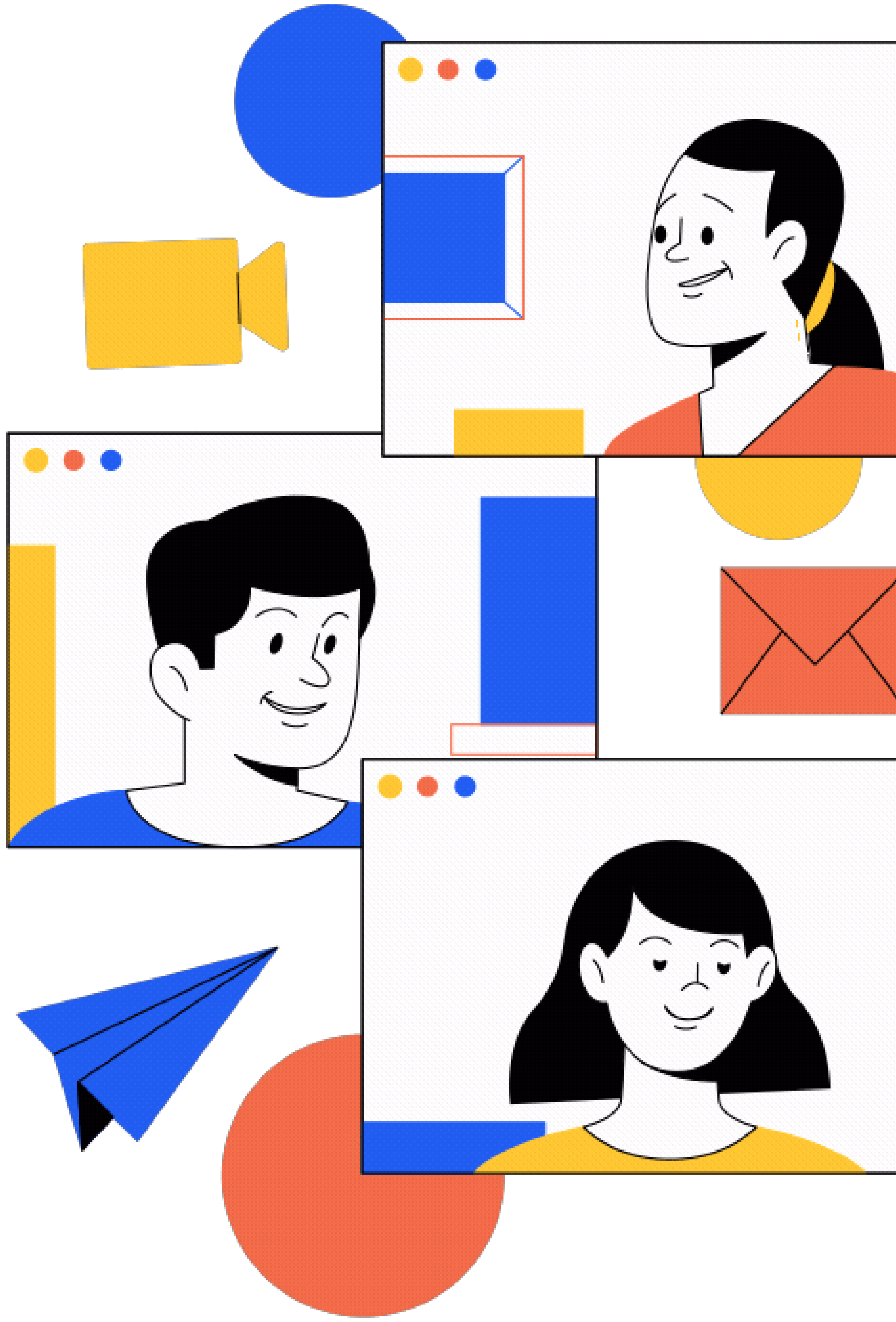
UDACITY

Week 2 - React Dev. Cross-Skilling ND

Are your ready for some state management?



