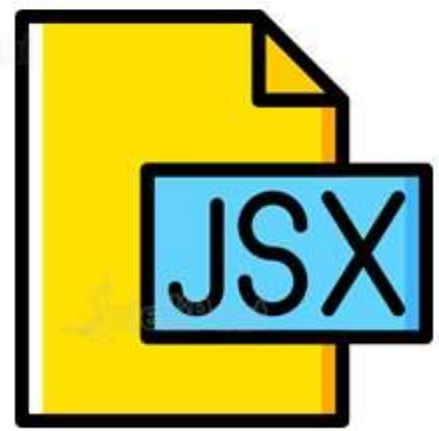




JSX

React does not mandate us to use JSX, but since it makes our code easier to read and understand, it is recommended that we do use it





Hands-on: Creating a React Component with JSX

Props

Props

Props stand for properties, and they are arguments passed to a component by a parent component for it to use

Props

Props are read-only and should not be modified by a component on its own. They can only be modified by the parent, passing those props

Props

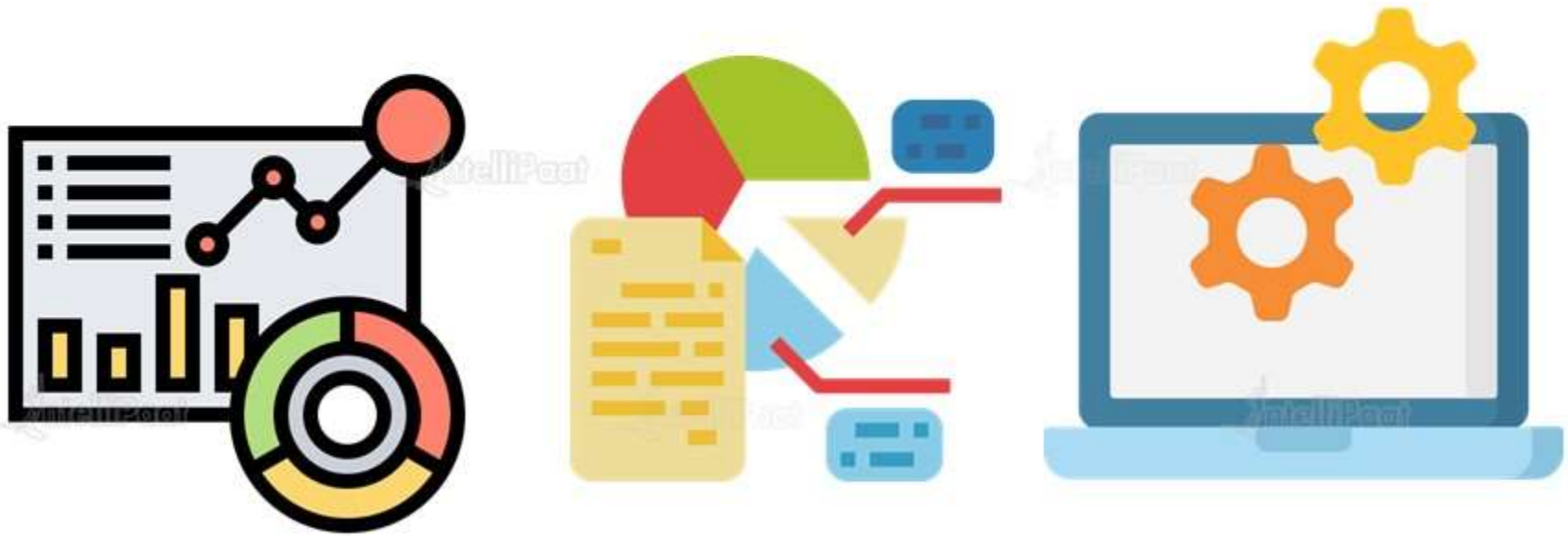
A change in props from the parent will lead to child components being updated and re-rendered to the DOM

