React JS

State Management

Agenda

01 What is State?

02 Local State

03 Global State

04 Lifting up State

05 Context

06 Context: Providers

07 Context: Consumers

08 useContext Hook

What is State?

What is State?

For a component to be truly encapsulated and reusable it needs to be able to access and manipulate some data

What is State?

In a React Components, state is just some data that is managed internally by the component

What is State?

State is managed by react so you don't have to keep track of changes made to state and updating the UI accordingly

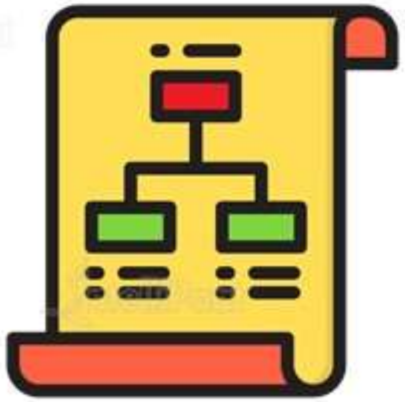
What is State?

There is two types of state in React Components: Local State and Global State

Local State

Local State

Local State is the data that is maintained by a component internally



Local State

Local State is internal to a component and should not be modified by external components

Local State

Local state can be shared with children components via props

Local State

For propagating state changes to parent components we need pass functions which can be called to change the state which causes re render

Global State

Global State

Global State is data that is global in your applications, which means that it's data that needs to be shared across several components

Global State

Since several components need access to the data it cannot be stored in any component and made available for all components

Global State

There are several solutions available in react without the use of any other external library for global state management

Global State

There are two ways of dealing with global state, these are lifting state up and context

Lifting up State

Lifting up State

In ReactJS when we have to share some portion of state with multiple components we can use a technique called lifting up state

Lifting up State

Lifting Up State is the process of moving the shared portion of state to a common parent element

Lifting up State

To lift up the state we need to figure out the state that needs to be shared and components to be shared among

Lifting up State

This state then needs to be moved to the lowest common ancestor component and passed down to needed components as props

Hands On: Lifting up State

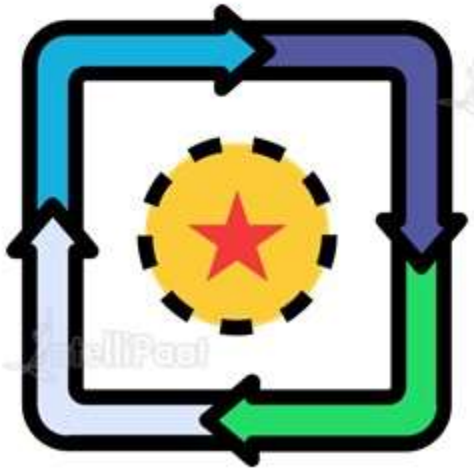
Context

Context

Lifting up state is a good first step but it leads to some problems when used with state that needs to be shared with child components nested deep within the component tree

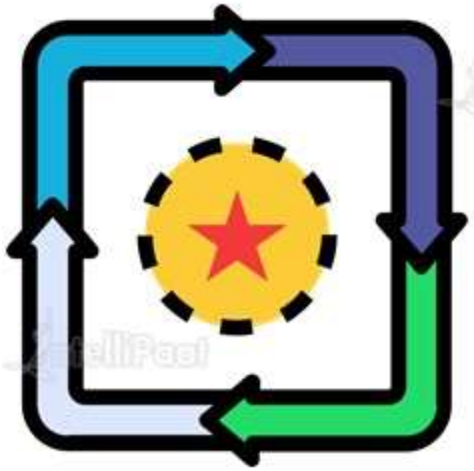
Context

When state needs to be shared with components that are deeply nested they need to be passed through many components through props



Context

In order to avoid this issue we can use context in react, which is a built in solution in react for exact same issue