Ahmed Abdelbakey Ghonem

React Session Lead

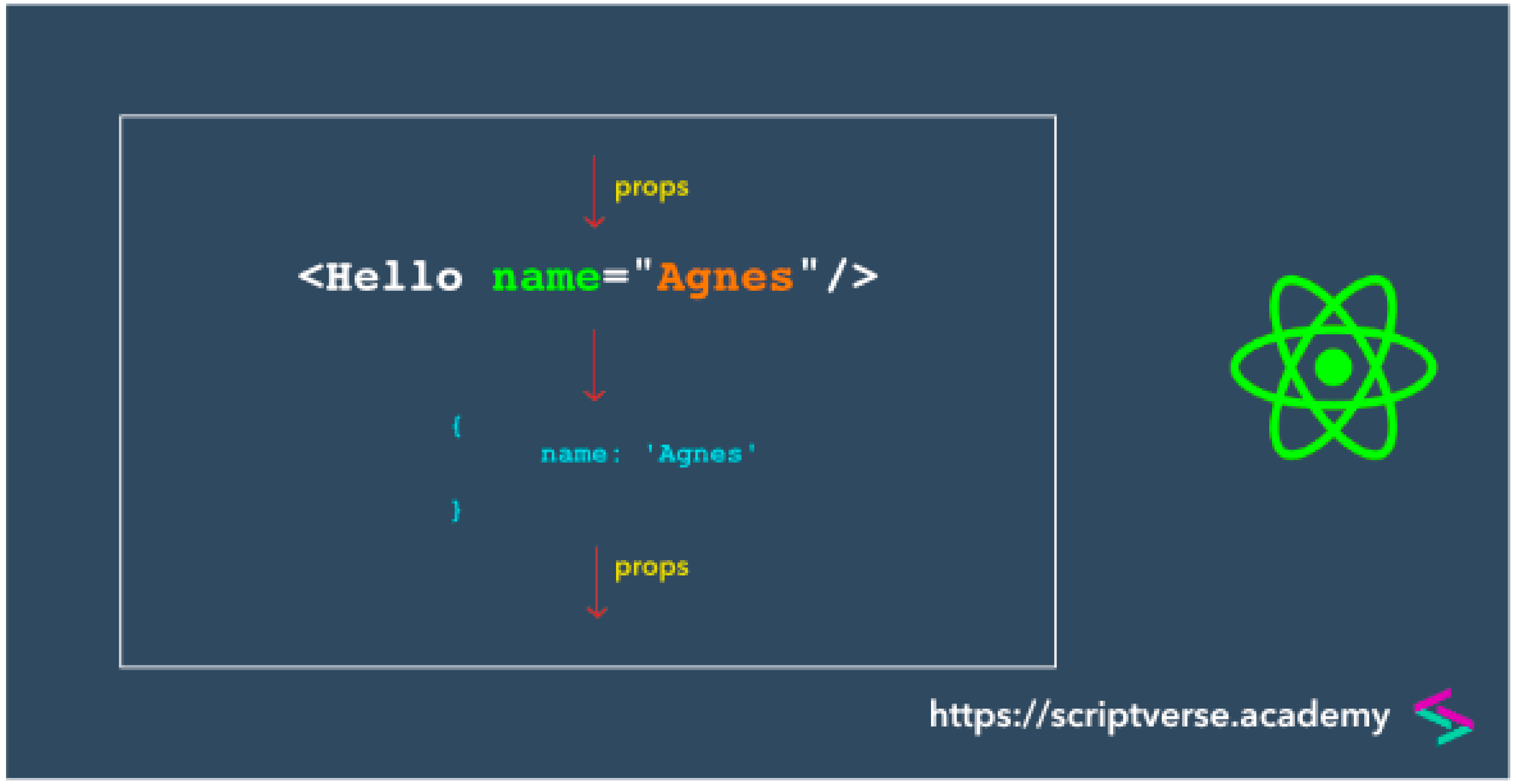
Agenda

What we'll cover in this session

- What are State & Props?
- Difference between State and Props
- Managing State in React
- Component Lifecycle
- Lifecycle Methods
- Live Demo

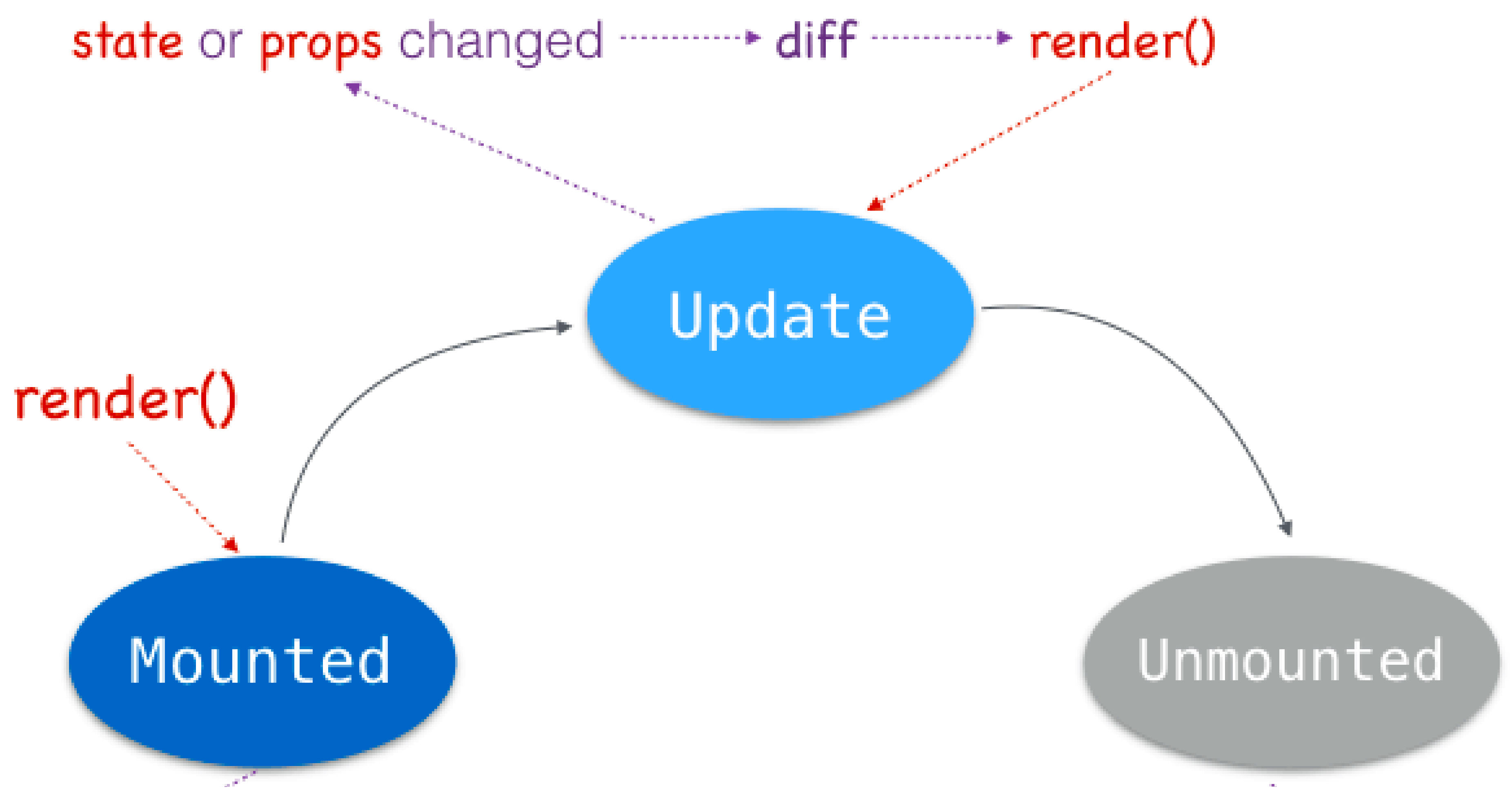
What is **Props?**

- Props is acronym for Properties, They are **read-only** JS objects which must be kept pure and immutable.
- They are passed from parent to child components.



What is a **State**?

- State is **mutable** JS objects that used by react to determine or represent information about the component's current situation.
- If any part of these states change, The component will **re-render** .