Hands-on: Passing Props

State

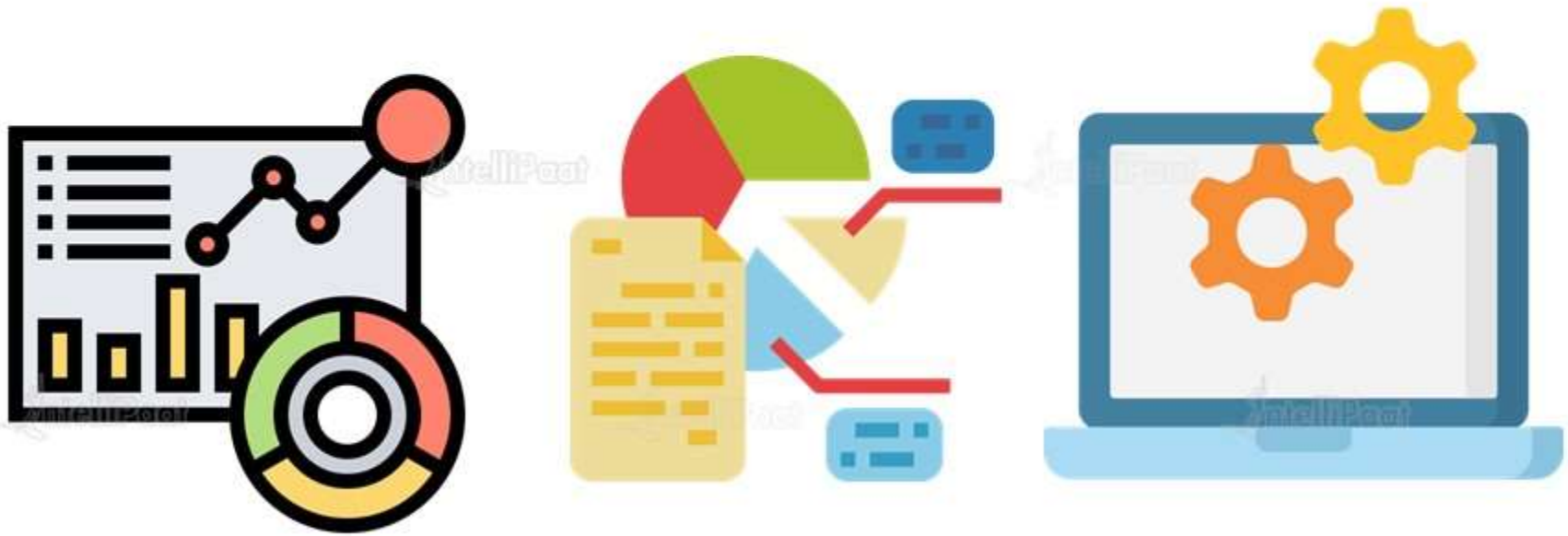
State

State is similar to props, but it is private and fully controlled by the components



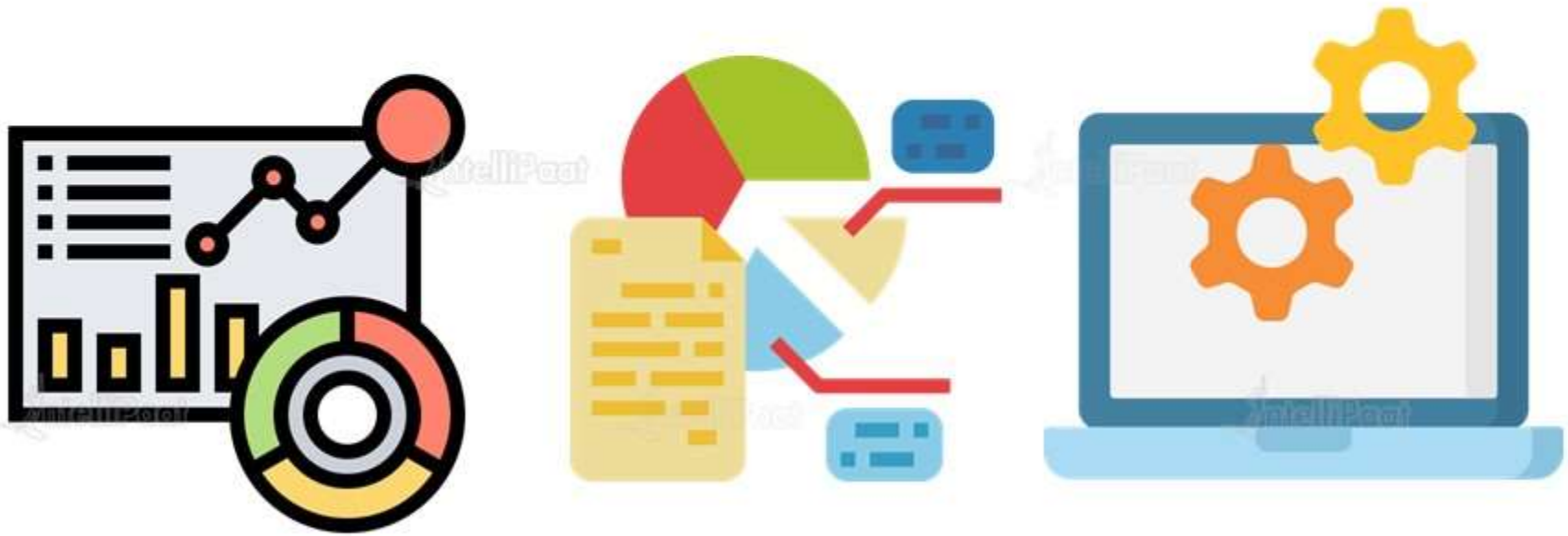
State

The state is used by a component to determine what to display on the screen and has to be managed by the component itself as other components do not have access to this state