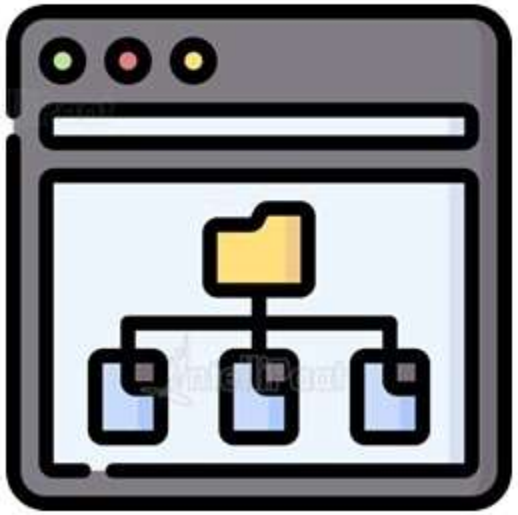
Context

To Create react context we use createContext function from react, However to use a react context we need to understand two parts of react context namely Providers and Consumers

Context Providers

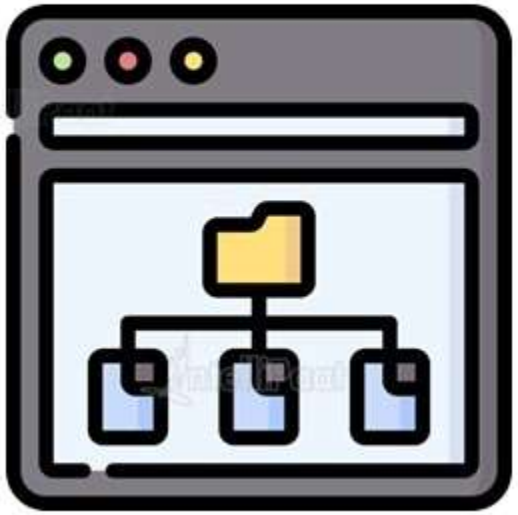
Context Providers

When we create a context it contains two React Components named Providers and Consumers



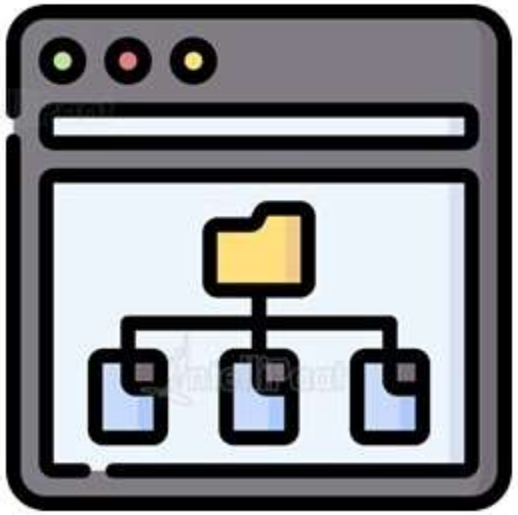
Context Providers

Providers need to be the parent components to all the Components that need to use data within a context



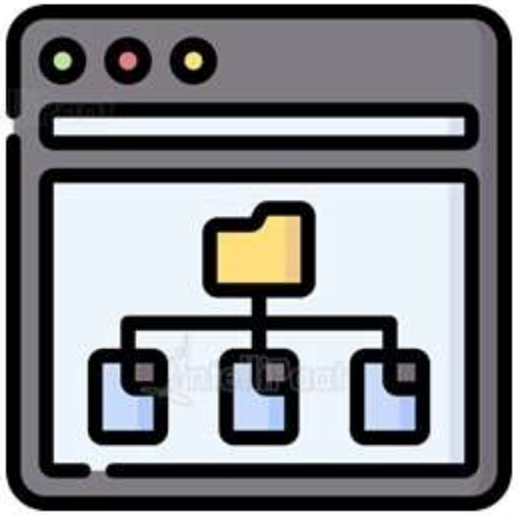
Context Providers

A Providers basic job is to provide values to any child component that explicitly asks for it



