deeecode

LIFECYCLE METHODS

IN REACT CLASS COMPONENTS

swipe left

deeecode

LIFECYCLE METHODS

Think of components as **living things**, which go through several lifecycles from **birth** (mounted on the DOM) until they **die** (unmounted from the DOM).

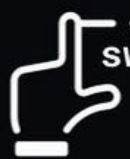
With lifecycle methods in React, you can execute some code when a lifecycle event is triggered.

There are many lifecycle methods, but we'll look at few of them as used in class components.

Lifecycles can be divided into three phases:

- birth (**mouting the component**)
- growth (**component data changing overtime**)
- death (**unmounting the component**)

We'll look at some methods that applies to these phases.



swipe left

LIFECYCLE METHODS

Mounting Lifecycle methods

When a component is to be mounted on the DOM, it goes through some lifecycles but we will look at **render** and **componentDidMount**.

The **render** method returns the element that will be rendered on the DOM (can be string, arrays, or React elements)

The **componentDidMount** method is called immediately the component is rendered on the DOM. Read this lifecycle as "**yes, the component mounted indeed**".

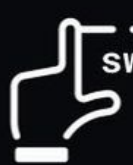
You can use both methods like this...

swipe left

LIFECYCLE METHODS

```
class App extends React.Component {
  // ...

  componentDidMount() {
    // run some code when component mounts
  }

  render() {
    return <h1>Hello world</h1>
  }
}
```

The component above renders the **h1** element with the “Hello world” text on the DOM, and immediately, the **componentDidMount** method is called. In this method, you can make API requests, update state, trigger some element animations and do whatever you want to do when the component mounts.



swipe left

LIFECYCLE METHODS

In the **componentWillUnmount** method, you can cleanup side effects, or do other things before the component leaves the DOM.

With React lifecycle methods, you can do many things at different points in a component's life. Right from birth, and as they grow until they die, you can make your API requests, update state or cause other types of side effects.

The methods we looked at here are for class components. To access these methods in functional components, you have to use hooks like **useEffect** and others.

Check out my React post on "**Concept of React Hooks**" to understand how to access state and lifecycles in functional components.

swipe left

deeecode

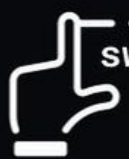
LIFECYCLE METHODS

Unmounting Lifecycle methods

When a component is about to die (is to be unmounted from the DOM), the **componentWillUnmount** lifecycle method is called.

In this method, the component can say its last prayers 😂

```
class App extends React.Component {
  // ...

  componentWillUnmount() {
    console.log("So sad to leave this world")
  }

  render() {
    return <h1>Hello world</h1>
  }
}
```



swipe left

LIFECYCLE METHODS

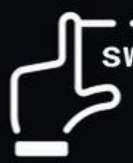
Update Lifecycle methods

By default, **shouldComponentUpdate** returns true.

Then, the **render** method is called again, to re-render the component (with the new state or props) on the DOM.

Immediately after re-rendering, similar to componentDidMount, **componentDidUpdate** is called. In this method, you can execute some code after the component updates. This method also receives the previous props and states as arguments, so that you can compare the props and state in the previous render with the ones in the new render, before running some code.

Here's what the three methods would look like:



swipe left

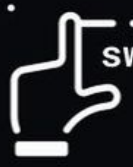
deeecode

LIFECYCLE METHODS

```
class App extends React.Component {
  // ...

  shouldComponentUpdate(nextProps,nextState) {
    if (nextState.isLoading === true) return false

    return true
  }

  componentDidUpdate(prevProps, prevState) {
    // run some code when component updates
  }

  render() {
    return <h1>Hello world</h1>
  }
}
```

shouldComponentUpdate receives the next props and state as argument so you can use that to determine whether the component should component. If **false**, **render** and **componentDidUpdate** will not be called.

If the component updates, **componentDidUpdate** will be called and you can run any code here.



swipe left

LIFECYCLE METHODS

Update Lifecycle methods

As a living thing, a component grows while on the DOM. The state of the component can change (for example, an **isLoading** state can be **false** initially then **true** later, that's growth). Or the props that the component receives from its parent element can also change.

When either of the props or state changes, you can say, the component is "**growing**" and some lifecycle methods are also triggered in this process. Three of them are **shouldComponentUpdate**, **render** and **componentDidUpdate**.

The **shouldComponentUpdate** method determines whether the component should be re-rendered on the DOM. If you return **true** here, it will be re-rendered, if **false**, it won't.

swipe left