



#RahoAmbitious



Full Stack Software Development

Course: Advanced Frontend Development Using React

Lecture on: React State and Life-Cycle

Instructor: Parth Dalal

TODAY'S AGENDA

- What is State in react?
- React life-cycle methods
- Other API methods

React State

REACT STATE AND LIFE-CYCLE

Before moving to react state, let's understand what declarative programming is, on which react is solely based.

- Declarative programming is one of the core features of React components
- Changes to visual elements are driven using data

REACT STATE AND LIFE-CYCLE

For reference, let's consider the following UI, which is a checkout screen.

1. Tracing Sheets A3 for Artists	<input type="text" value="1"/> <input type="button" value="X"/>	\$ 4.55
2. Water Colour Pencils	<input type="text" value="1"/> <input type="button" value="X"/>	\$ 2.90
3. Oil Pastel – 50 Shades	<input type="text" value="1"/> <input type="button" value="X"/>	\$ 3.00
Continue to Payment		Total \$ 10.45

REACT STATE AND LIFE-CYCLE

The data that drives this ui is as follows:

```
cart = [  
  {  
    product: "Tracing Sheets A3 for Artists",  
    quantity: 1,  
    cost: 4.55  
  },  
  {  
    product: "Water Colour Pencils",  
    quantity: 1,  
    cost: 2.9  
  },  
  {  
    product: "Oil Pastels - 50 Shades",  
    quantity: 1,  
    cost: 3.0  
  }  
];
```

1. Tracing Sheets A3 for Artists	<input type="text" value="1"/> <input type="button" value="x"/>	\$ 4.55
2. Water Colour Pencils	<input type="text" value="1"/> <input type="button" value="x"/>	\$ 2.90
3. Oil Pastel – 50 Shades	<input type="text" value="1"/> <input type="button" value="x"/>	\$ 3.00
<div>Continue to Payment</div>		Total \$ 10.45

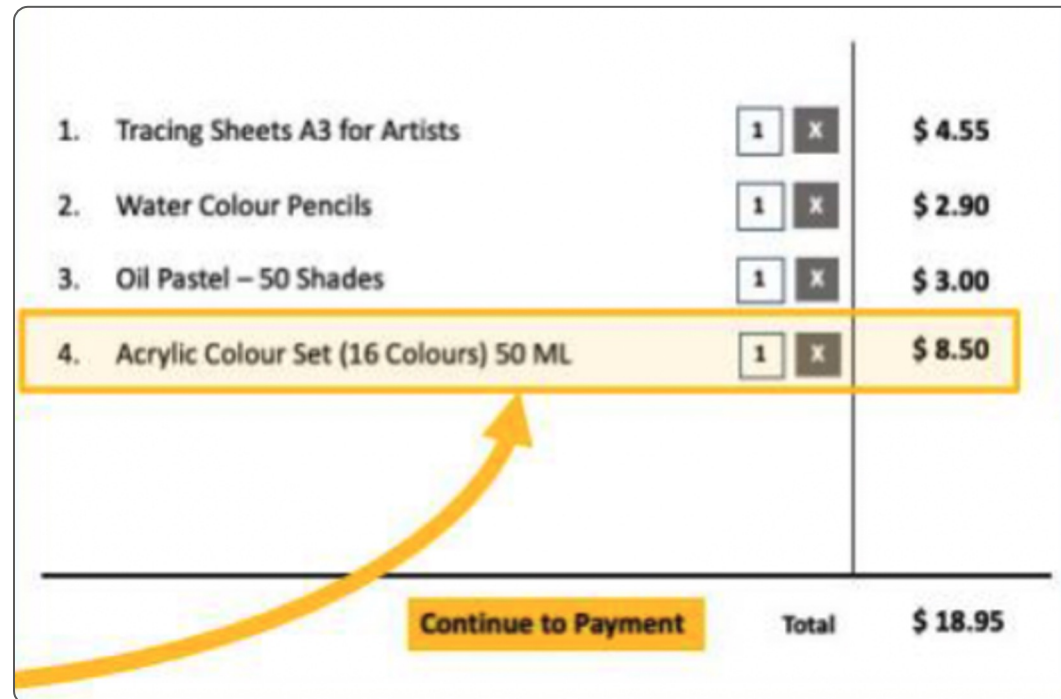
REACT STATE AND LIFE-CYCLE

Now, let's add one more data like this:

```
{  
  product:"Acrylic Colour Set(16 Colours)  
  50ML",  
  quantity:1,  
  cost:8.5  
}
```


REACT STATE AND LIFE-CYCLE

Then, the UI comes out to be:



1. Tracing Sheets A3 for Artists	<input type="text" value="1"/> <input type="button" value="X"/>	\$ 4.55
2. Water Colour Pencils	<input type="text" value="1"/> <input type="button" value="X"/>	\$ 2.90
3. Oil Pastel – 50 Shades	<input type="text" value="1"/> <input type="button" value="X"/>	\$ 3.00
4. Acrylic Colour Set (16 Colours) 50 ML	<input type="text" value="1"/> <input type="button" value="X"/>	\$ 8.50
<hr/>		
<div>Continue to Payment</div>		Total \$ 18.95

Adding another product is as simple as adding a product to the array on the left. It automatically updates the UI on the right.