Context Providers

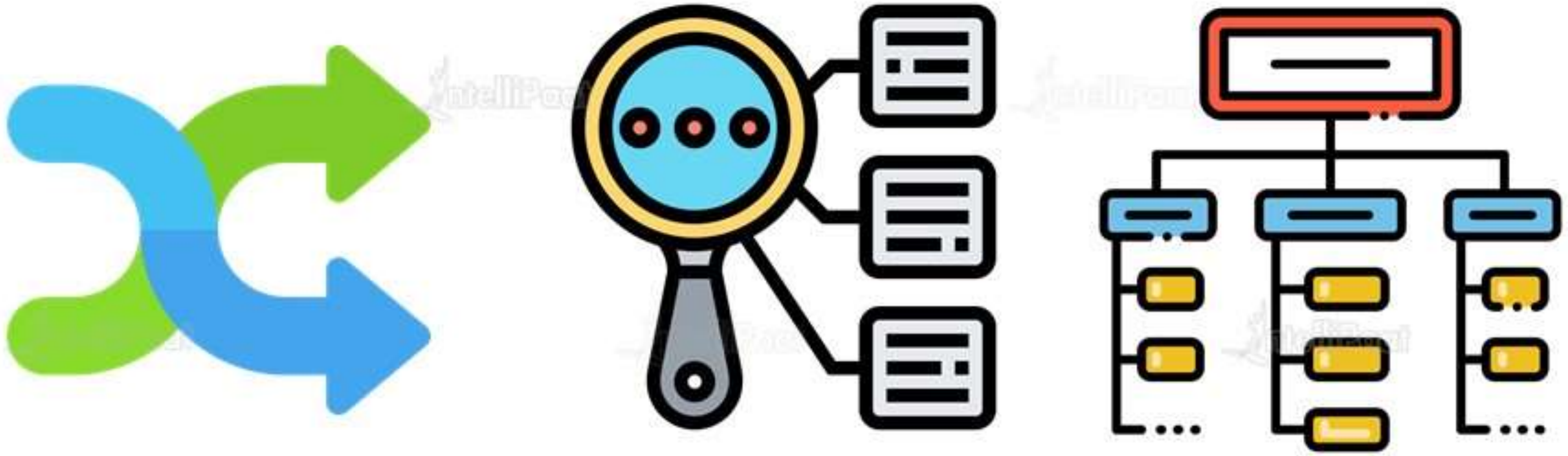
Providers can contain any kind of value, including objects that contain data and functions to manipulate that data



