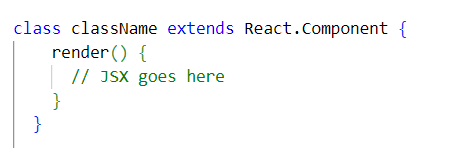
<https://cedar-bottle-e6c.notion.site/Class-Based-Components-679a922a492247049aadc6d5077623f6>

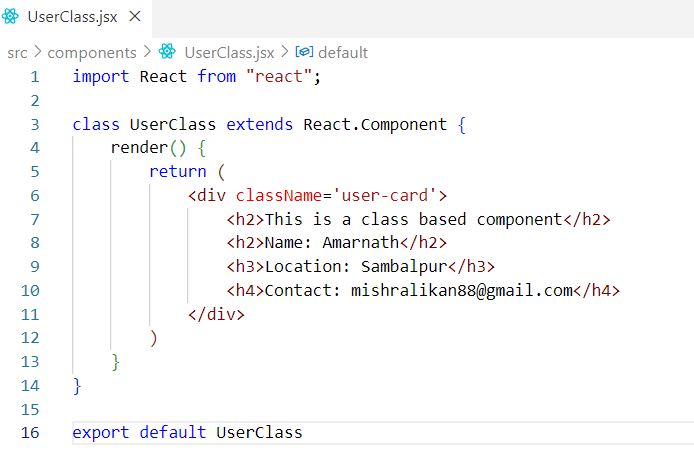
What is a Class-based Component in React?

A **class-based component** is a regular JavaScript class that extends React.Component. The syntax for creating a class-based component is:



* **React.Component**: This is the base class that your component extends. It provides access to various React features, including lifecycle methods and the render method. These features are available inside your class because it inherits from React.Component.
* **Render method**: Every class-based component must implement the render method. This method returns the JSX that defines the component's UI.

Creating a Class based component

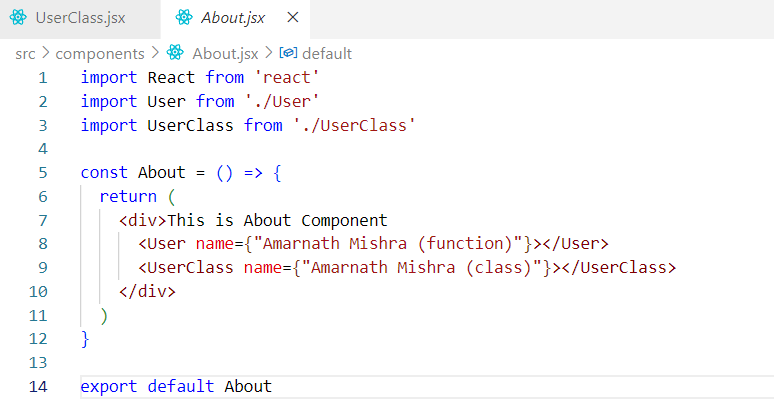


UserClass is a class that extends React.Component and includes a render method that returns JSX.

What is React.Component?

React.Component is a built-in class provided by React. When creating a class-based component, you extend this class to gain access to React's built-in methods, such as render (), componentDidMount (), and others that are part of the React lifecycle. To use React.Component in your component file, you must import React, as it is part of the React package. Without importing React, you won't be able to extend React.Component to create a class-based component.

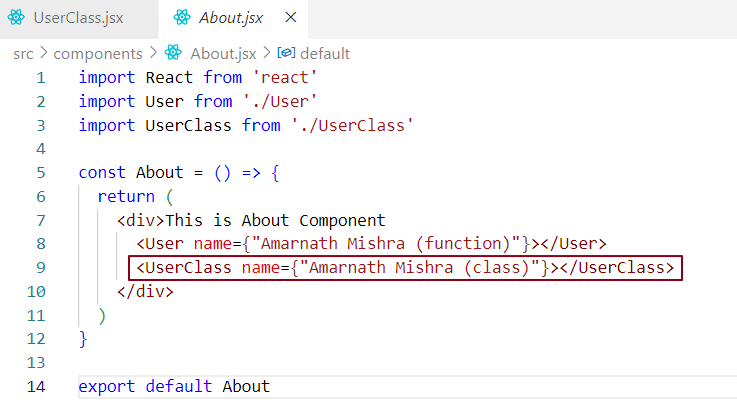
How do we receive props in class-based component?