REACT STATE AND LIFE-CYCLE

React State

- React uses the concept of state, which essentially describes the contents of the interface.
- So, whenever there is a change in state, the ui updates automatically.
- So, we declaratively update the ui by updating the underlying state.

REACT STATE AND LIFE-CYCLE

Some of the important points regarding state can be summarised as follows:

- A state can be maintained inside a class component only.
- A state is always initialised inside the class constructor.
- If you define the constructor of a class, you need to call the `super()` method in the first statement of the constructor definition. This method calls the constructor of the parent class.
- To set the state, you must always use the `setState()` method and must never directly manipulate the application's state. However, `setState()` should never be called inside the constructor.
- Whenever a state is changed, the `render()` method of the class is called again.

REACT STATE AND LIFE-CYCLE

As discussed earlier, there are two types of components:

- **Class** components, also known as stateful components
- **Functional** components, also known as stateless components
- As the name states, class contains state object that handles the data of the component, whereas functional components accept data using props and have no state object inherently.

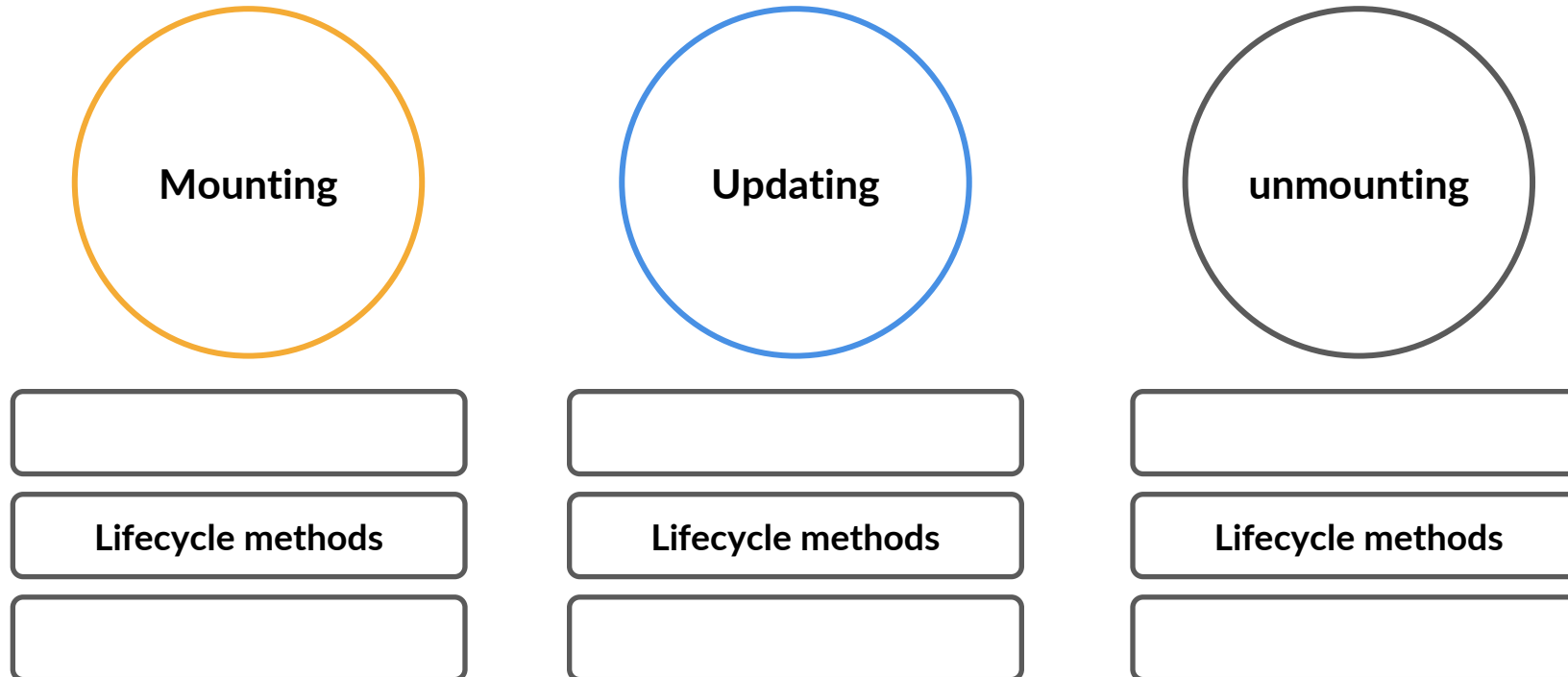
REACT STATE AND LIFE-CYCLE

There are mainly three stages of React life-cycle:

- Mounting
- Updating
- Unmounting

In each stage of this life-cycle, various life-cycle methods are used to define how the complete react application works. We will go one by one.

REACT STATE AND LIFE-CYCLE



REACT STATE AND LIFE-CYCLE

Mounting Phase

The mounting phase occurs when a React component instance is created and inserted into the DOM.

Life-cycle methods include:

- **constructor()** method,
- **render()** method and
- **componentDidMount()** method.

REACT STATE AND LIFE-CYCLE

```
class Greeting extends Component {  
  constructor(props) {  
    super(props); // mandatory to call because we  
    are
```

subclassing from component

class.

```
    this.state = {  
      isLoading: false,  
      userName: "", //initial value setting using  
      state object  
      shoppingList: []  
    }  
    this.onClickHandler =  
    this.onClickHandler.bind(this);  
    //Binding methods to the context  
  }  
}
```

constructor() Method

The constructor() method is called before the component is mounted into the DOM. It is used to initialise local state variables and bind class methods such as this example.

Poll 1

Which among the following are true for React state?

(**Note:** More than one option may be correct.)

- A. A parent component can change the state within a child component.
- B. Changing the state of a component will cause it to be re-rendered.
- C. Mutating the state directly, for example, using the assignment operator, will cause the component to be re-rendered.
- D. Functional components cannot have a state.

Poll 1 (Answer)

Which among the following are true for React state?

(Note: More than one option may be correct.)

- A. **A parent component can change the state within a child component.**
- B. **Changing the state of a component will cause it to be re-rendered.**
- C. Mutating the state directly, for example, using the assignment operator, will cause the component to be re-rendered.
- D. Functional components cannot have a state.

Poll 2

Choose the correct statement using which you think you can fix the error message given below.

'this' is not allowed before super()

- A. You need to reference the super() method after referencing 'this' keyword inside the constructor of a class.
- B. You need to reference the super() method before referencing 'this' keyword inside the constructor of a class.
- C. You need to pass 'this' as an argument to the super() method inside the constructor of a class.
- D. You cannot reference 'this' keyword inside the constructor of a class.

Poll 2 (Answer)

Choose the correct statement using which you think you can fix the error message given below.

'this' is not allowed before super()

- A. You need to reference the super() method after referencing 'this' keyword inside the constructor of a class.
- B. You need to reference the super() method before referencing 'this' keyword inside the constructor of a class.**
- C. You need to pass 'this' as an argument to the super() method inside the constructor of a class.
- D. You cannot reference 'this' keyword inside the constructor of a class.

