REACT NATIVE

Component Lifecycle

“Render Phase”

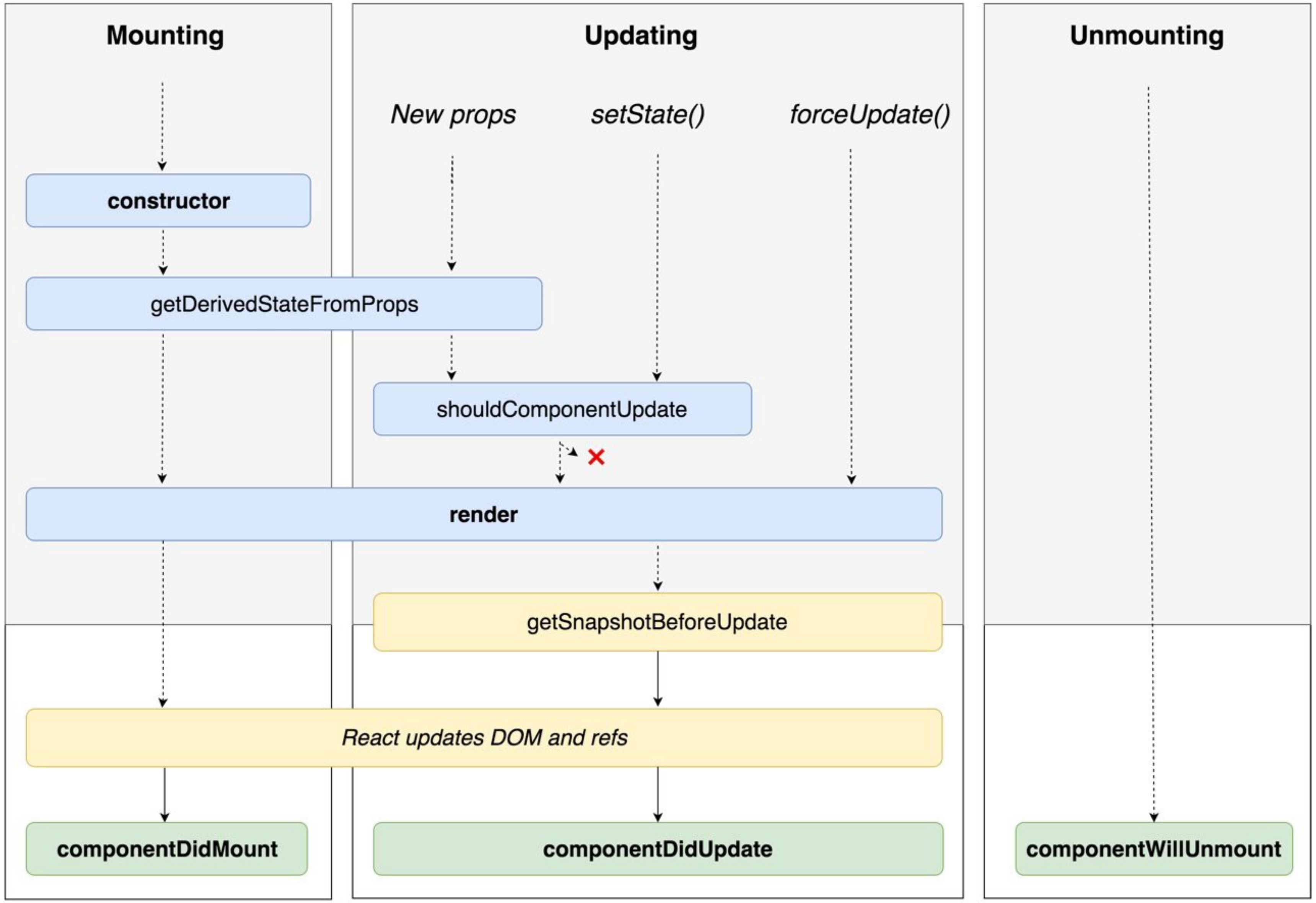
Pure and has no side effects. May be paused, aborted or restarted by React.

“Pre-Commit Phase”

Can read the DOM.

“Commit Phase”

Can work with DOM, run side effects, schedule updates.

