Life Cycle methods in React JS

Life cycle methods allows us to hook into different stages of a component's life cycle.

These methods help manage & control behaviours during a component's rendering, updating & unmounting

Phases of a Component's Lifecycle

1. mounting – when the component is being created & inserted into the DOM.

2. updating – when the component's state or props change, causes a re-render.

3. unmounting – when the component is removed from the DOM.

Mounting

1. Constructor - initialize the state.

```
constructor (props) {
    super (props);
    this.state = { count : 0 };
}
```

2. render — Defines what gets rendered on the screen

" you should not include any side effects fn"

3. Component Did Mount() → Side effect method

"Runs only once after the component is mounted"

```
Component Did Mount() {
    fetch ('api'). then ( res => res. json()) .
}
```

② Updating

① shouldComponentUpdate ⇒ whether to re-render or not

↓ true

re-render will happen

↓ false

re-render will not happen

② render

* getSnapshotBeforeUpdate

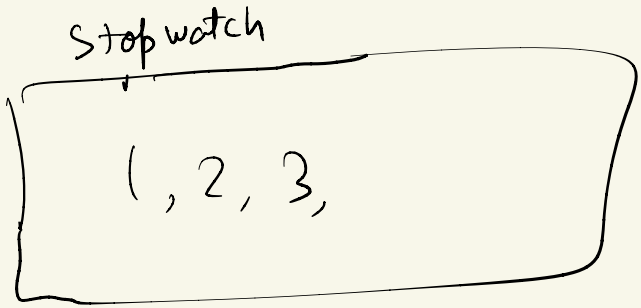
↳ capture DOM before DOM updates

* componentDidUpdate ⇒ performs side effects after updating

---

(3) Un mounting

1. Component Will unmount ()



useEffect(() => {

~~true~~ → (A)

return () => {

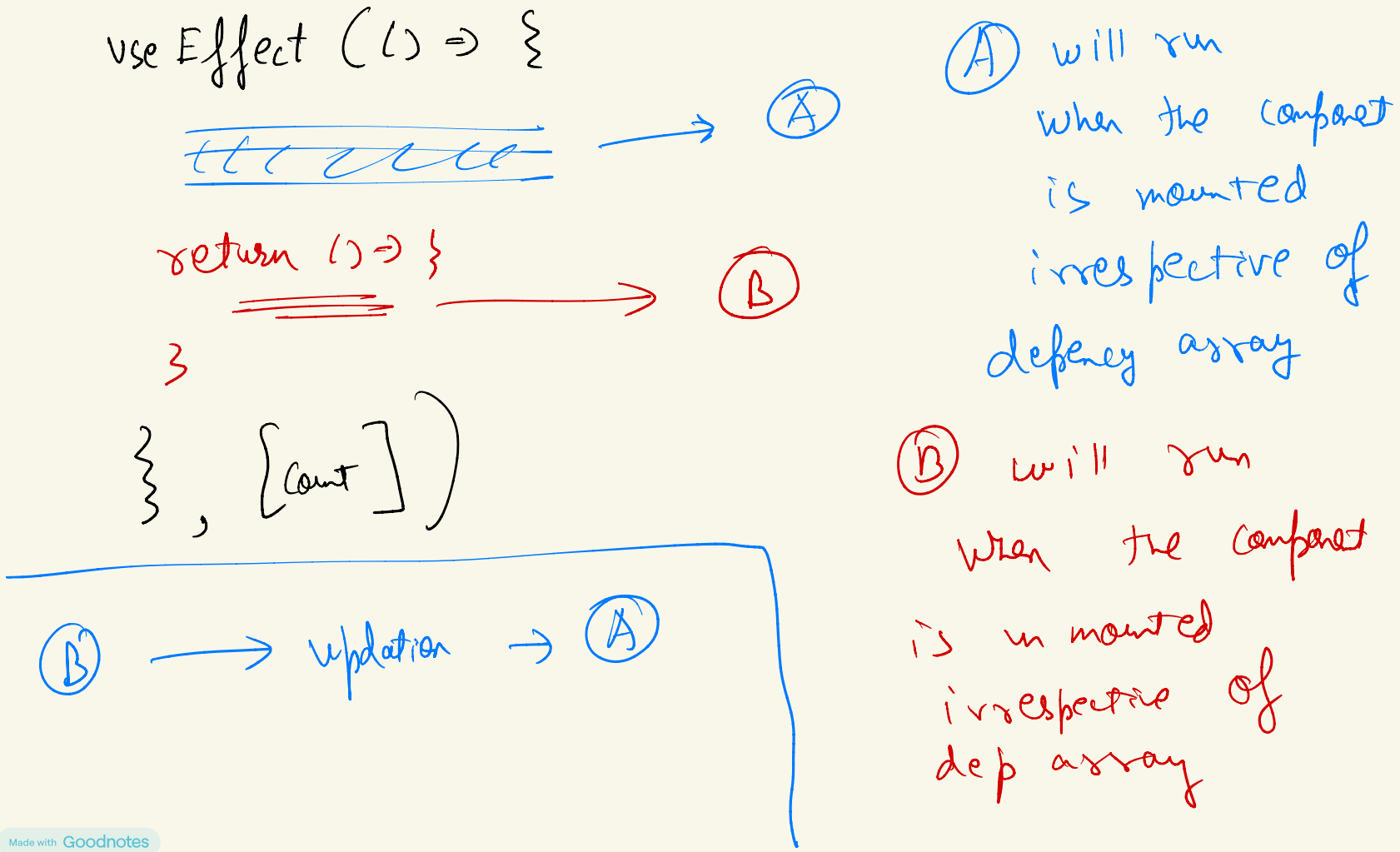
→ (B)