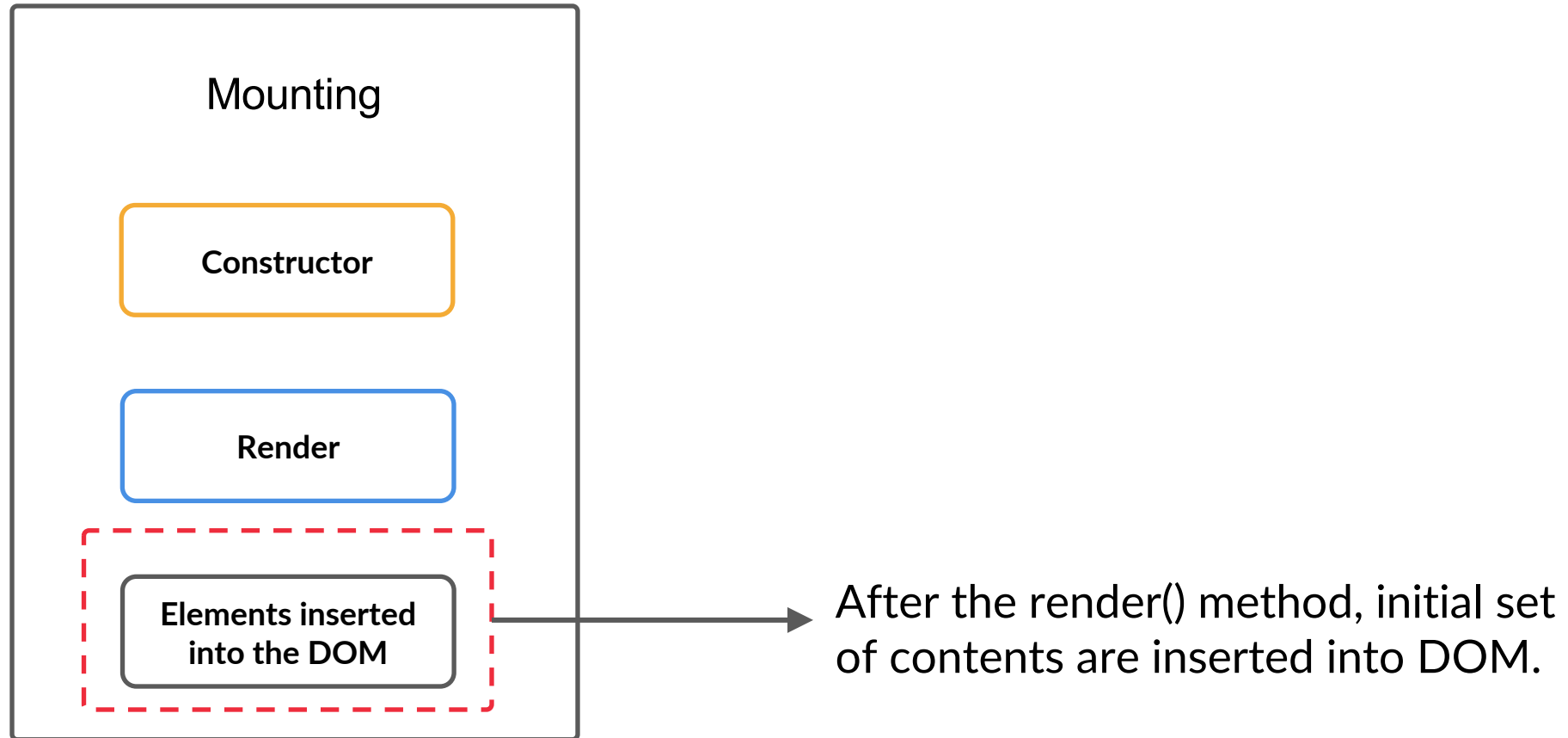
REACT STATE AND LIFE-CYCLE

render() Method

This life-cycle method is only needed if you are building class components and its purpose is to return a React element. This is also why the render() method should stay pure. The render() method must never modify the state and should always return the same predictable result.

```
class Card extends Component {
  state = {
    fullName: "Johnny Doe",
    designation: "VP - Sales"
  };
  render() {
    return <div className="v-card">
      <div
        className="name">{this.state.fullName}</div>
      <div
        className="designation">{this.state.designation}</div>
    </div>
  }
}
```

REACT STATE AND LIFE-CYCLE



REACT STATE AND LIFE-CYCLE

Updating

Any component can be updated by introducing changes to props or state. Within an updating process, the following methods are called when a component is re-rendered:

- **render()**
- **componentDidUpdate()** - Not called for the initial render

Note: There are times when you do not want the component's output to be affected due to any change in props or state. In such cases, the *shouldComponentUpdate()* method is used. It is invoked while receiving new props or state, that is, before rendering.

REACT STATE AND LIFE-CYCLE

Updating Phase

Updating phase occurs when you interact with a component and its re-renders.

This interaction includes receiving updated data through:

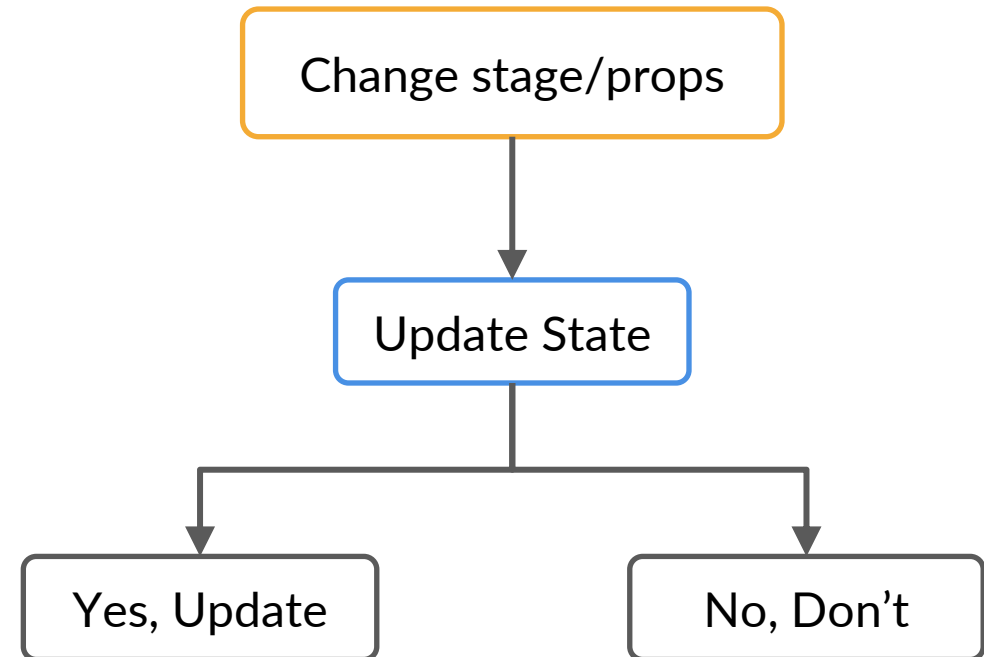
- Props and
- Update to the state.