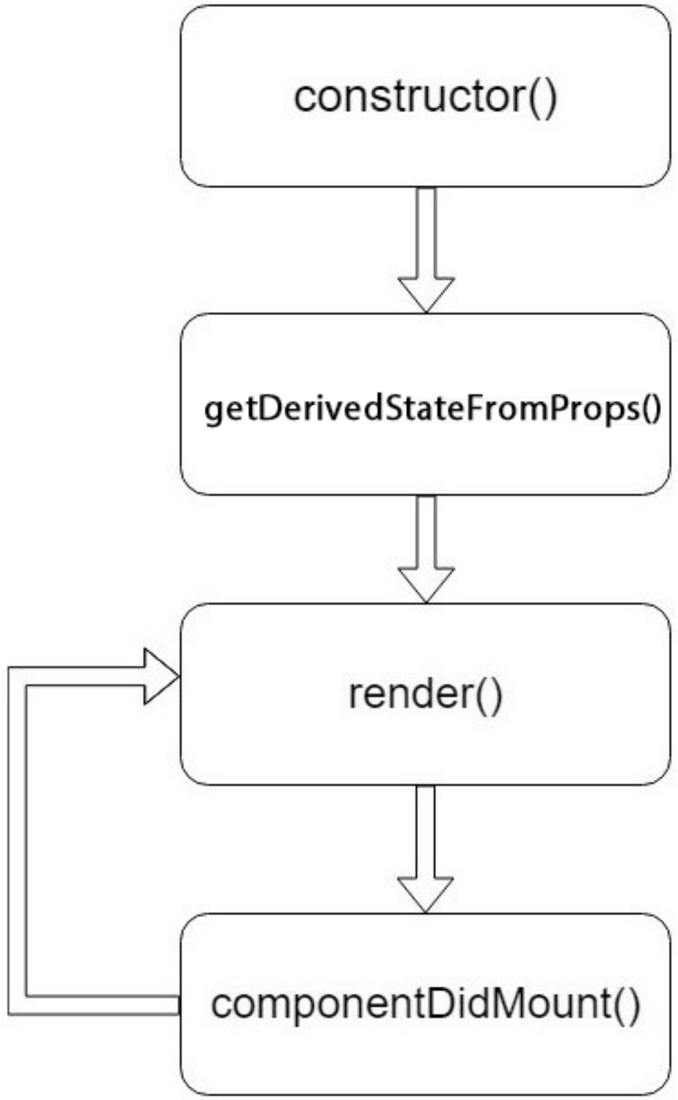


React Component Lifecycle

- Lifecycle method can be grouped into 3 groups, corresponding to 4 stages of the component
  - Mounting
  - Updating
  - Unmounting
  - Error Handling

Mounting

- It will be in the following order
  1. constructor()
  2. static getDerivedStateFromProps()
  3. render()
  4. componentDidMount()



constructor(props)

- This method create a component, if not initializing the state or binding methods, does not need to declare this method

```
export default class Clicker extends Component {
    constructor(props) {
        super(props);
        this.handleClick = this.handleClick.bind(this);
        this.state = {
            clicks: 0
        };
    }

    handleClick() {
        this.setState({
            clicks: this.state.clicks + 1
        })
    }
    //...
}
```

constructor(props)

- Don’t transfer props to state! Handling logic will be very complicated later

```
constructor(props) {
    super(props);

    // DON'T DO THIS
    this.state = { color: props.color };
}
```

static getDerivedStateFromProps(props, state)

- This method is invoked right before calling the render method, both on the initial mount and on subsequent updates.
- It should return an object to update the state, or null to update nothing
- This method exists for only one purpose. It enables a component to update its internal state as the result of **changes in props**

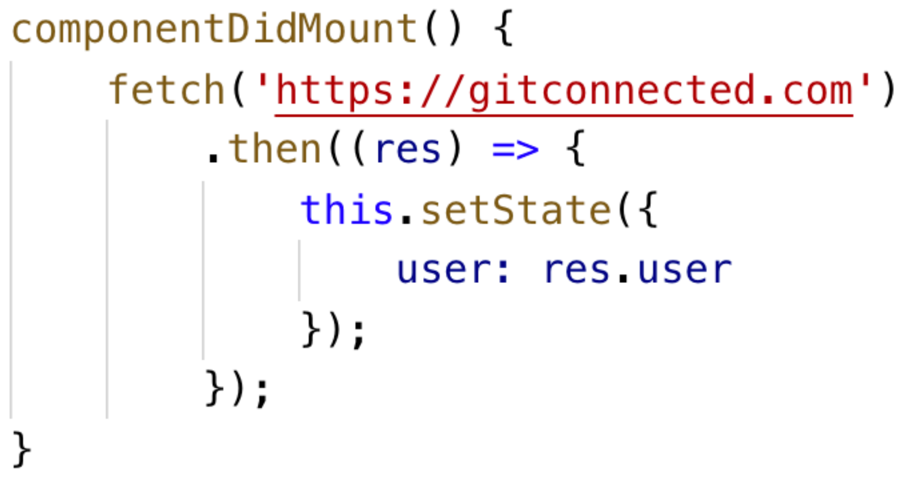
```
static getDerivedStateFromProps(props, state) {
    // Any time the current user changes,
    // Reset any parts of state that are tied to that user.
    if (props.userID !== state.prevPropsUserID) {
        return {
            email: props.defaultEmail,
            prevPropsUserID: props.userID
        };
    }
    return null;
}
```

render()

