WEB TECHNOLOGIES

React JS – Component States and LifeCycle Methods

Prof. Vinay Joshi and Dr. Sarasvathi V
Department of Computer Science and Engineering

Acknowledgement
The slides are created from various internet resources with valuable contributions from multiple professors and teaching assistants in the university.

Component States and LifeCycle Methods

Agenda

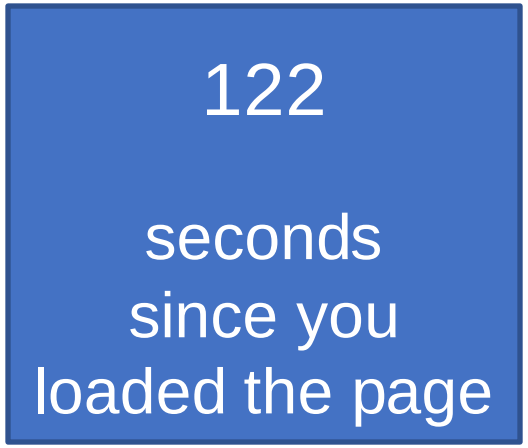
PES UNIVERSITY

- Introduction to Life cycle
- Pictorial Representation of methods
- Life cycle methods in brief
- Key Methods in the React Component Lifecycle: From Mounting to Unmounting
- Demo

React.JS – Stateful Components

Counter Component

- Let’s consider this Component that shows the number of seconds the user has been on the page



React.JS – Stateful Components

Counter Component…(cntd.)

- We consider two Components, CounterDisplay and Counter

```
ReactDOM.render(
        <CounterDisplay/>,
        document.querySelector("#container")
);
```

```
class CounterDisplay extends React.Component {
        render() {
                return (
                        <div>
                                <Counter/>
                                <h2>seconds </h2>
                                <h2>since you loaded the page</h2>
                        </div>
                );
        }
}
```

```
class Counter extends React.Component {
        render() {
                return (
                        …
                );
        }
}
```

React.JS – Stateful Components

Counter Component…(cntd.)

- We use the ***constructor*** method to initialize the counter
- The Component also has the method ***componentDidMount*** that can be used to start the counter
- It also exposes the ***setState*** method to update the state (the counter)

React.JS – Stateful Components

Counter Component…(cntd.)

- We use these methods as follows

```
class Counter extends React.Component {
        constructor(props, context) {
                super(props, context);

                this.state = {
                        seconds: 0
                };
        }

        render() {
                return (
                        <h1>{this.state.seconds}</h1>
                );
        }
}
```

Add this code after the constructor:

```
componentDidMount() {
        setInterval(this.timerTick, 1000);
}
timerTick() {
        this.setState({
                seconds: this.state.seconds + 1
        });
}
```

React.JS – Stateful Components

Counter Component…(cntd.)

- To bring in the stateful Component behaviour, that also ensures that state variable seconds is never out of sync

```
this.setState((prevState) => {
    return {
        seconds: prevState.seconds
        + 1
    };
});
```

```
timerTick() {
    this.setState({
        seconds: this.state.seconds + 1
    });
}
```

