React Lifecycle & Synthetic Event

by Kulinskyi Vitalii

softserve

AGENDA

- Lifecycle
  - Lifecycle of Components
  - Mounting
  - Updating
  - Unmounting
- SyntheticEvent
  - React Events
  - Events Handler
  - Supported Events

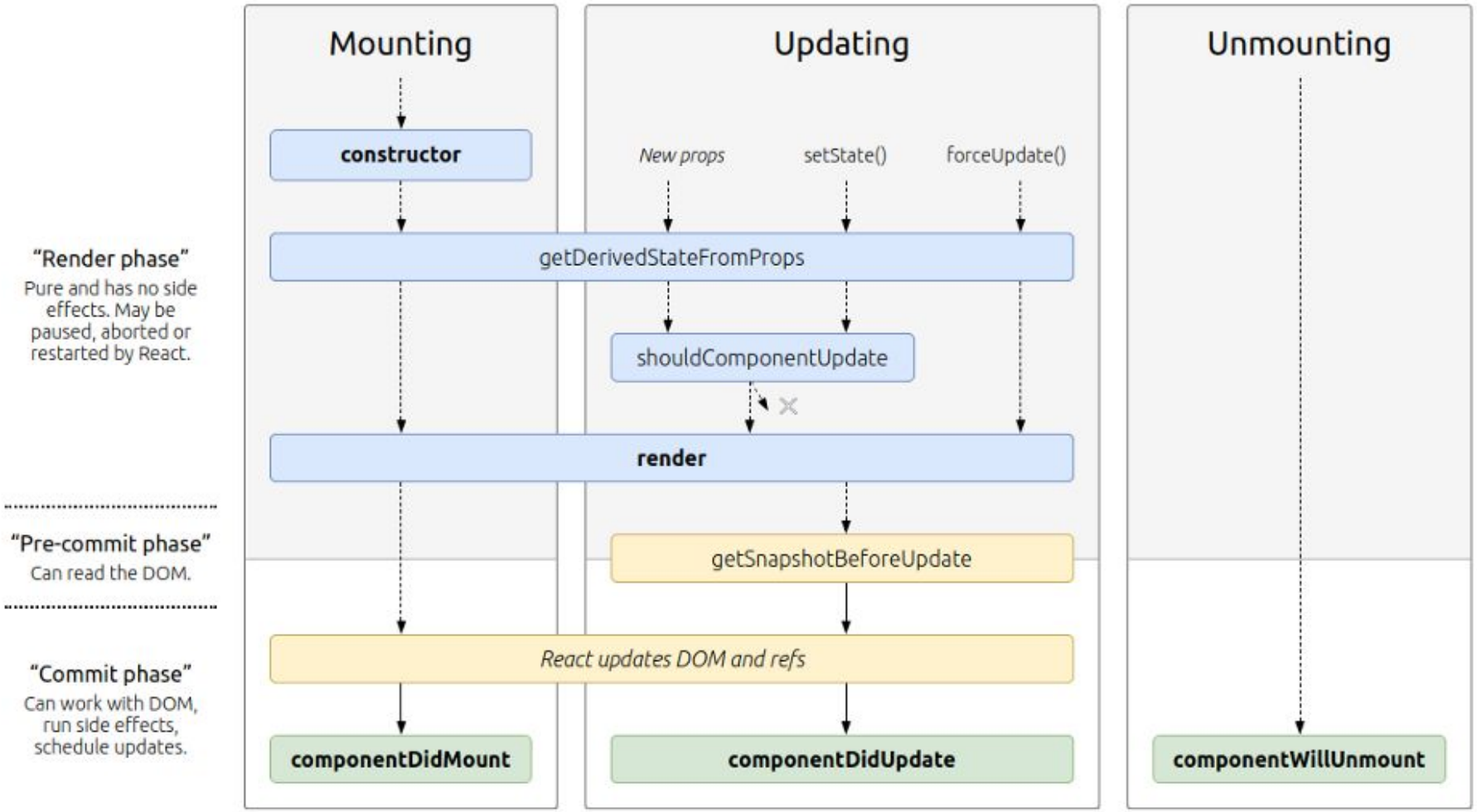
softserve

Lifecycle of Components

Each component in React has a lifecycle which you can monitor and manipulate during its three main phases:

- Mounting
- Update
- Unmounting

Lifecycle of Components



softserve

Mounting

- After preparing with basic needs, state and props, our React Component is ready to mount in the browser DOM.
- This phase gives hook methods for before and after mounting of components.