In our case, UserClass is a class-based component that receives props from its parent component, About.jsx. This means the parent component passes data to the child class component UserClass, via props.

How the code works?



Step by step deep down process

**1. Instance Creation:**

When an instance of child class component UserClass is created (i.e., when it is rendered within the parent About.jsx component, the constructor of the child class component UserClass is called.

**2. Child Constructor (UserClass) Execution**

* The constructor(props) method of UserClass is called, where the props are passed as an argument.
* These props are provided by the parent component when rendering UserClass.

**3. Parent Constructor (React.Component) Execution**

* Inside UserClass, you call super(props), which invokes the constructor of the parent class (React.Component).
* This is where React internally sets up the **props** and **state** for the component.
* this.props = props; sets the props that were passed to the component.
* this.state = null; initializes the component's state, which can be customized in UserClass.

**4. Completion of UserClass Constructor**

* After calling super(props), the constructor of UserClass continues, allowing additional setup, such as logging props, setting initial state of the USerClass, or any other custom initialization.

**5. Render Method:**

* Once the constructor completes, react automatically calls the render() method of UserClass.
* The render() method is responsible for returning JSX that represents the component's UI.
* In the render() method, you can access both this.props (which was set via the constructor) and this.state to render the component.

**Flow Summary:**

1. When the component instance is created the UserClass constructor is called.
2. super(props) calls the React.Component constructor.
3. The React.Component constructor sets up this.props and this.state.
4. The UserClass constructor can now use this.props and this.state for further initialization.
5. Finally, the render() method is called to render the component's UI.

To access the pro we use this in cbc ..lik this.prop.propnsame……………

Or we can destrcute tese proeprties from this,prop anduse it right away …short hand syntax …

Add ss

**Loading a class-based component on our Web page = creating an instance of that class and giving it props to it if required.**

### ****State in Class-Based Components****

In React, **state** is a JavaScript object that holds information or data about a component. This data is private to the component and determines how the component behaves and renders.

In a **class-based component**, state is created inside the constructor and can be updated using the setState method. Let’s dive deeper into how it works and the internal mechanisms behind it.

**Whenever we create an instance of a class, the class constructor is called and constructor is the bets place that can props and this is the best place to create state variables.**



**Deos that mean in this .state we are ioverriding thedeaflt state initialisation in reacr.componet which is this.state = null**

**Late r its this.satte = {count …..}**

**Sending state from parent to child in cbc = same as fbc**

**Receiving / Accessing state in cbc ..use this ..this.sgtate.<stateName>**

**You can destrcuture this as wlle to aboid …..**

**Create the state in FBC andcompare both ..**

**Note – In function-based component. Behind the scene react creates a big state object and push the state variables into this**

**39>>>>>**

**Never update state variable directly by accessing tis.state.count = this,statecount+1**

**Sattach ss**

**Because this will cretaeet inconsitencey across your app..samefor fbc**

**Then howto update properly ?  
use this.setState({modified object}) insidea thehandler calbackfunction.**

**Attache ss**

**When setter is invoked rerender cycle oif react will be triggered …**

**Batch updates … scenatio count 21 count 2 ..cover this as well**

**Where this set count is coming from>>**

**Modified object – count:this.state.count + 1**

**Update state variables in cbc-**

**Setter is coming from retc.componet as well**

**Internatl representation of reeqct.componnet..**

**// This is a very simplified version (React source is written in Flow/TypeScript)**

**class Component {**

**constructor(props) {**

**this.props = props;**

**this.state = null; // set by user**

**}**

**// This method is inherited by your component**

**setState(partialState, callback) {**

**// Internally schedules an update (not immediate)**

**// Merges partial state with existing state**

**enqueueSetState(this, partialState, callback);**

**}**

**// Placeholder — must be overridden by your class**

**render() {d**

**throw new Error('Component subclasses must implement render()');**

**}**

**}**

**export { Component };**

**Lifecycle cbc – how the cbc is mounterd on awebpage**

**48.38 >>>>**

Make notes ... consolidate everything

Setter >>> count increase

How setState works behind the scene in class-based component ...



why can’t you use async in useEffect callback?

Use of async in component did mount?

**Lifecycle cbc – how the cbc is mounterd on awebpage**

**48.38 >>>>**