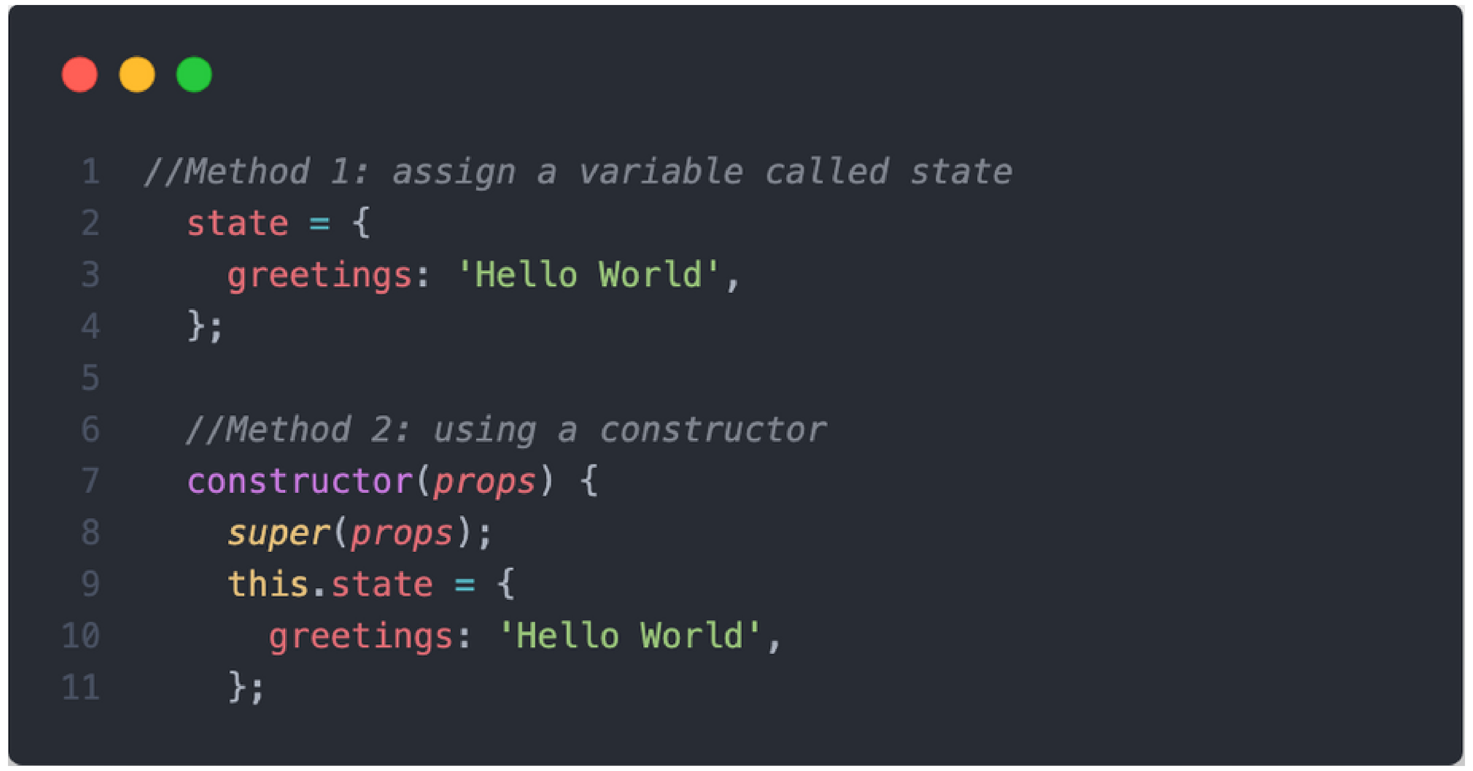
Props

- Props are **read-only** JS objects.
- **Immutable** Objects
- Passed from parent to child.
- If you want to change a prop, You must use a **callback function**.

State

- **Mutable** JS objects.
- State has **methods** to modify its properties.
- **State updates** are asynchronous.
- Usually Parent's state are passed as props to children.

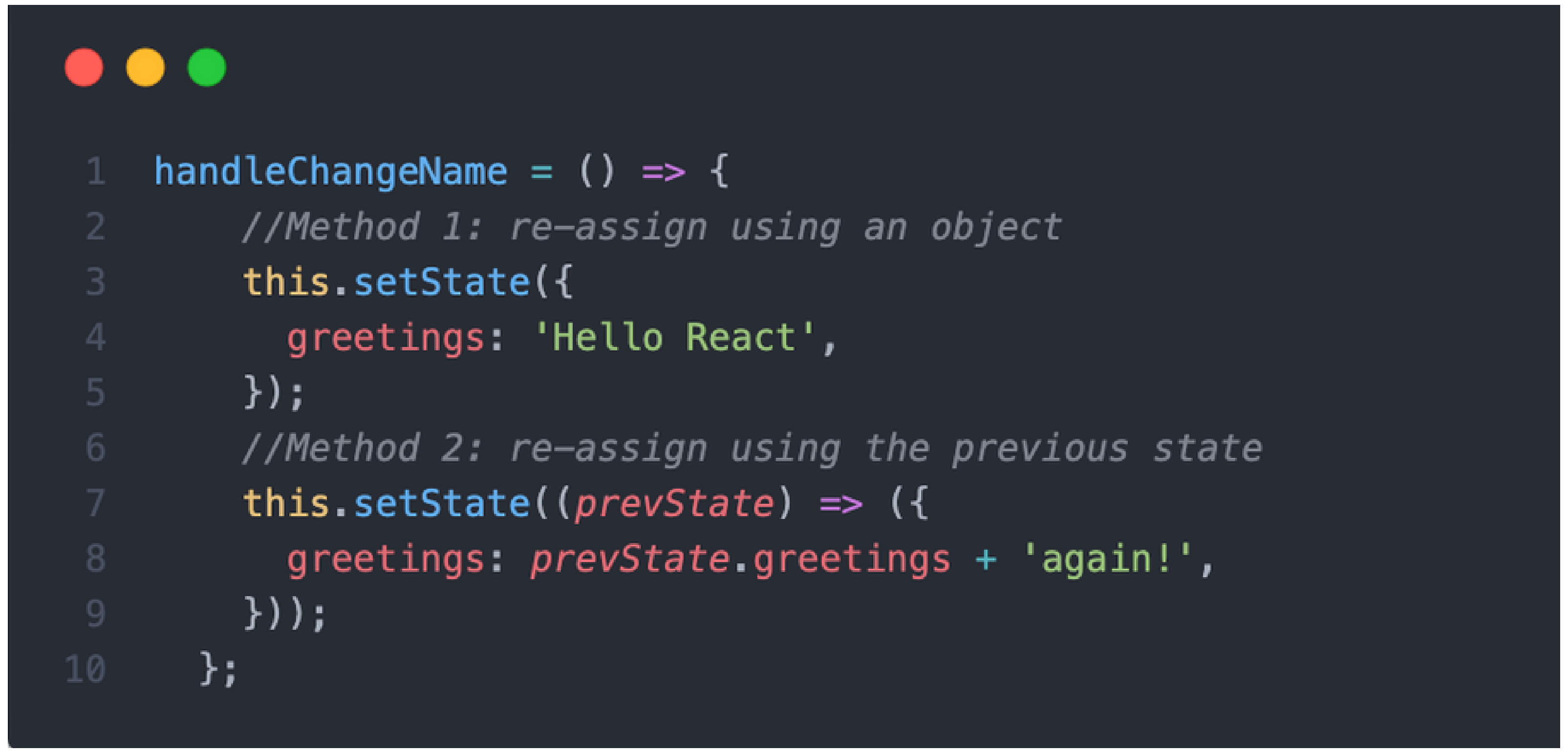
Creating the State



```javascript
//Method 1: assign a variable called state
  state = {
    greetings: 'Hello World',
  };

  //Method 2: using a constructor
  constructor(props) {
    super(props);
    this.state = {
      greetings: 'Hello World',
    };
```