- This is the only required method when creating a component, which requires return one of the values below:
  - React element
  - Arrays and fragments
  - Portals
  - String and Numbers
  - Booleans or null
- This method will not be called if **shouldComponentUpdate()** return false

componentDidMount()

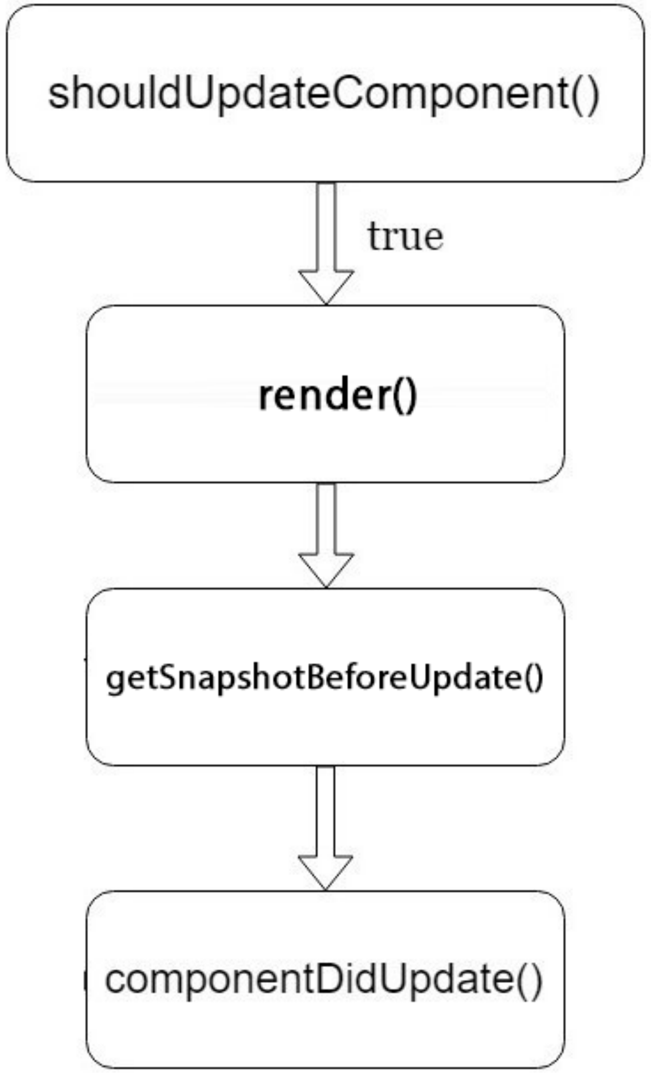
- Component has been rendered, it’s time to call AJAX or setState



```
componentDidMount() {
    fetch('https://gitconnected.com')
        .then((res) => {
            this.setState({
                user: res.user
            });
        });
}
```

Updating

- These methods will be called when there is a change of state or props
    1. static getDerivedStateFromProps()
    2. shouldComponentUpdate()
    3. render()
    4. getSnapshotBeforeUpdate()
    5. componentDidUpdate()



shouldComponentUpdate(nextProps, nextState)

- Improve performance of React
- Is invoked before rendering when new props or state are being received
- Default value is true
- Not called for the initial render or when **forceUpdate()** is used

```
shouldComponentUpdate(nextProps, nextState) {
    return this.props.clicks !== nextProps.clicks;
}
```

getSnapshotBeforeUpdate()

- Is invoked right before rendered output is committed to the DOM
- It enables component to capture some information from the DOM (Ex: scroll position) before it is potentially changed
- Values return from this function will be passed as a parameter to **componentDidUpdate()**

componentDidUpdate(prevProps, prevState, snapshot)

- Is invoked immediately after updating occurs
- This method is not called for the initial render
- If call setState in this function, the conditional sentence must be included, otherwise it will be repeated infinitely
- If the method **getSnapshotBeforeUpdate()** is implemented, the return value will be include in snapshot parameter, otherwise undefined
- This function will not be called if **shouldComponentUpdate()** return false