Context Consumers

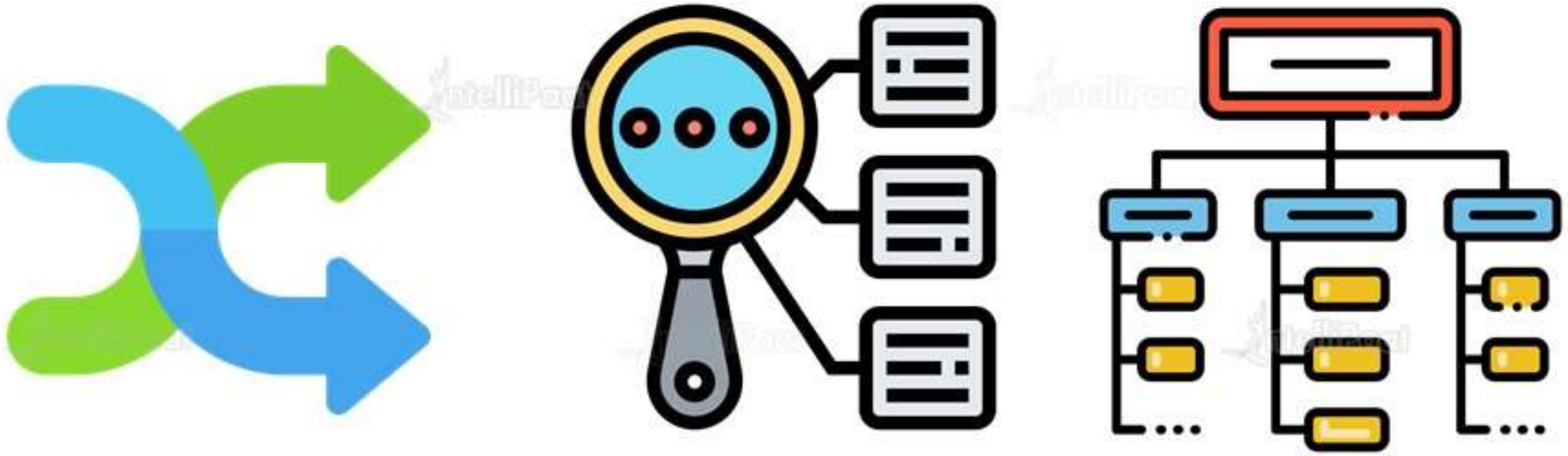
Context Consumers

Consumers are the counterpart of the Provider component of a context



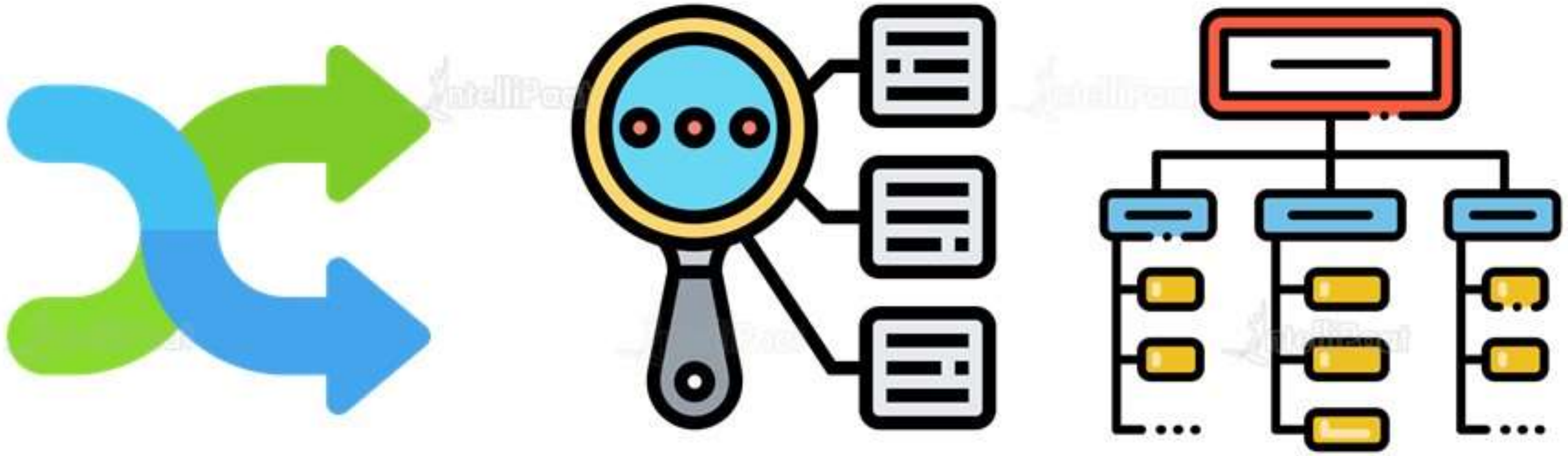
Context Consumers

Consumers are the child components or descendants of providers that have access to the value that Provider holds