Updating the State

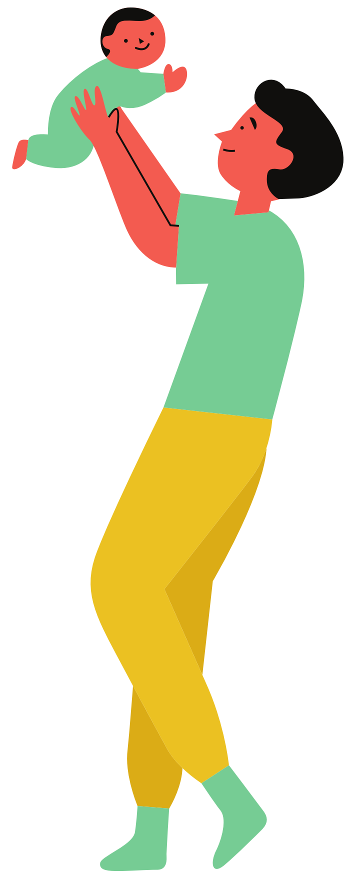


```javascript
handleChangeName = () => {
    //Method 1: re-assign using an object
    this.setState({
      greetings: 'Hello React',
    });
    //Method 2: re-assign using the previous state
    this.setState((prevState) => ({
      greetings: prevState.greetings + 'again!',
    }));
  };
```

Article: Passing a function to setState

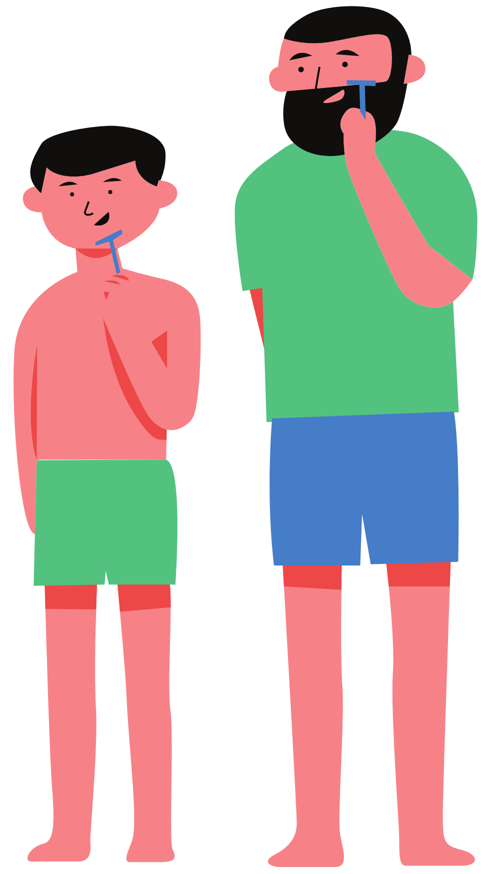
React Component Lifecycle

The react component passes through 3 different phases: Mounting, Updating, and Un-mounting.



1. Mounting

Component is initialized
and added to the DOM



2. Updating

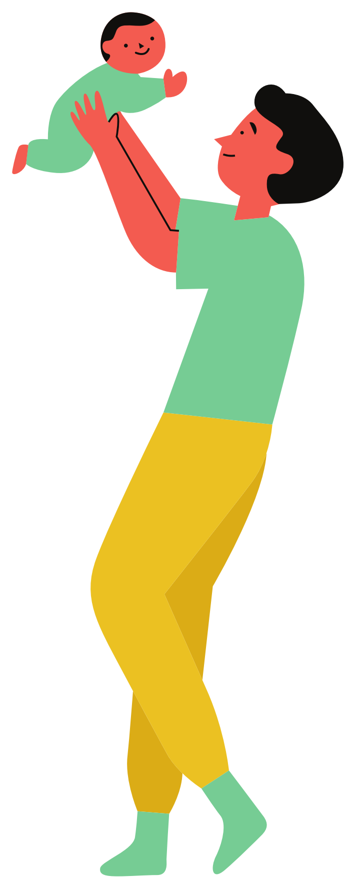
Component is being updated
(state or props change)