In both cases, the updating phase begins with an invocation to the `getDerivedStateFromProps()` method.

REACT STATE AND LIFE-CYCLE

`getDerivedStateFromProps()` Method

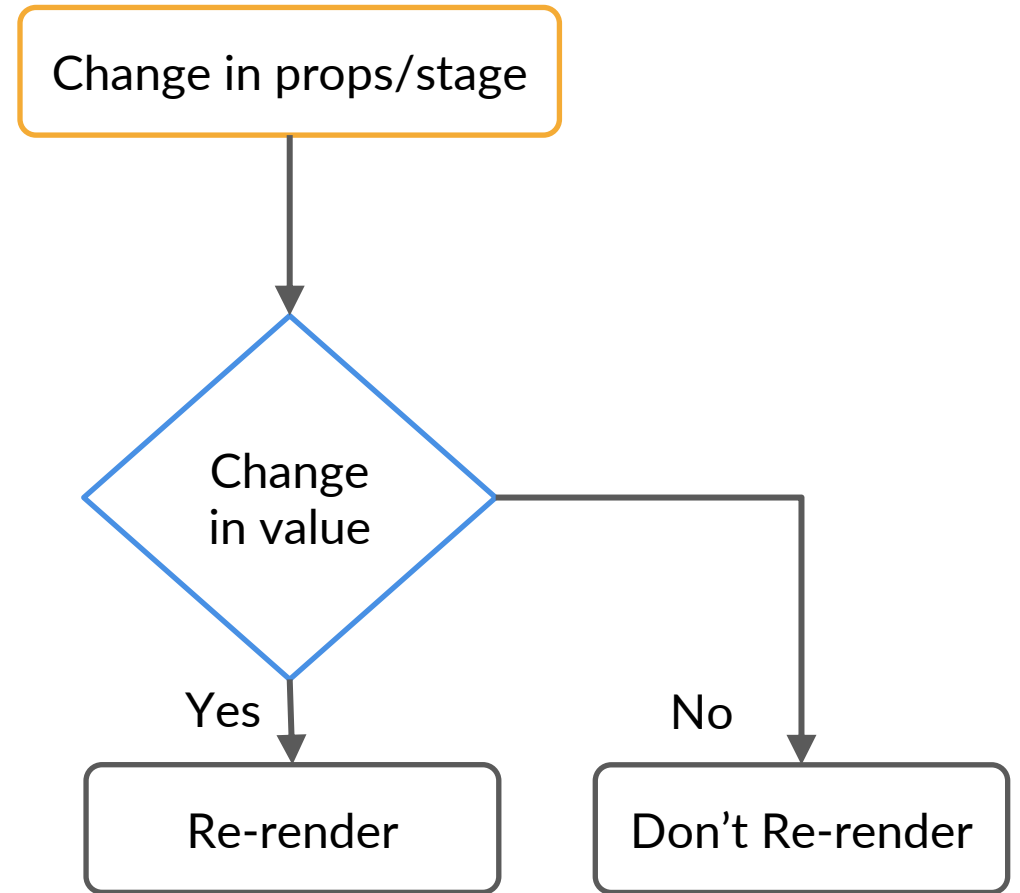
This method allows you to decide whether, based on the changes to data in the props, you want the state to be updated or not.



REACT STATE AND LIFE-CYCLE

shouldComponentUpdate() Method

This function lets you decide if, based on changes to data in state or props, you want the component to re-render. This helps in prevention of DOM reconciliation for components if the data in props and state has not changed, thereby boosting performance.