State

In React, a component's state should not be changed directly. It should be done only via special setter methods because React keeps track of the state

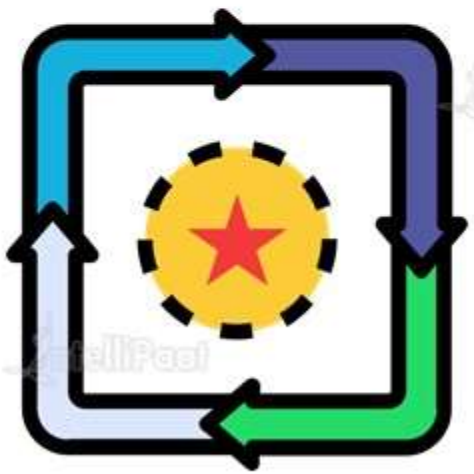


Hands-on: Using State

Lifecycle Methods

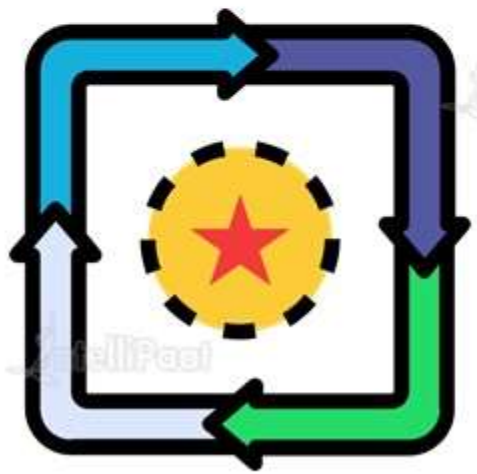
Lifecycle Methods

React manages the lifecycle of our components, i.e., tasks such as adding to and removing a component from the DOM



Lifecycle Methods

Sometimes, there will be tasks we wish to perform during these lifecycle tasks. They include adding or removing event handlers, fetching data, etc.



Lifecycle Methods

To allow developers execute code during these lifecycles, React provides them with lifecycle methods, which they can override

componentDidMount

componentDidUpdate

componentWillUnmount

This lifecycle method allows developers to execute code after a component is added to the DOM. It is ideal for adding the fetched data to the DOM nodes or adding event listeners to the DOM

Lifecycle Methods

To allow developers execute code during these lifecycles, React provides them with lifecycle methods, which they can override

componentDidMount

componentDidUpdate

componentWillUnmount

This lifecycle method allows developers to execute code after a component gets updated, which could be due to the updating of props and/or state

Lifecycle Methods

To allow developers execute code during these lifecycles, React provides them with lifecycle methods, which they can override

componentDidMount

componentDidUpdate

componentWillUnmount

This lifecycle method allows developers to execute code before a component is removed from the DOM. It is ideal for removing event listeners from the component to stop memory leak

Hands-on: Lifecycle Methods

Components

Components

In React, a UI is built using individual, independent pieces called components

Components

The user interface is built as a tree of components that are managed and converted into native DOM elements by React

Components

Components allow us to build a single, reusable, and isolated piece of the UI, which can be used to efficiently render the user interface