3. Unmounting

Component is dead and
removed from the DOM

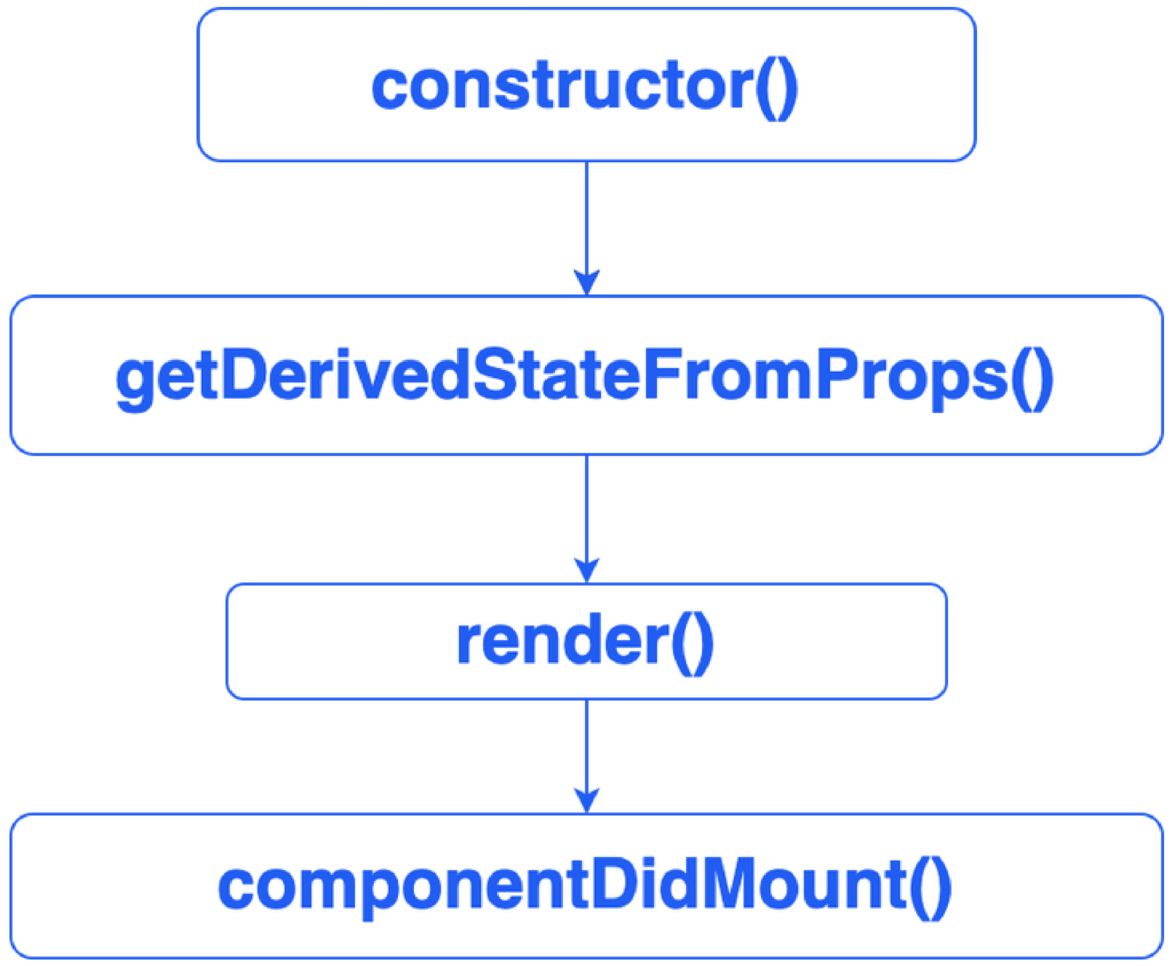
1. Mounting Phase

This phase refers to the component's creation. This is where the component is added to the DOM.
and the mounting phase has 4 different methods

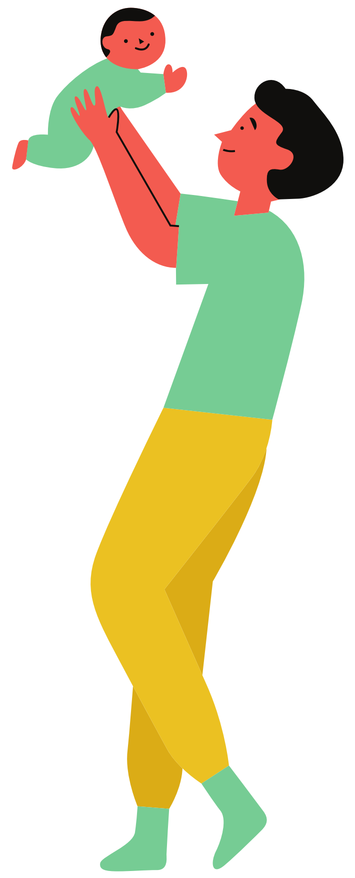


1. Mounting

Component is initialized
and added to the DOM



1. Mounting Phase



1. Mounting

Component is initialized and added to the DOM

1. `constructor()`

Constructor can be used to

1. Initiate the state of the component
2. Binding methods to the current instance of the component

2. `static getDerivedStateFromProps()`

This method is used to update the current state based on changes in the passed props

3. `render()`

The render method is a required method that the component must implement,It is being used to render the actual component content and *its main job is painting the component content into the page*

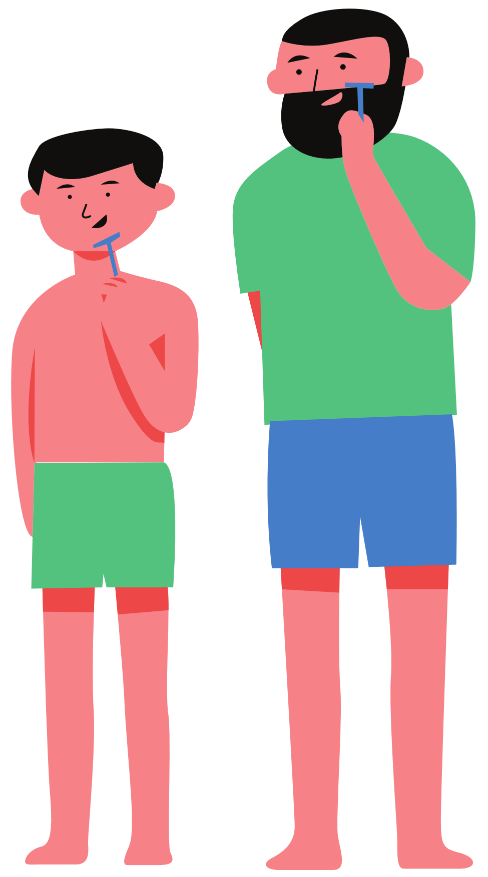
4. `componentDidMount()`

This function is being invoked after the render method and its a perfect place to make API calls and update the state