REACT STATE AND LIFE-CYCLE

- In large applications where components are deeply nested, the efficiency and performance can be improved by using the `shouldComponentUpdate()` method as it prevents instances from re-rendering just because their parent component was refreshed.
- Now, instead of implementing this method for optimisation, React also offers something known as a Pure Component, which automates this process.

REACT STATE AND LIFE-CYCLE

A Pure Component has built-in `shouldComponentUpdate()` and doesn't need to be manually set up as in the case of a component instance.

The built-in `shouldComponentUpdate()` performs a shallow comparison of state and props and offer no customisation of behaviour.

A Pure Component will prevent its subtrees from re-rendering.

```
import React, {PureComponent} from
"react";

class Notification extends PureComponent {
  render() {
    return <div
      className="notification">{this.props.messa
    ge}</div>
  }
}
```

REACT STATE AND LIFE-CYCLE

getSnapshotBeforeUpdate() Method

- This function lets you compare current and previous values in state and props, but this time you have access to DOM nodes before React implements them.
- This means that you can access and update DOM properties using React references.

REACT STATE AND LIFE-CYCLE

getSnapshotBeforeUpdate() Function Example

```
getSnapshotBeforeUpdate(prevProps, prevState) {  
  
    // Displaying the previous value of the state  
    document.getElementById('prev').innerHTML =  
        'Previous Name: ' + prevState.name;  
}
```

Full code [here](#)

REACT STATE AND LIFE-CYCLE

ComponentDidUpdate() Function

- This function is useful when a component needs to decide if, based on changes in state or props, it needs to perform side effects such as querying an API for more data.
- This method is not invoked on the first render of a component and is only available when a component updates.