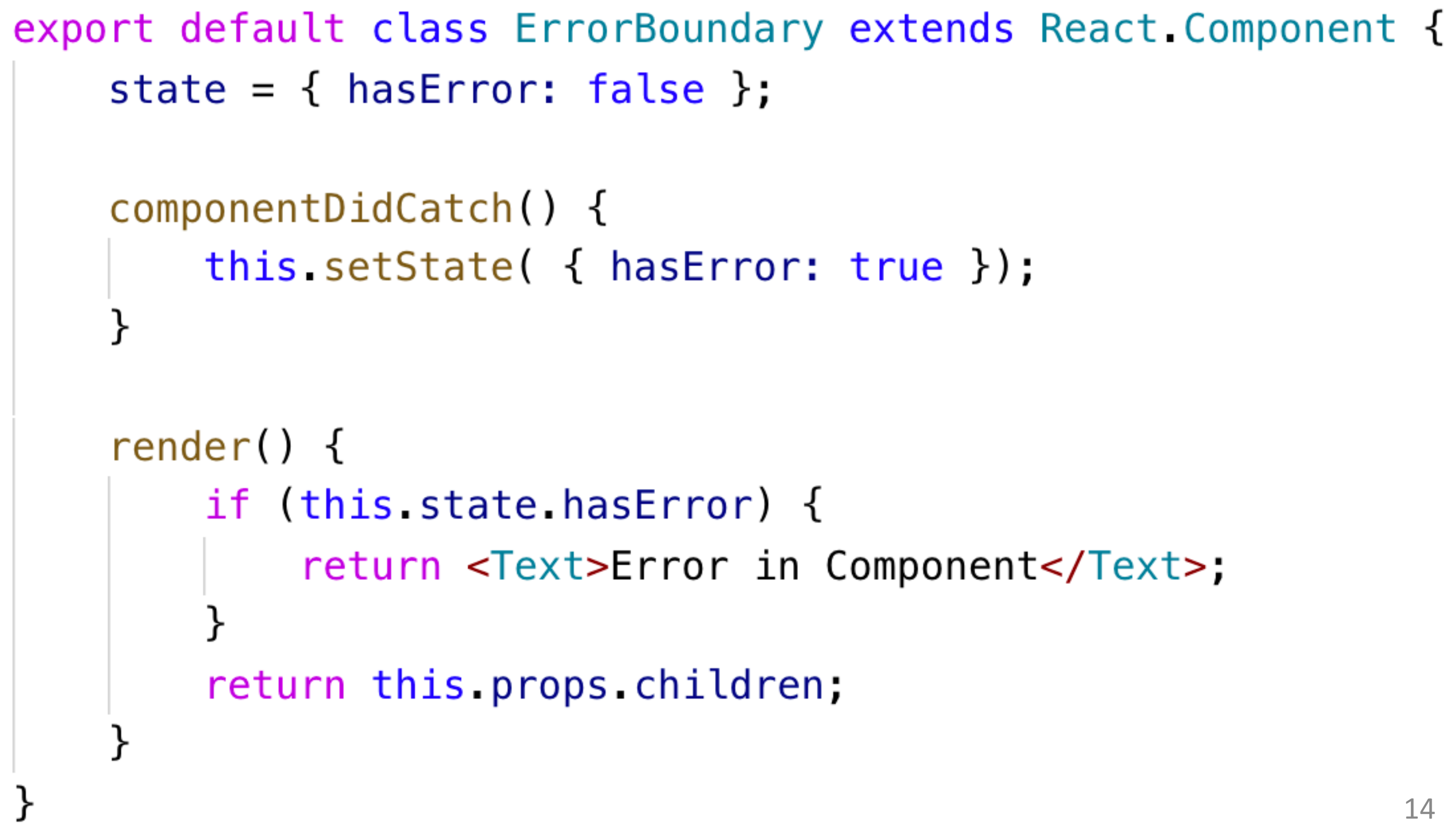
Unmounting

- The method is called before removing the component from DOM
  - componentWillUnmout()
- This method can be use to remove listener, setInterval functions or cancel network request

```
componentWillUnmount() {
    window.removeEventListener('resize', this.resizeEventHandler);
}
```

Error Handling

- Regardless of where the error is in component, it will call this method
  - componentDidCatch()
- This function will handle error when a component fails, and it will show the error on UI



```
export default class ErrorBoundary extends React.Component {
    state = { hasError: false };

    componentDidCatch() {
        this.setState( { hasError: true });
    }

    render() {
        if (this.state.hasError) {
            return <Text>Error in Component</Text>;
        }
        return this.props.children;
    }
}
```

Thank you!