[Demo: Mounting Phase in Action]

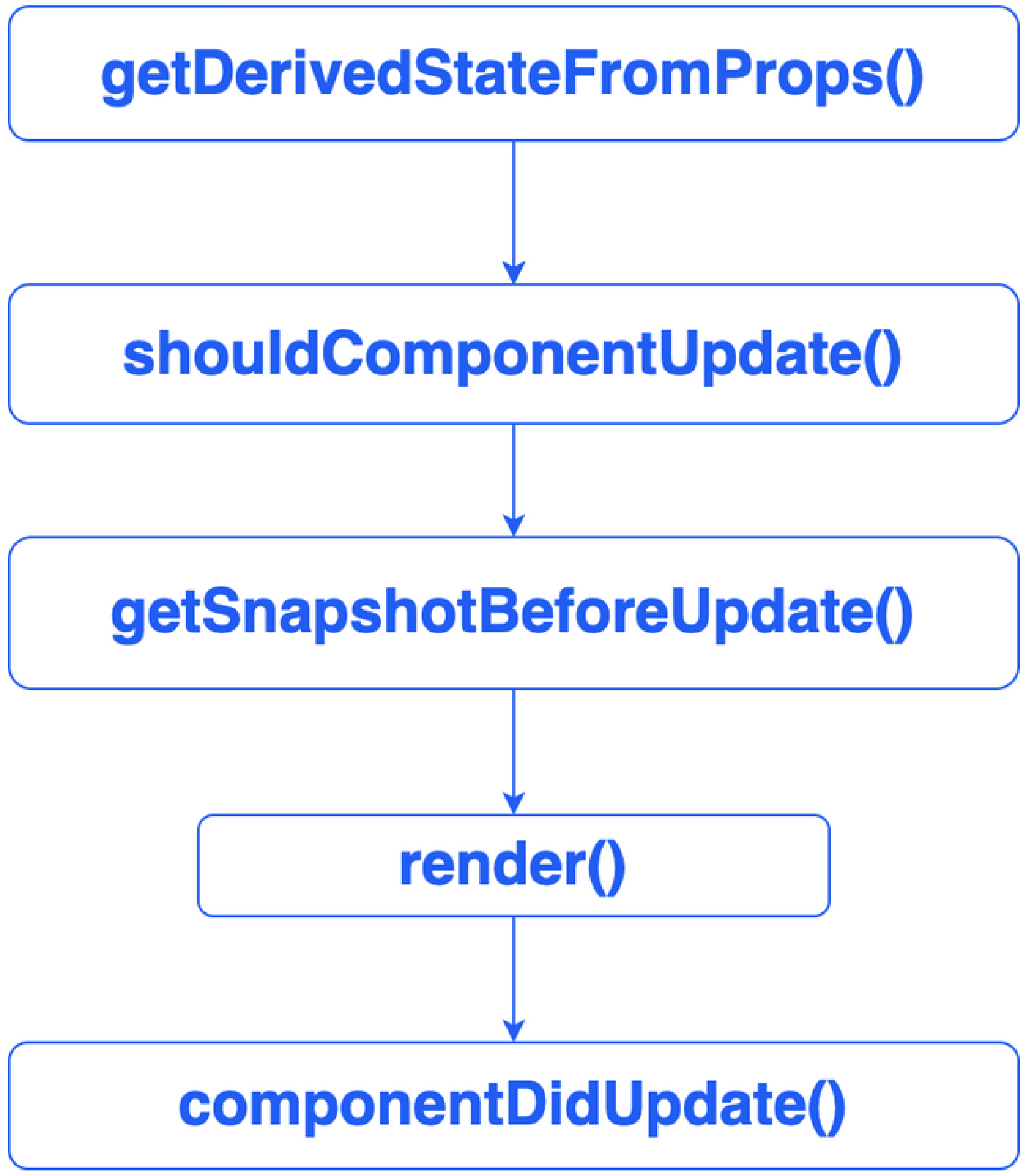
2. Updating Phase

This second phase represents times where a component needs to be updated due to a change in its current state or props, and it has 5 different lifecycles:



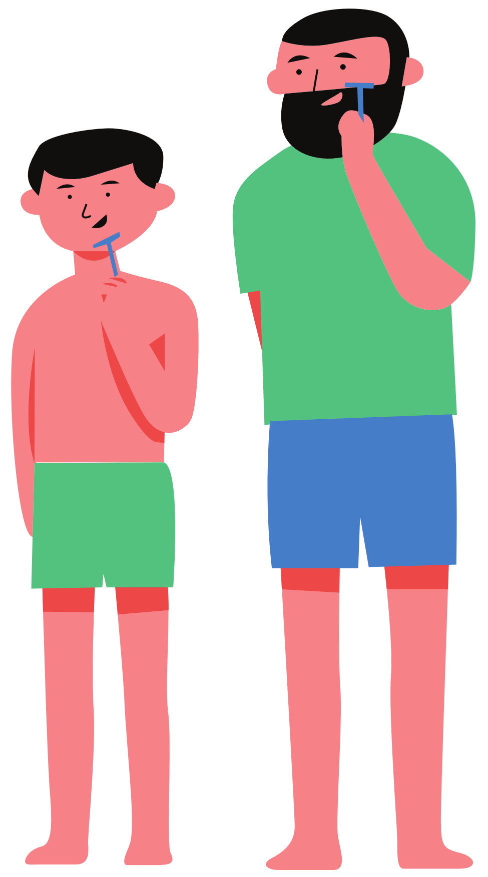
2. Updating

Component is being updated (its state or props changes)



$$\begin{tikzcd}
\text{getDerivedStateFromProps()} \arrow[d] \\
\text{shouldComponentUpdate()} \arrow[d] \\
\text{getSnapshotBeforeUpdate()} \arrow[d] \\
\text{render()} \arrow[d] \\
\text{componentDidUpdate()}
\end{tikzcd}$$