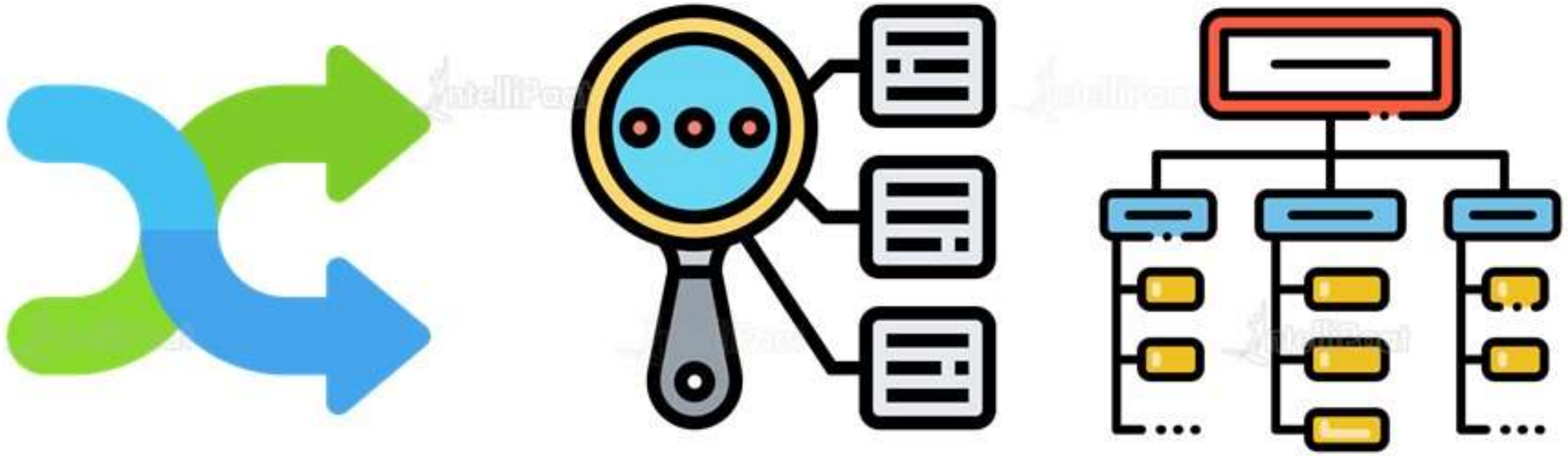
Context Consumers

Consumer uses render prop pattern, in which we have to pass a function which has access to value in the context and returns a react components



Context Consumers

Consumer component walk up the parent component chain and find the nearest matching Provider to access value from