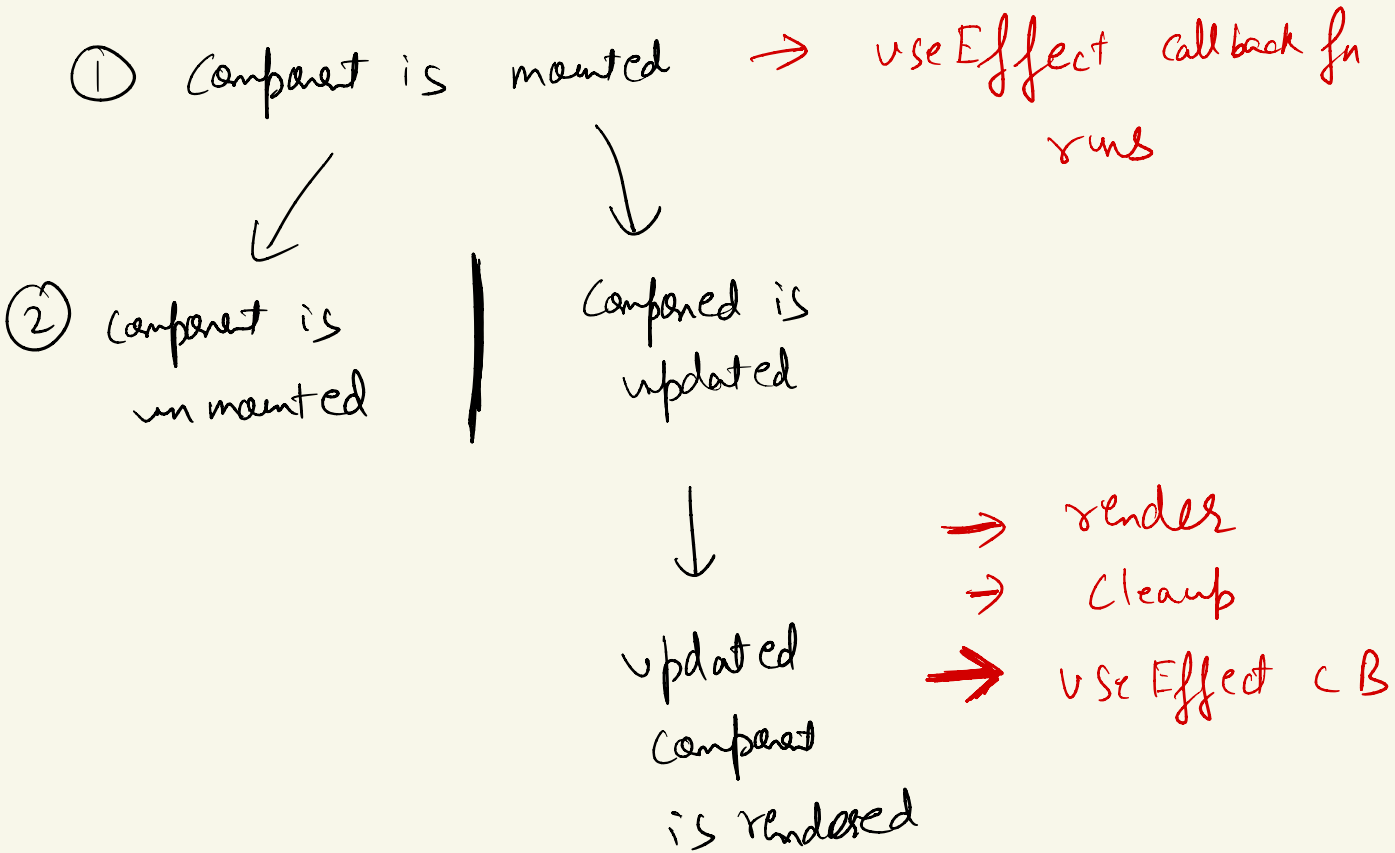
}

}, [ ])

(A) will run when the component is mounted irrespective of dependency array

(B) will run when the component is unmounted irrespective of dep array





```
useEffect(() => {

});
```

```jsx
const UsePrevComp = () => {
  const [count, setCount] = useState(0);
  const prevCount = useRef(count);

  useEffect(() => {
    prevCount.current = count;
  }, [count]);

  return (
    <div>
      <h1>current count: {count}</h1>
      <h2>prev count: {prevCount.current}</h2>
      <button onClick={() => setCount(count + 1)}>increase count</button>
    </div>
  );
};

export default UsePrevComp;
```