2. Updating Phase



2. Updating

Component is being updated
(its state or props changes)

[Demo: Updating Phase in Action]

1. `static getDerivedStateFromProps()`

The same as before

2. `shouldComponentUpdate()`

This method returns true or false based on a certain condition and determines whether a component should be updated or not based on its props or state

```
function shouldComponentUpdate(nextProps, nextState):boolean{
    // compare it with the component's current prop and state
   // and determine if you should update it or not
  return true // should update -> invoke render()
}
```

This method is useful in performance optimization

3. `render()`

if the `shouldComponentUpdate()` returns true, then render function will be re-invoked

4. `getSnapshotBeforeUpdate()`

In this method, we are given access to the props and state value before the update is committed to the DOM.

5. `componentDidUpdate()`

This method is the last method on updating phase, it receives the *former props and state* values as arguments and it receives the return value of `getSnapshotBeforeUpdate()` as third argument

3. Unmounting Phase

The last phase of the component represents the death of the component where it is being removed from the DOM and it has only one lifecycle method: **componentWillUnmount**

3. Unmounting

Component is dead and removed from the DOM

componentWillUnmount()

3. Unmounting Phase

3. Unmounting

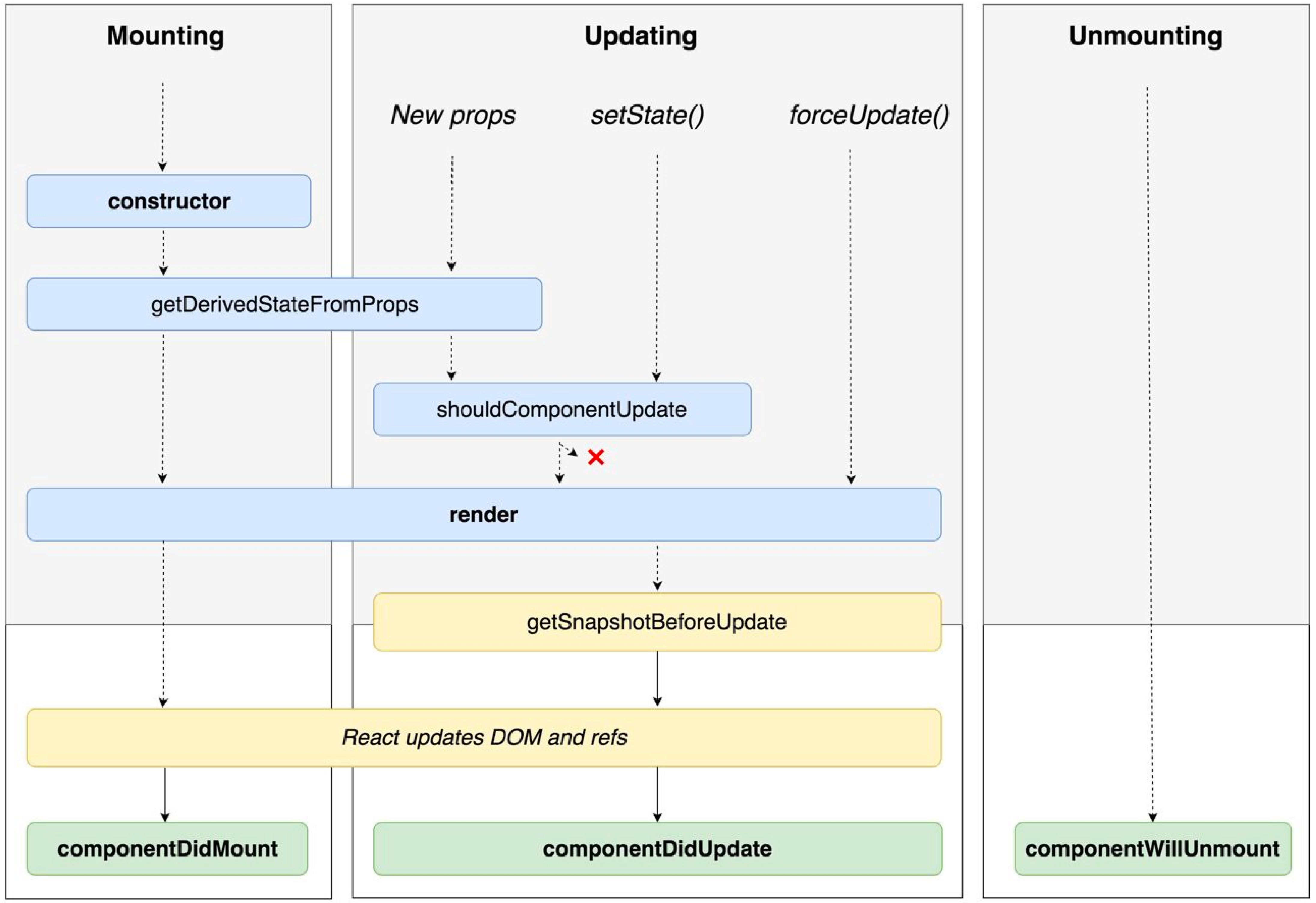
Component is dead and removed from the DOM

1. `componentWillUnmount()`

This method is executed right before the component is unmounted from the DOM. You can think of this method as a way to ***clean up*** anything that is needed to be removed before the component is destroyed.