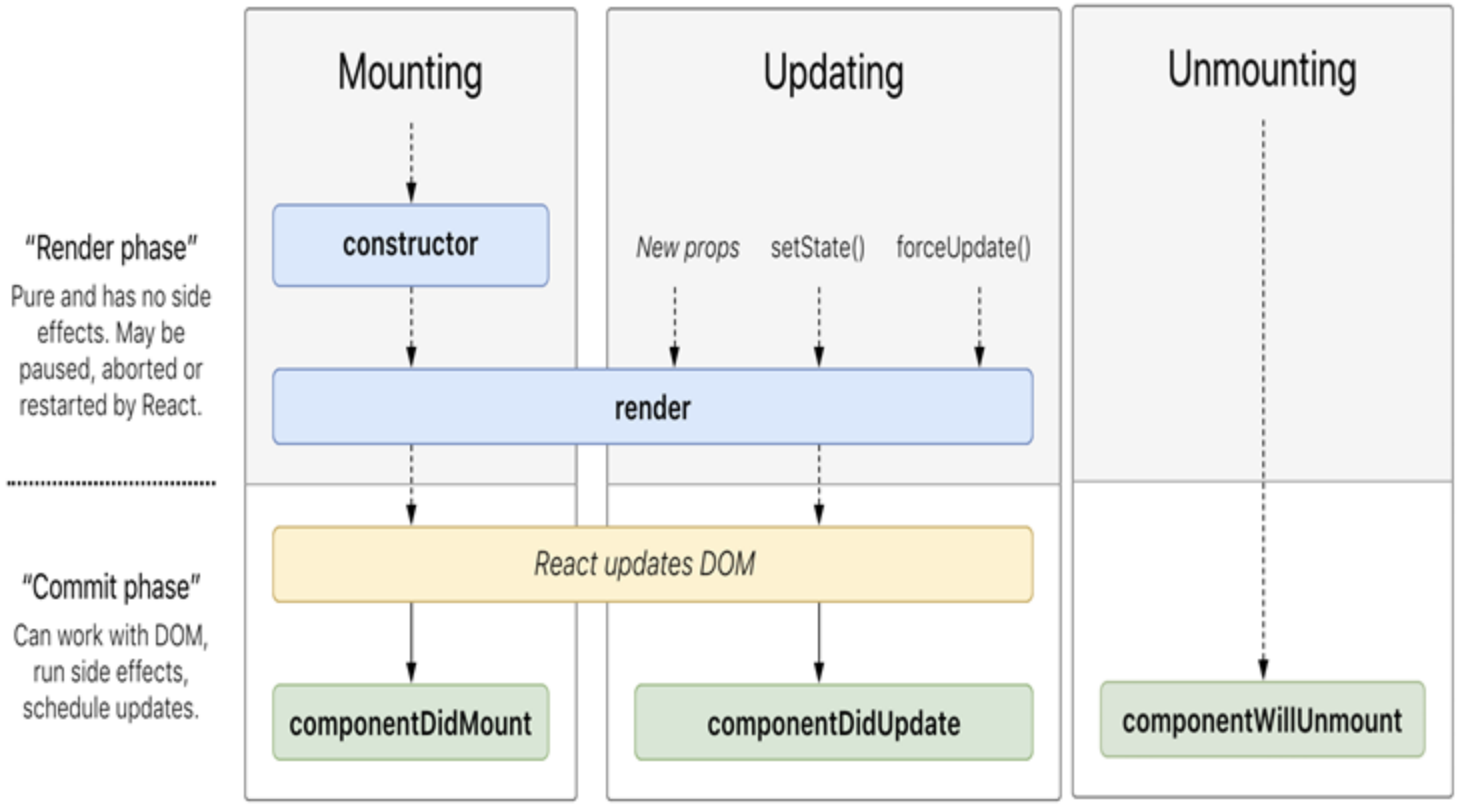
- To ensure that the this.state works well in the timerTick method, add the following line to call that function with the context of the Component

```
constructor(props, context) {
    …
    this.timerTick = this.timerTick.bind(this);
}
```