React has four built-in methods that gets called, in this order, when mounting a component:

- constructor( )
- getDerivedStateFromProps( )
- render( )
- componentDidMount( )

The render( ) method is required and will always be called, the others are optional and will be called if you define them.

constructor( )

- Called before anything else, when the component is initiated, and it is the natural place to set up the initial state and other initial values.
- Called with the props, as arguments, and you should always start by calling the super(props) before anything else

```
class MyComponent extends Component {
    constructor(props) {
      super(props);
      this.state = {
        counter: 0,
      };
    }
  }
```

softserve

getDerivedStateFromProps( )

- Called right before rendering the element(s) in the DOM.
- This is the natural place to set the state object based on the initial props.
- It takes state as an argument, and returns an object with changes to the state.

```
static getDerivedStateFromProps(props, state) {
    return { blocks: createBlocks(props.numberOfBlocks) };
  }
```

softserve

render( )

- The render() method is required, and is the method that actually outputs the HTML to the DOM.

```
class Header extends React.Component {
  render() {
    return (
      <h1>This is the content of the Header component</h1>
    );
  }
}

ReactDOM.render(<Header />, document.getElementById('root'));
```

componentDidMount( )

- This is the hook method which is executed after the component did mount on the DOM.
- This method is executed once in a lifecycle of a component and after the first render.
- As, in this method, we can access the DOM

Usage: this is the right method to integrate API

Updating

- A component is updated whenever there is a change in the component's state or props.
- React has five built-in methods that gets called, in this order, when a component is updated:

  - getDerivedStateFromProps()
  - shouldComponentUpdate()
  - render()
  - getSnapshotBeforeUpdate()
  - componentDidUpdate()

The render() method is required and will always be called, the others are optional and will be called if you define them.

Updating

- getDerivedStateFromProps( ) - first method that is called when a component gets updated
- render( ) - called when a component gets updated, it has to re-render the HTML to the DOM, with the new changes.

shouldComponentUpdate()

- This method tells the React that when the component receives new props or state is being updated, should React re-render or it can skip rendering?
- Method you can return a Boolean value that specifies whether React should continue with the rendering or not.
- The default value is true.

```
shouldComponentUpdate(nextProps, nextState) {
    // Only update if bricks change
    return nextState.blocks.length > this.state.blocks.length;
  }
```

softserve

getSnapshotBeforeUpdate()

```
class ScrollingList extends React.Component {
  constructor(props) {
    super(props);
    this.listRef = React.createRef();
  }

  getSnapshotBeforeUpdate(prevProps, prevState) {
    // Are we adding new items to the list?
    // Capture the scroll position so we can adjust scroll later.
    if (prevProps.list.length < this.props.list.length) {
      const list = this.listRef.current;
      return list.scrollHeight - list.scrollTop;
    }
    return null;
  }

  componentDidUpdate(prevProps, prevState, snapshot) {
    // If we have a snapshot value, we've just added new items.
    // Adjust scroll so these new items don't push the old ones out of view.
    // (snapshot here is the value returned from getSnapshotBeforeUpdate)
    if (snapshot !== null) {
      const list = this.listRef.current;
      list.scrollTop = list.scrollHeight - snapshot;
    }
  }

  render() {
    return (
      <div ref={this.listRef}>{/* ...contents... */}</div>
    );
  }
}
```

- In the method you have access to the props and state before the update, meaning that even after the update, you can check what the values were before the update.
- If the method is present, you should also include the componentDidUpdate() method, otherwise you will get an error.

softserve

componentDidUpdate()

- Method is called after the component is updated in the DOM
- Method is not called for the initial render.

```
componentDidUpdate(prevProps) {
  // Typical usage (don't forget to compare props):
  if (this.props.userID !== prevProps.userID) {
    this.fetchData(this.props.userID);
  }
}
```

Unmounting

- In this phase, the component is not needed and the component will get unmounted from the DOM.
- The method which is called in this phase :
  - componentWillUnmount()

componentWillUnmount( )

- This method is the last method in the lifecycle.
- This is executed just before the component gets removed from the DOM.

Usage: In this method, we do all the cleanups related to the component.

For example, on logout, the user details and all the auth tokens can be cleared before unmounting the main component.


TO
THE
FUTURE
softserve