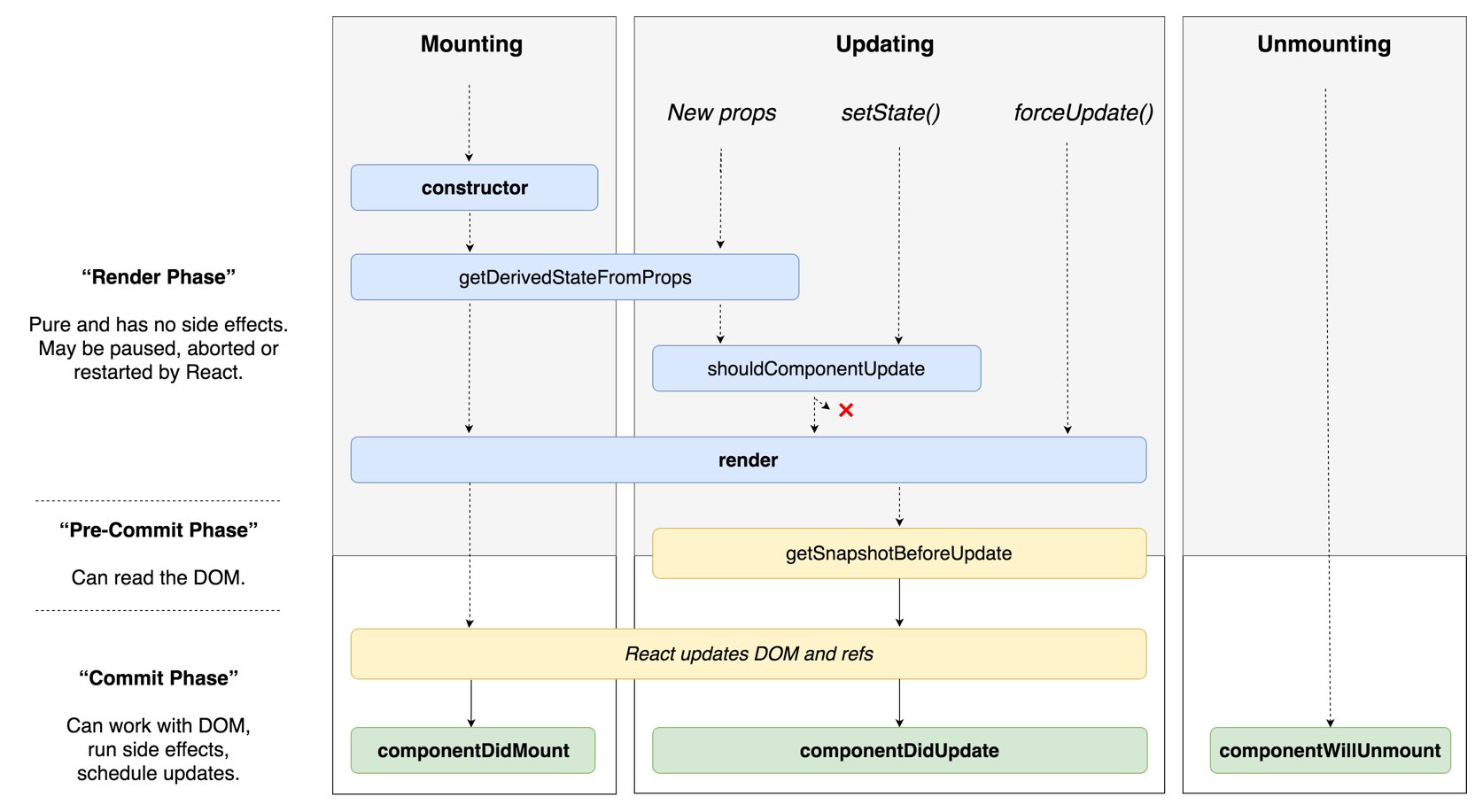
**React Lifecycle Methods:**

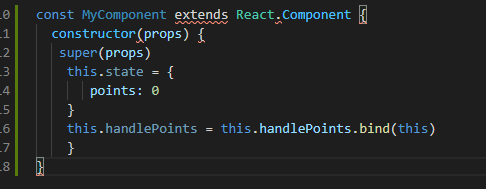
Each component in React has a lifecycle which we can monitor and manipulate during its three main phases. The three phases are: Mounting, Updating, and Unmounting.



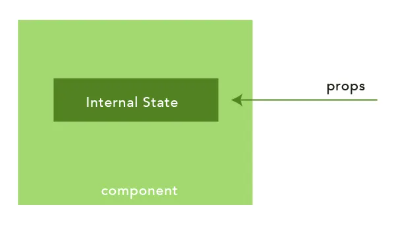
**The Mounting lifecycle methods:** The mounting phase refers to the phase from when a component is created and inserted to the DOM.

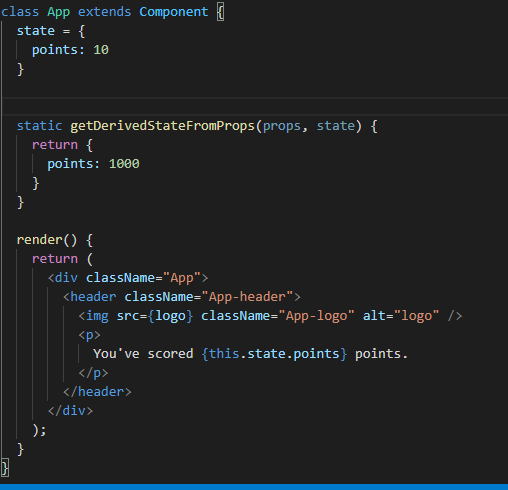
The following methods are called (in order):

* **constructor ():** This is the very first method called as the component is “brought to life”. The constructor method is called before the component is mounted to the DOM. Usually, we would initialize state and bind event handlers’ methods within the constructor method.