it is helpful for optimization and prevent memory leaks

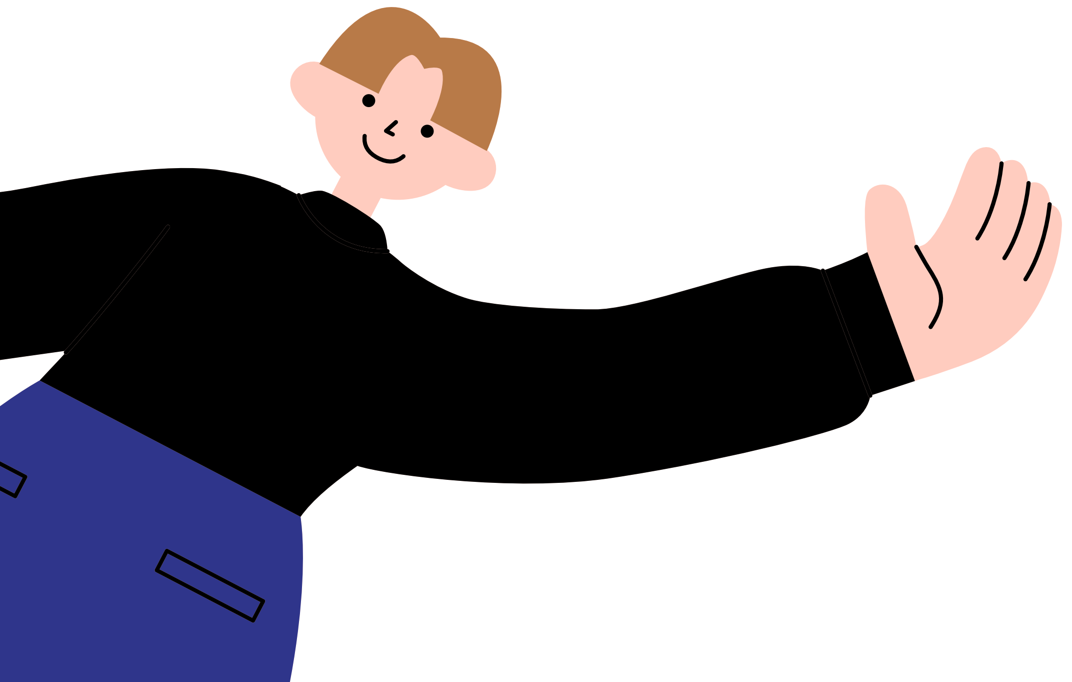
The Big Picture



[Component Lifecycle Methods in action]

Its Demo Time

Any Questions?

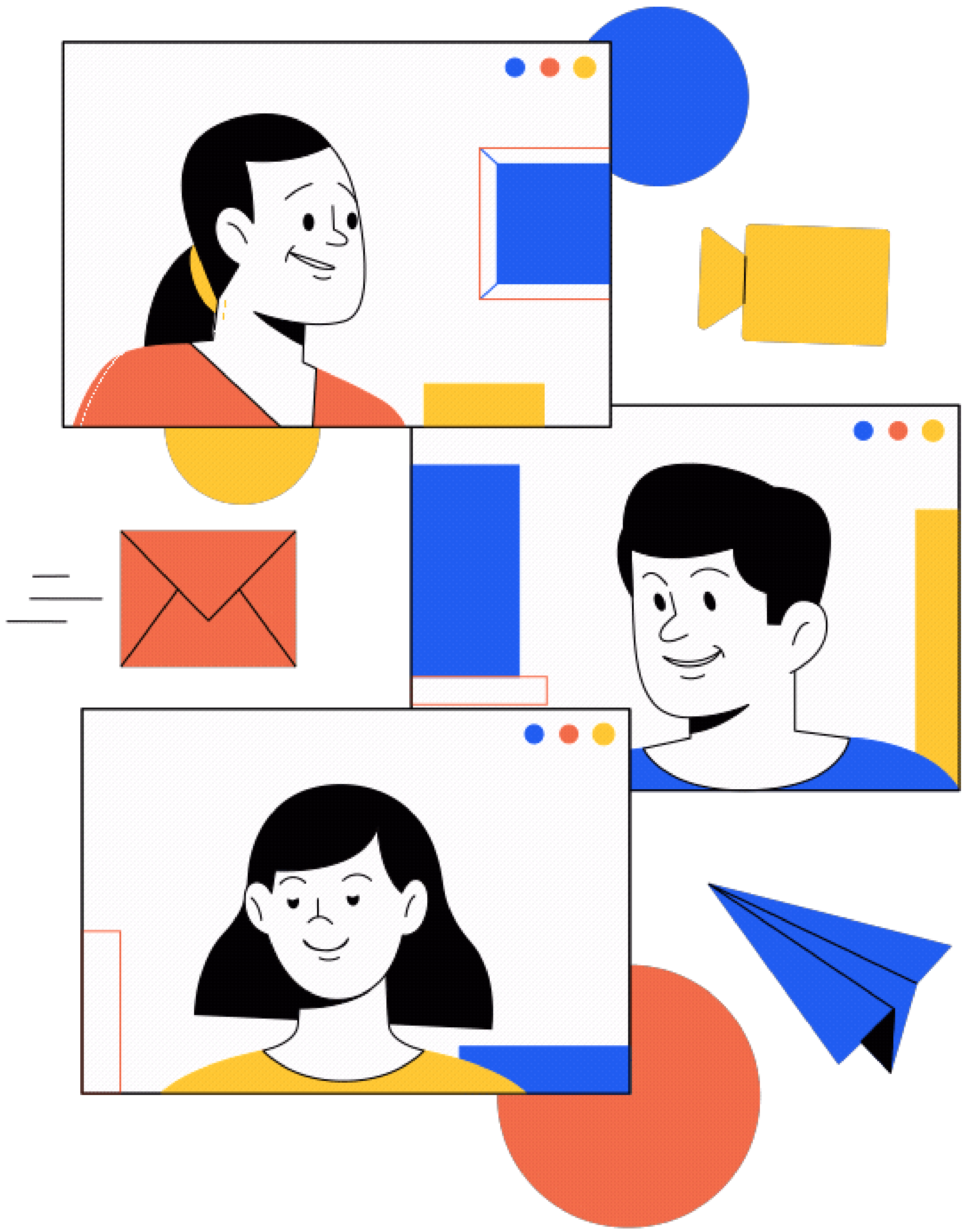


Did you like the session? What could be improved?

Please leave a review after the session and let me know your feedback.

Thank you!

For questions, requests and anything, please reach out to me on slack or email me at **aghonem2011@gmail.com**



Follow me on Github **@3ba2ii**
code and slides are found at this [github repo]