Component States and LifeCycle methods

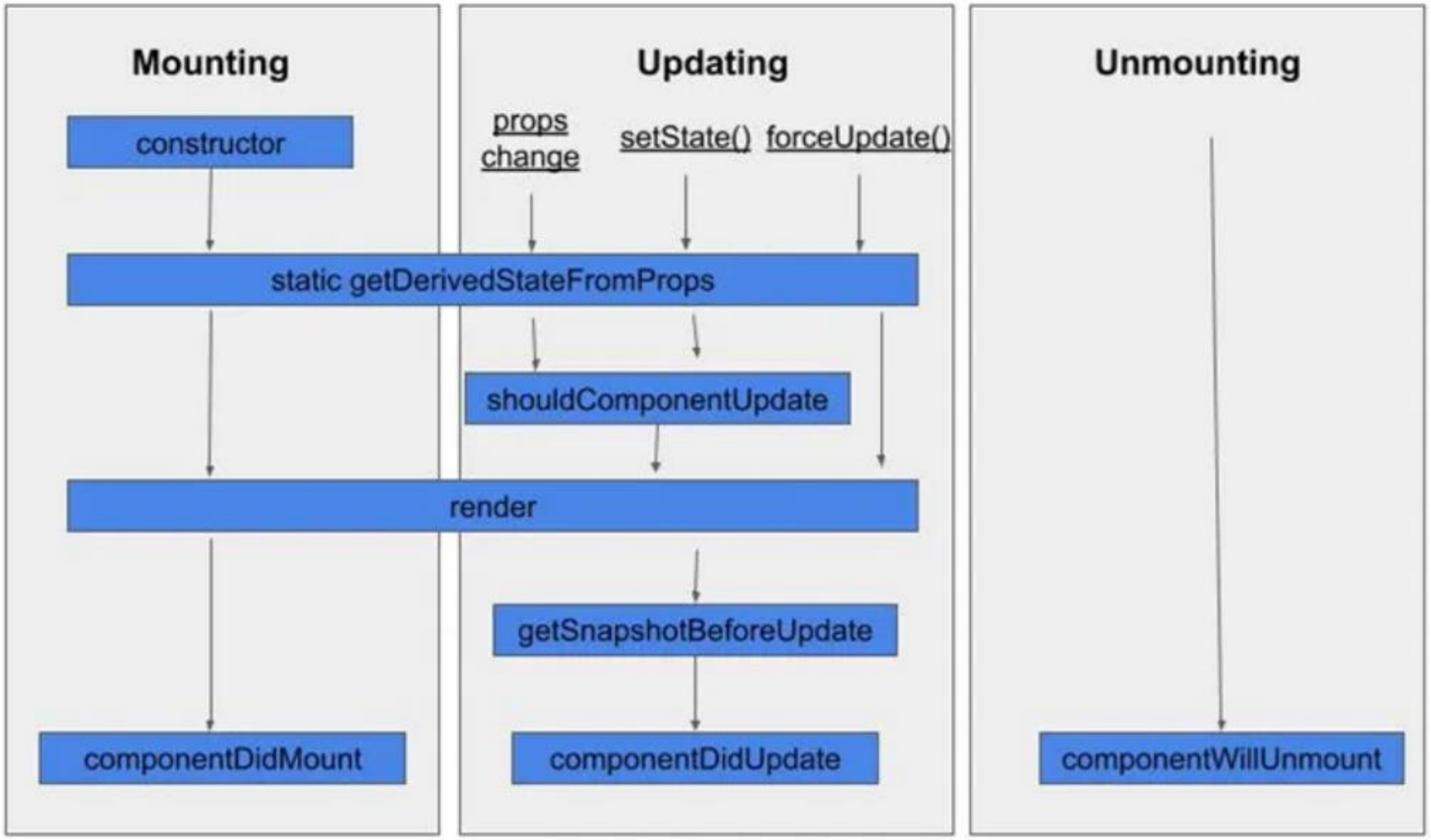
Introduction to Life Cycle

- The series of events that happen from the starting of a React component to its ending.
- Every component in React should go through the following lifecycle of events.
  - **Mounting** - Birth of the Component
  - **Updating**- Growing of component
  - **Unmounting**- End of the component



Component States and LifeCycle methods

Pictorial Representation of methods



Component States and LifeCycle methods

Methods

- **constructor()**
  - Called before the component is mounted.
  - Implementation requires calling of super(props) before further moving. Otherwise, this.props will be undefined in the constructor, which can lead to a major error in the application.
  - Supports Initializing the state and Binding our component
- **render()**
  - Most useful life cycle method as it is the only method that is required
  - Handles the rendering of component while accessing ***this.state*** and **this.props**

Component States and LifeCycle methods

Methods continued..

- **componentDidMount()**
  - The best place to initiate API calls in order to fetch data from remote servers
  - Use setState which will cause another rendering but It will happen before the browser updates the UI. This is to ensure that the user won't see the intermediate state
  - AJAX requests and DOM or state updates should occur here
  - Also used for integration with other JavaScript frameworks like Node.js and any functions with late execution such as **setTimeout** or **setInterva**l

Component States and LifeCycle methods

Methods continued..

getDerivedStateFromProps()

- This method is invoked right before rendering, both during the initial mount and when the component is re-rendered due to changes in props.
- It was used to compare the incoming props (nextProps) with the current props (this.props) and make state updates or logic adjustments accordingly before the next render.

- **shouldComponentUpdate()**
  - allows you to control whether a component should re-render when there are changes to its props or state.
  - Allows a component to exit the Update life cycle if there's no reason to use a replacement render.
  - It returns either true (the component will re-render) or false (the component will not re-render).

Component States and LifeCycle methods

Methods continued..

- **getSnapshotBeforeUpdate()-** for capturing information from the DOM before re-rendering. To capture some properties (a "snapshot") from the DOM before updates occur, which you can then pass into componentDidUpdate() to make adjustments after the update

- **componentDidUpdate()**
  - Is invoked immediately after updating occurs. Not called for the initial render.
  - Will not be invoked if **shouldcomponentUpdate() returns false**.
- **componentWillUnmount()**
  - Called when a component is being removed from the DOM

Component States and LifeCycle methods

Key Methods in the React Component Lifecycle: From Mounting to Unmounting

PES UNIVERSITY

Mounting (Birth of the Component): This phase occurs when a component is created and inserted into the DOM for the first time. The key lifecycle methods or hooks that run during this phase include:

- **constructor():** Initializes the component's state.
- **render():** Returns the JSX to display.
- **componentDidMount()** (class components) / **useEffect(() => {}, [])** (functional components): Executed after the component is mounted, typically used for API calls or subscriptions.

Component States and LifeCycle methods

Key Methods in the React Component Lifecycle: From Mounting to Unmounting

Updating (Growth of the Component): This happens when the component's state or props change, causing it to re-render. The key methods/hooks during this phase include:

- **shouldComponentUpdate()** (class components): Determines whether the component should re-render.
- **render():** Re-renders the JSX when state or props change.
- **componentDidUpdate()** (class components) / **useEffect()** (functional components): Runs after the component has updated.

Component States and LifeCycle methods

Key Methods in the React Component Lifecycle: From Mounting to Unmounting

PES UNIVERSITY

Unmounting (End of the Component): This phase happens when the component is removed from the DOM. The key lifecycle method is:

- **componentWillUnmount()** (class components) / Cleanup function in **useEffect()** (functional components): Used for cleanup tasks like cancelling API calls or removing event listeners.

Component States and LifeCycle methods

Demo of diff lifecycle methods

PES UNIVERSITY

THANK YOU

Vinay Joshi and Dr.Sarasvathi V

Department of Computer Science and Engineering

vinayj@pes.edu

sarsvathiv@pes.edu

Acknowledgement

The slides are created from various internet resources with valuable contributions from multiple professors and teaching assistants in the university.