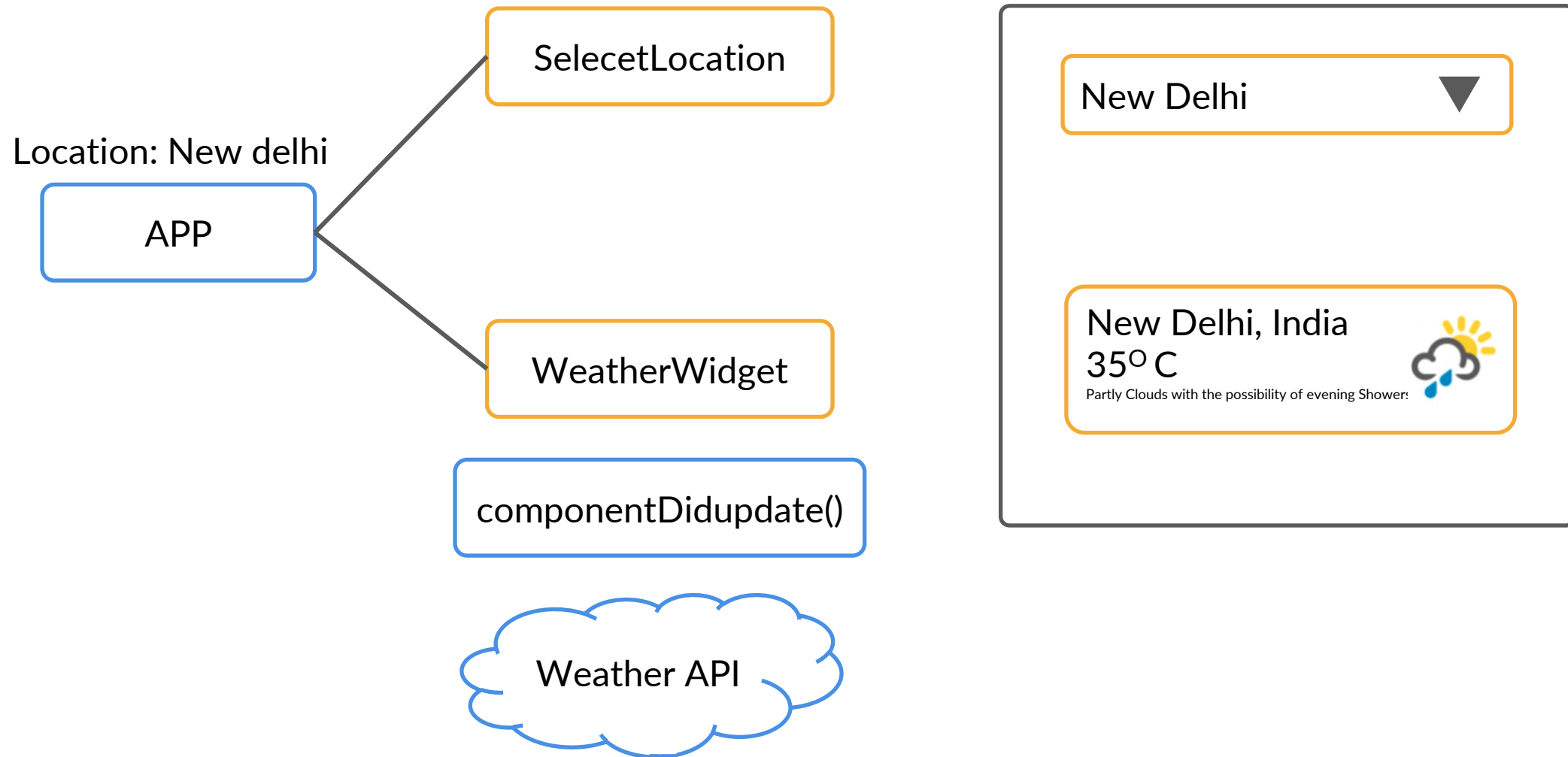
REACT STATE AND LIFE-CYCLE

ComponentDidUpdate() Function Example

```
componentDidUpdate() {  
  
    // Displaying the current value of the state  
    document.getElementById('new').innerHTML =  
        'Current Name: ' + this.state.name;  
}
```

Full code [here](#)

REACT STATE AND LIFE-CYCLE



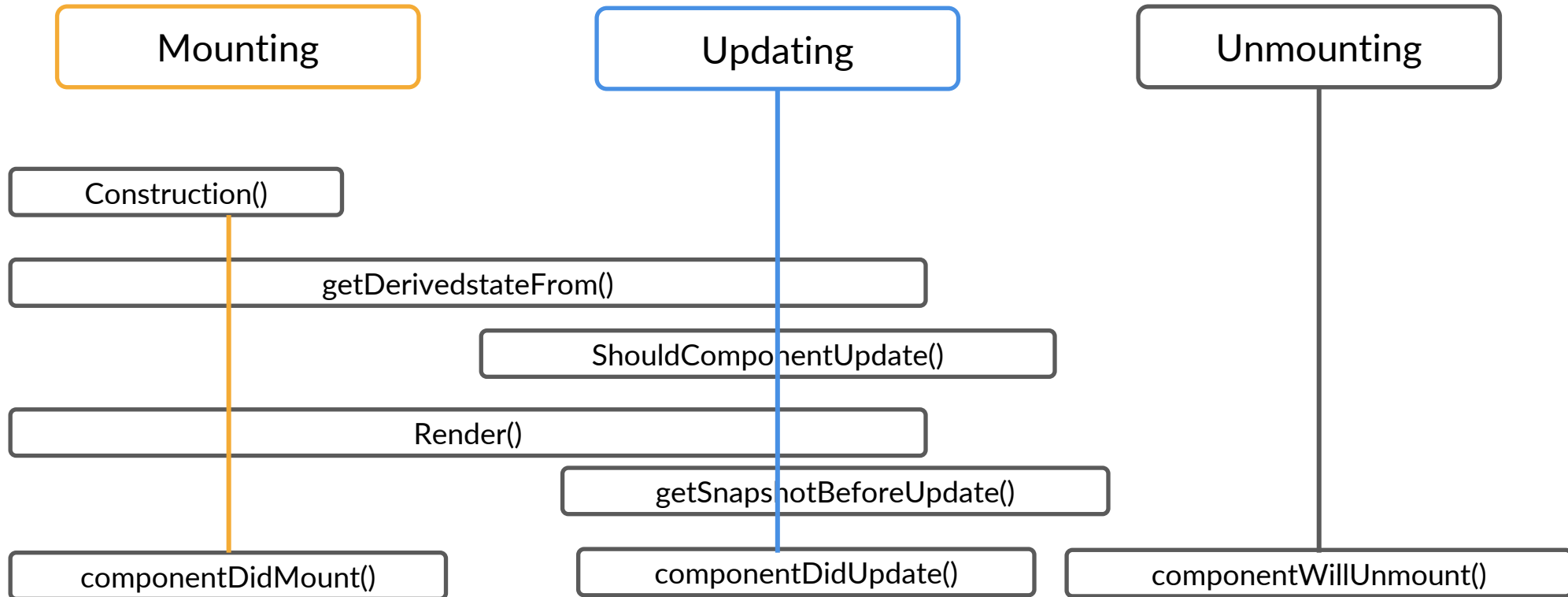
In this example, if you change the dropdown, you can use `componentDidUpdate()` to fetch data from weather API.

REACT STATE AND LIFE-CYCLE

Unmounting Phase

- This is the final phase that the component goes through.
- This phase offers only one method, known as `componentWillUnmount()`, and occurs right before the component is destroyed.
- This method is useful for cleanup, such as removing timer instances and detaching or unsubscribing from real-time data sources, such as Websockets, when a component is removed.
- This method also lets your free up memory when the component is destroyed and unmounted.

