@CODE.CLASH

React : useEffect Hooks

Next →

Hey Everyone 👋

React hook is very expressive. You can do so much by writing so little.

But, they're relatively difficult to get started. Especially useEffect().

In this post, you'll learn how and when to use useEffect() hook.

Do Like, save and Share This Post If You Found This Helpful.

Next →

Why useEffect?

- The useEffect Hook allows you to perform side effects in your components.
- If the functional component makes calculations that don't target the output value, then these calculations are named side-effects.
- Examples are : fetching data, directly updating the DOM, and timers.

Next →

useEffect?

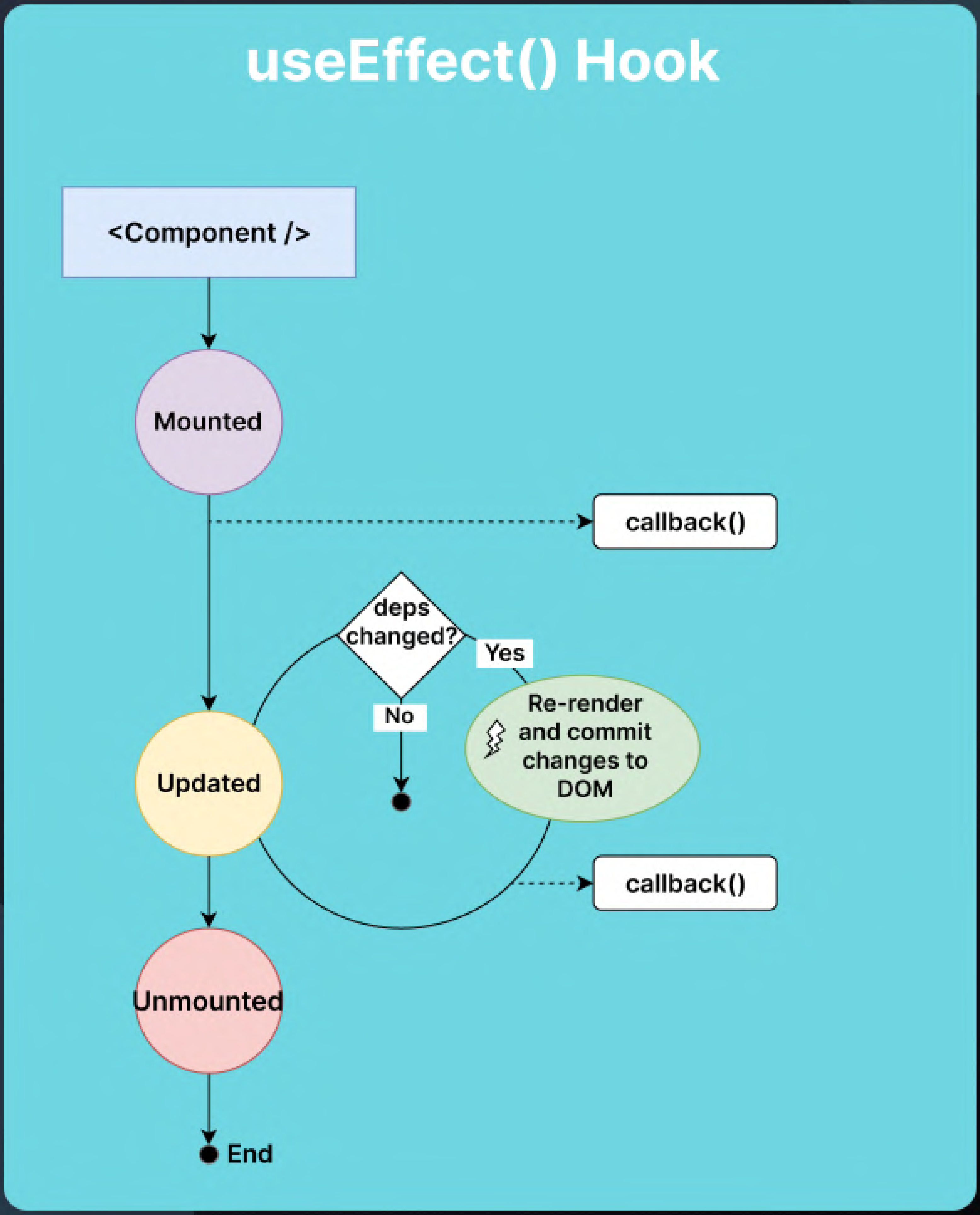
- useEffect() hook accepts 2 arguments:

```
useEffect(callback, [dependencies]);
```

- callback is the function containing the side-effect logic.
- dependencies is an optional array of dependencies.
- useEffect() executes callback only if the dependencies have changed between renderings.

NEXT →

- Put your side-effect logic into the callback function, then use the dependencies argument to control when you want the side-effect to run.



NEXT →

The correct way to perform the side effect in our User component is as follows:

1. We import useEffect from "react"
2. We call it above the returned JSX in our component
3. We pass it two arguments: a function and an array

```jsx
import { useEffect } from 'react';

function Greet({ name }) {
  const message = `Hello, ${name}!`; // Calculates output

  useEffect(() => {
    document.title = `Greetings to ${name}`; // Side-effect!
  }, [name]);

  return <div>{message}</div>; // Calculates output
}
```

NEXT →

Dependencies argument

dependencies argument of useEffect lets you control when the side-effect runs.

1. Not provided: the side-effect runs after every rendering.

```
import { useEffect } from 'react';
function MyComponent() {
  useEffect(() => {
    // Runs after EVERY rendering
  }); //Dependencies argument Not provided
}
```

NEXT →

2. An empty array [ ]: the side-effect runs once after the initial rendering

```
import { useEffect } from 'react';
function MyComponent() {
  useEffect(() ⇒ {
    // Runs ONCE after initial rendering
  }, []); //Dependencies argument Is empty array[]
}
```

3. Has props or state values : the side-effect runs only when any dependendecy value changes.

```
function MyComponent({ prop }) {
  const [state, setState] = useState('');
  useEffect(() ⇒ {
    // Runs ONCE after initial rendering
    // and after every renderingONLY IF `prop` or `state` changes
  }, [prop, state]);
}
```

NEXT →

Best Of Luck :)

NEXT →

@Code.Clash

THANKS FOR YOUR ATTENTION

Code.Clash

Imtiyaz Nandasaniya

LIKE AND SAVE IT FOR LATER