Components

In React, there are two types of components, which can be created

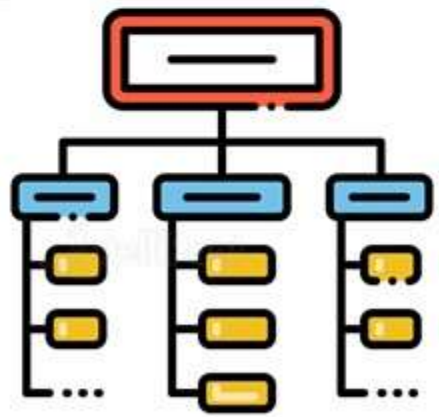
Class Components

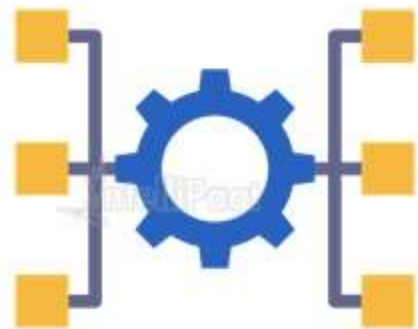
Function Components

Class Components

Class Components

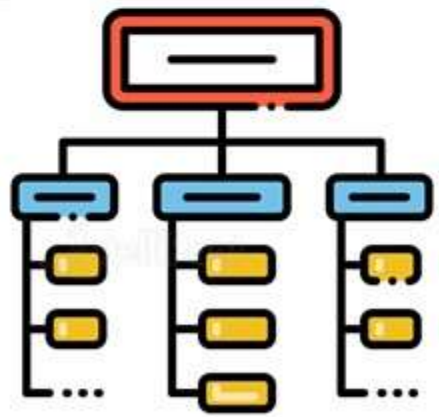
Class components are created by creating a class the extends the React.Component class. The class name will be the name of the component

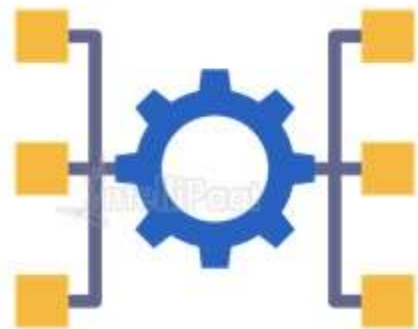




Class Components

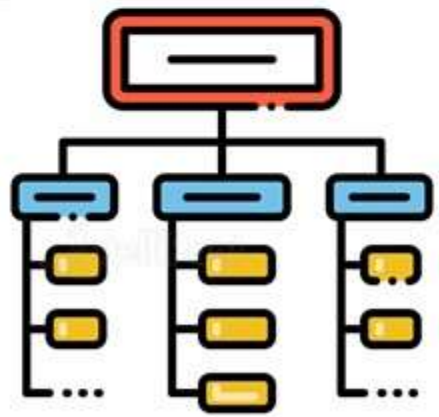
A class component needs to override the render method using JSX to describe the markup of the component

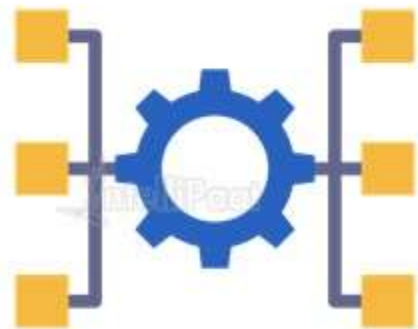




Class Components

Class components are still supported, but in React, it is preferable to use function components to reduce boilerplate code



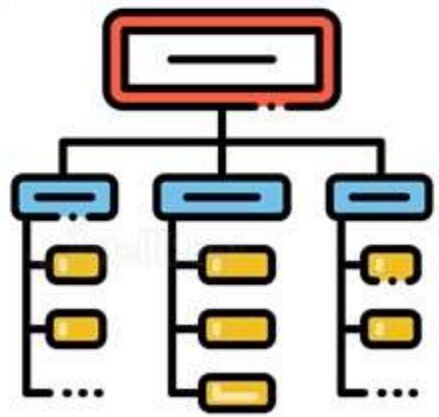


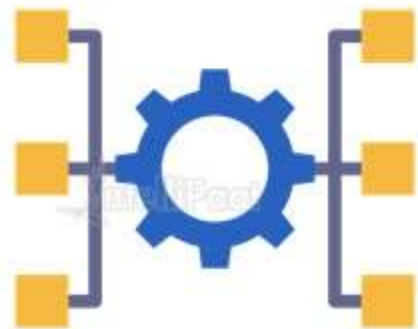
Hands-on: Creating Class Components

Functional Components

Functional Components

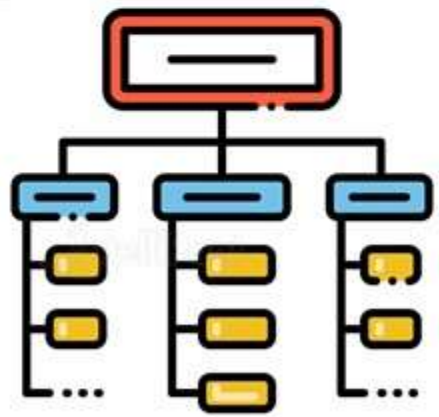
In a functional component, we create a function, instead of a class, that returns JSX to describe our component markup