REACT STATE AND LIFE-CYCLE



Summary

Poll 3

When is the *componentDidMount* life-cycle method called?

- A. Component is updated
- B. Component is created for the first time
- C. Both of the above
- D. None of the above



Poll 3 (Answer)

When is the *componentDidMount* life-cycle method called?

- A. Component is updated
- B. Component is created for the first time**
- C. Both of the above
- D. None of the above



Poll 4

Which of the following methods is NOT used to define a component's life-cycle inside the mounting process?

- A. constructor()
- B. render()
- C. componentDidMount()
- D. componentDidUpdate()

Poll 4 (Answer)

Which of the following methods is NOT used to define a component's life-cycle inside the mounting process?

- A. constructor()
- B. render()
- C. componentDidMount()
- D. **componentDidUpdate()**



Poll 5

Which method should be overridden in order to stop a React component from updating?

- A. `componentNotUpdate()`
- B. `componentDidUpdate()`
- C. `willComponentUpdate()`
- D. `shouldComponentUpdate()`



Poll 5 (Answer)

Which method should be overridden in order to stop a React component from updating?

- A. `componentNotUpdate()`
- B. `componentDidUpdate()`
- C. `willComponentUpdate()`
- D. **`shouldComponentUpdate()`**



REACT STATE AND LIFE-CYCLE

getDerivedStateFromError()

- It is invoked if some error has occurred during the rendering phase of any life-cycle methods or any children components.
- This method is mainly used to implement the Error Boundaries for the React application.
- It is called during the render phase, so side-effects are not permitted in this method.

REACT STATE AND LIFE-CYCLE

getDerivedStateFromError() Example

Example [here](#)

REACT STATE AND LIFE-CYCLE

componentDidCatch()

- This method is invoked if some error occurs during the rendering phase of any life-cycle methods or any children components.
- This method is used to implement the Error Boundaries for the React application.
- It is called during the commit phase, so unlike `getDerivedStateFromError()`, which is called during the render phase, side-effects are allowed in this method. This method is also used to log errors.

Syntax: `componentDidCatch(error, info)`

REACT STATE AND LIFE-CYCLE

componentDidCatch() Example

```
componentDidCatch(error) {  
    // Changing the state to true  
    // if some error occurs  
    this.setState({  
        error: true  
    });  
}
```

Full code [here](#)

REACT STATE AND LIFE-CYCLE

setState()

- The setState() method is used to change the state object. It ensures that the component has been updated and calls for re-rendering of the component.
- setState is asynchronous call, which means if synchronous call gets called, it might not get updated at the right time. For instance, if you attempt to find the current value of an object after an update using setState, it might not give you the current updated value on console.

```
setState({ stateName : updatedStateValue })
```

```
// OR
```

```
setState((prevState) => ({  
  stateName: prevState.stateName + 1  
}))
```

REACT STATE AND LIFE-CYCLE

forceUpdate()

- The components in React will re-render only if the state of the component or the props passed to it change. But if you need to re-render the component if some data changes, then you use the forceUpdate() method of React.

Example [here](#)

REACT STATE AND LIFE-CYCLE

Class Properties

Default Props

- The `defaultProps` is a React component property that allows you to set default values for the props argument. If the prop property is passed, it will be changed. The `defaultProps` can be defined as a property on the component class itself to set the default props for the class.

DisplayName

- The `displayName` string is used to debug messages. Usually, it does not need to be set explicitly because it is inferred from the name of the function or class that defines the component.

REACT STATE AND LIFE-CYCLE

Instance Properties

Props

- `this.props` contains the props that were defined by the caller of this component. These props are passed from parent component to child component.

State

- The state is a plain javascript object that contains data specific to this component that may change over time. The state is user-defined.

KEY TAKEAWAYS

- A state is controlled within a component unlike props that are controlled by a parent component. Also, a change in the state calls the `render()` method again.
- A state can be maintained inside a class component and is always initialised inside the class constructor.
- If you define the constructor of a class, you need to call the `super()` method in the first statement of the constructor definition. This method calls the constructor of the parent class.
- To set the state, you must always use the `setState()` method and must never directly manipulate the application's state. However, `setState()` should never be called inside the constructor.

KEY TAKEAWAYS

- The component life-cycle in React varies from one process to another and, in total, there are three processes, which are *mounting*, *updating*, and *unmounting*.
- Inside the mounting process, a component's life-cycle is defined by the following methods, which are called in the given order:
 - constructor()
 - render()
 - componentDidMount()



Thank You!