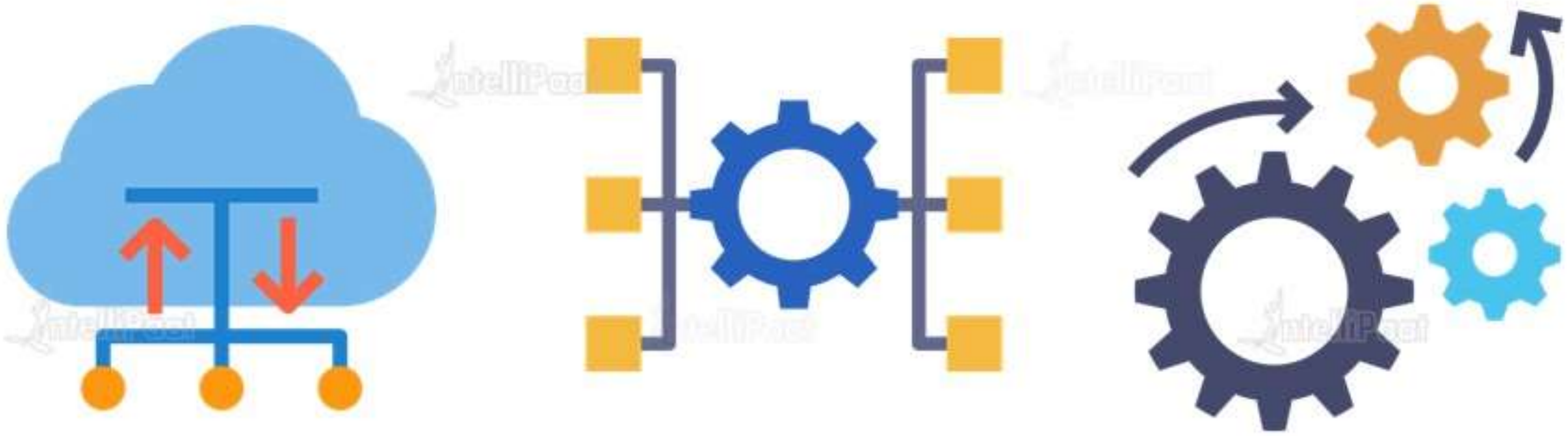


Hands On: React Context

useContext hook

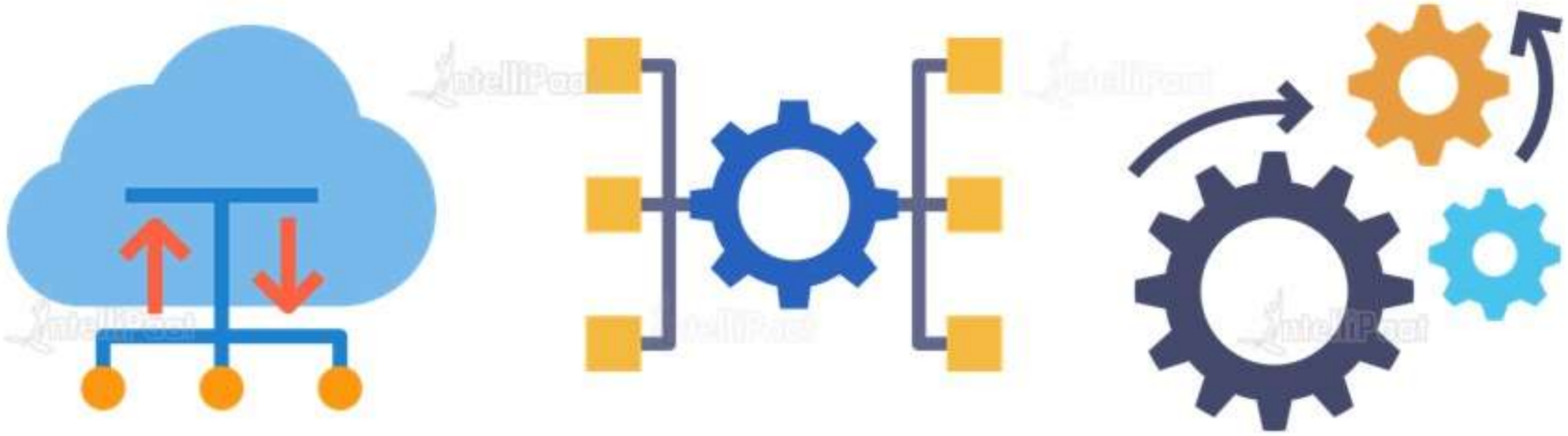
useContext hook

As using Consumer components in react requires you to need to use complicated pattern like render prop we can also use useContext hook