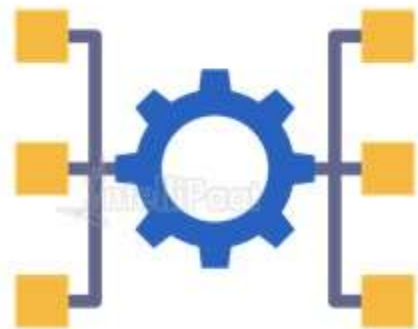




Functional Components

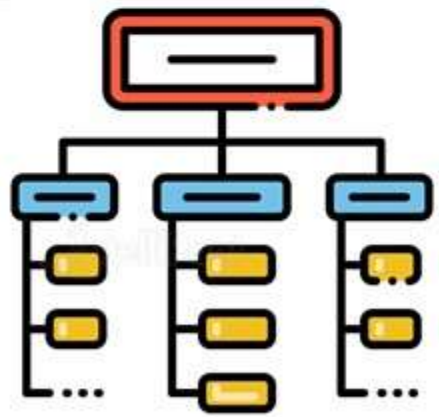
Functional components can take props as parameters and have the state to keep track of the data to be displayed

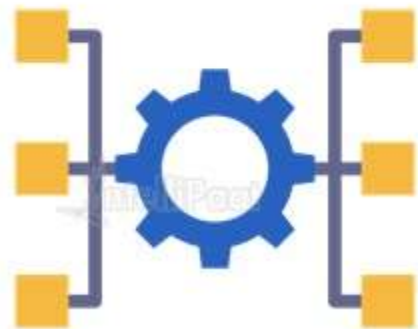




Functional Components

In React, functional components are given preference as they are less verbose and more maintainable





Hands-on: Creating Function Components

Class vs Function Components

Class vs Function Components

Backwards Compatibility	State and Props	Lifecycle Methods	Verbosity

Class Component	Function Component
Class components have been supported since the beginning and are still supported in React	Function components were always supported in React, but they could not hold any state until React version 16.8

Class vs Function Components

Backwards Compatibility | State and Props | Lifecycle Methods | Verbosity

Class Component	Function Component
In class components, for state we use setState and for props we use parameters passed in the render method	In function components, we use the useState hook function to create state, which when changed will cause a re-render of the component

Class vs Function Components

Backwards Compatibility | State and Props | Lifecycle Methods | Verbosity

Class Component	Function Component
In class components, we can just override lifecycle methods like componentDidMount, etc.	In function components, we use hooks like useEffect to define methods for the component lifecycle

Class vs Function Components

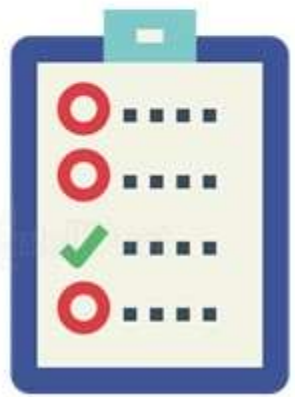
Backwards Compatibility	State and Props	Lifecycle Methods	Verbosity

Class Component	Function Component
Class components are extremely verbose, so many developers prefer using class components only when they are necessary	Function components are really concise. Hence, they are preferred over class components

PropTypes

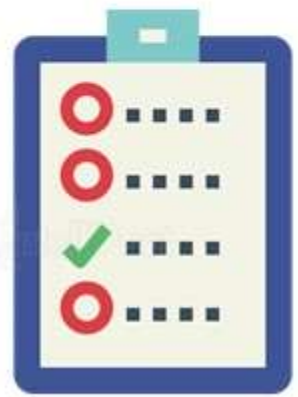
PropTypes

PropTypes allow us to define the shape of the props that our components are expecting



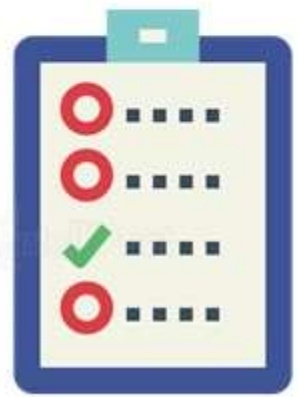
PropTypes

Using PropTypes, we can define what is the data type of a particular prop, if it is required, and more



PropTypes

PropTypes are validated by React automatically, so all we have to do is define the shape and constraints. Then, React will throw an error to let us know whether the props that we passed are incorrect



Hands-on: PropTypes

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

support@intellipaat.com

24/7 Chat with Our Course Advisor