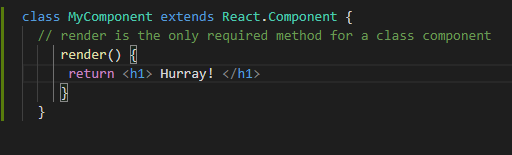


* **getDerivedStateFromProps():**  This method takes in “props” and “state.” And we can either return an object to update the state of the component return null to make no updates. Essentially, this method allows a component to update its internal state in response to a change in props.





* **render():** After the getDerivedStateFromProps methond is called, the next lifecycle method is line is the render method. The render() function should be pure, meaning that it does not modify component state, it returns the same result each time it’s invoked, and it does not directly interact with the browser.



* **componentDidMount():** After render is called, the component is mounted to the DOM, and the componentDidMount method is invoked. This function is invoked immediately after the component is mounted to the DOM. Sometimes we need to grab a DOM node from the component tree immediately after it is mounted. This is the right component lifecycle method to do this.

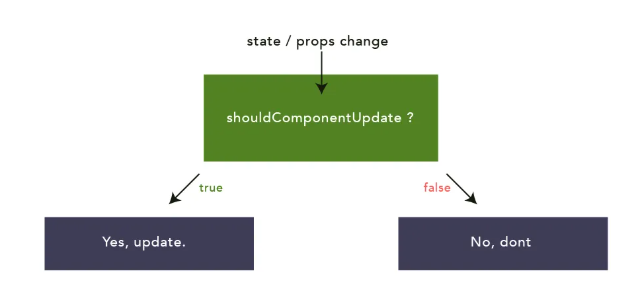


**The updating lifecycle methods:** A component is updated whenever there is a change in the component's state or props. React has five built-in methods that get called, in this order, when a component is updated

* **getDerivedStateFromProps:** During updates, the getDerivedStateProps method is called. What is important to note is that this method is invoked in both the mounting and updating phases. The same method.



* **shouldComponentUpdate:** Within this lifecycle method, we can return a Boolean -true or false and control whether the component gets re-rendered or not i.e. upon a change in state or props.



The example below shows what happens when the shouldComponentUpdate () method returns false. It stops the component from re-rendering during the update phase.



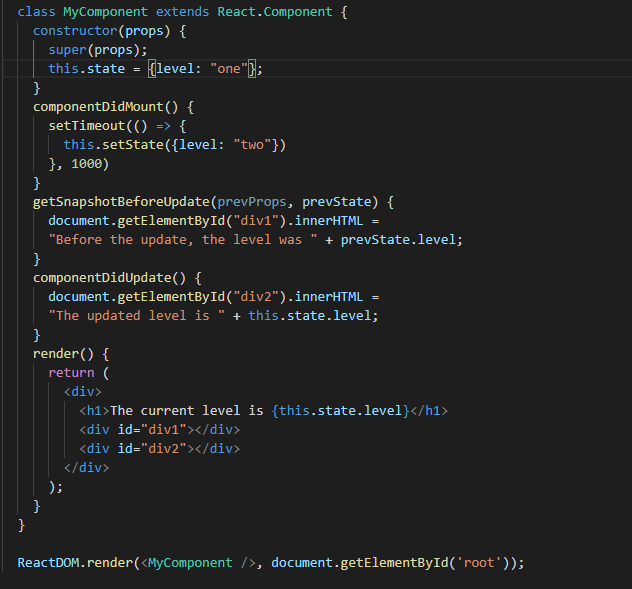
Now we’ll run the same example as above, but this time the shouldComponentUpdate () method returns true instead. It does not stop the component from rendering at each update.



* **render():** The render() method is of course called when a component gets updated, it has to re-render the HTML to the DOM, with the new changes.



* **getSnapshotBeforeUpdate():** getSnapshotBeforeUpdate() is invoked right before the most recently rendered output is committed to e.g. the DOM. It enables our component to capture some information from the DOM (e.g. scroll position) before it is potentially changed. Any value returned by this lifecycle will be passed as a parameter to componentDidUpdate (). This use case is not common, but it may occur in UIs like a chat thread that need to handle scroll position in a special way. A snapshot value (or null) should be returned.



In the above example, when the component is mounting, it is rendered with the level “one”. When the component has been mounted, a timer changes the state, and after one second, the level becomes “two”. This action triggers the update phase, and since this component has a getSnapshotBeforeUpdate () method, this method is executed and returns the string. Then the componentDidUpdate () method is executed and writes a message in the empty div1 element.

* **componentDidUpdate():** componentDidUpdate() is invoked immediately after updating occurs. This method is not called for the initial render. We may call setState () immediately in componentDidUpdate (). Typically, in componentDidUpdate, we wrap the logical in a conditional so that you do not execute it on every state or props change.



In the above example when the component is mounting it is rendered with the level “one”. When the component has been mounted, a timer changes the state, and after one second, the level becomes “two”. This action triggers the update phase, and then the componentDidUpdate () method is executed and writes a message in the empty mydiv element.

**The Unmounting lifecycle method:** The next phase in the lifecycle is when a component is removed from the DOM or unmounting as it is named in React. React has only one built-in method that gets called when a component is unmounted. The following method is invoked during the component unmounting phase.

* **componentWillUnmount():** componentWillUnmount() is invoked immediately before a component is unmounted and destroyed. Perform any necessary cleanup in this method, such as invalidating timers, canceling network requests, or cleaning up any subscriptions that were created in componentDidMount ().

We **should not call**setState () in componentWillUnmount() because the component will never be re-rendered. Once a component instance is unmounted, it will never be mounted again.