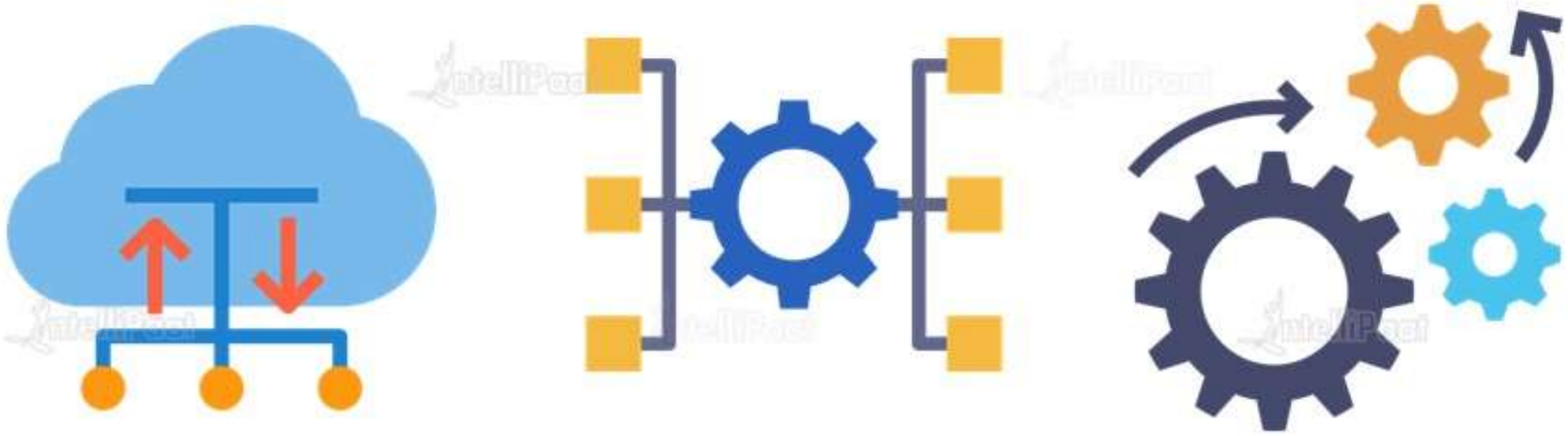
useContext hook

In useContext hook we do not need to use any other component all we have to do is is pass the context as an argument to the hook



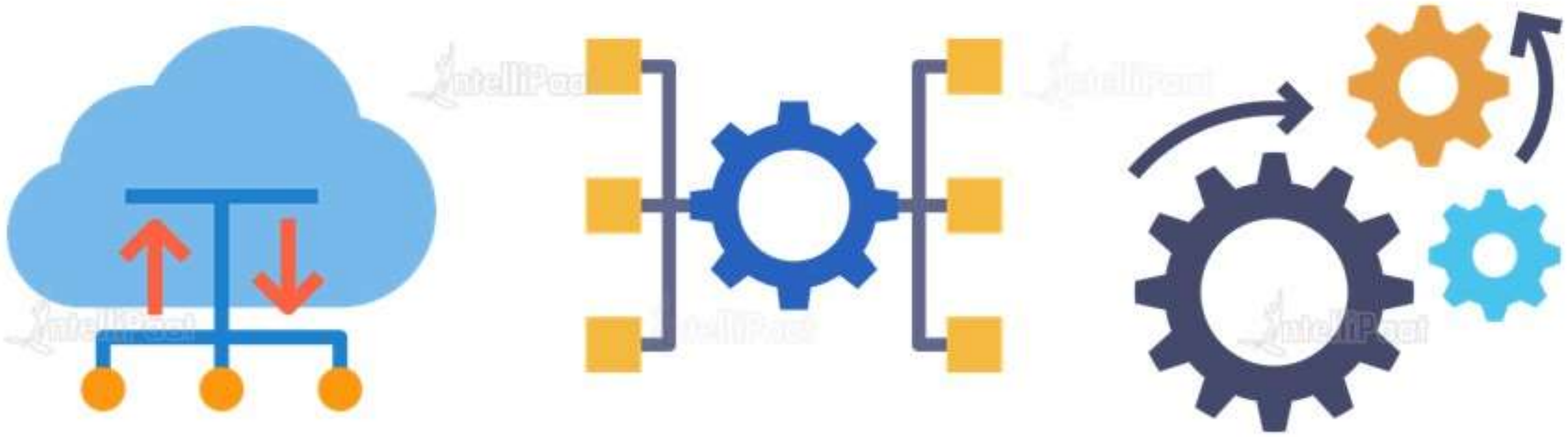
useContext hook

With useContext you can simply access value by calling the method and without using render prop pattern



useContext hook

It works the same way as Consumer context by getting the value from a matching Provider component that is the current components parent



Hands On: useContext hook

India: +91-7847955955

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

support@intellipaat.com

24/7 Chat with Our Course Advisor