W8D3 - UseEffect()

miguel.garrido@ironhack.com

Components

They re-render whenever there is a change on a **prop** or **state**

Components and Side Effects

Some functions are outside the React rendering flow:

- Fetching
- Timer (SetInterval, setTimeout)
- Subscriptions (WebSockets)
- Event Listeners.

UseEffects()

Short for side effects. They help us to setup on three stages of a component:

1. Mounting
2. On update phase
3. On unmounting phase

UseEffects()

Basic syntax:

```
useEffect(fn, dependencies);
```

`fn`

Side effect logic

`dependencies`

An array of values that trigger the effect to re-run when they change.

UseEffects() - [ ]

Basic syntax:

```
useEffect( fn, dependencies);
```

When:

```
dependencies = []
```

The fn logic will only be executed on **mounting**

UseEffects() - no dependencies

Basic syntax:

```
useEffect( fn );
```

When **dependencies** is not specified **fn** logic will render on every update ⚠️

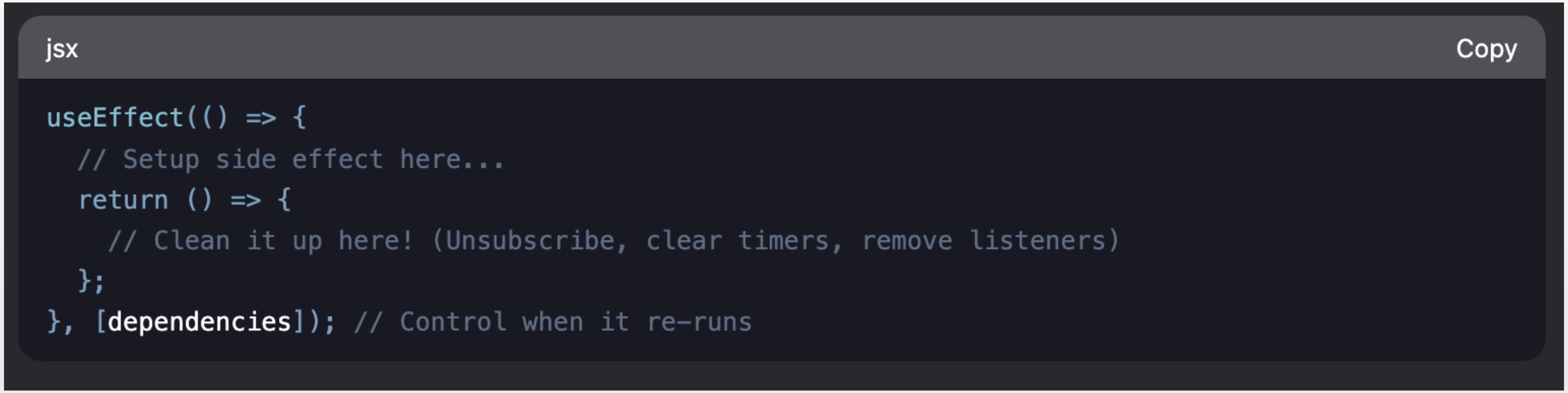
UseEffects() - [val1, val2]

Basic syntax:

```
useEffect( fn, [val1, val2]);
```

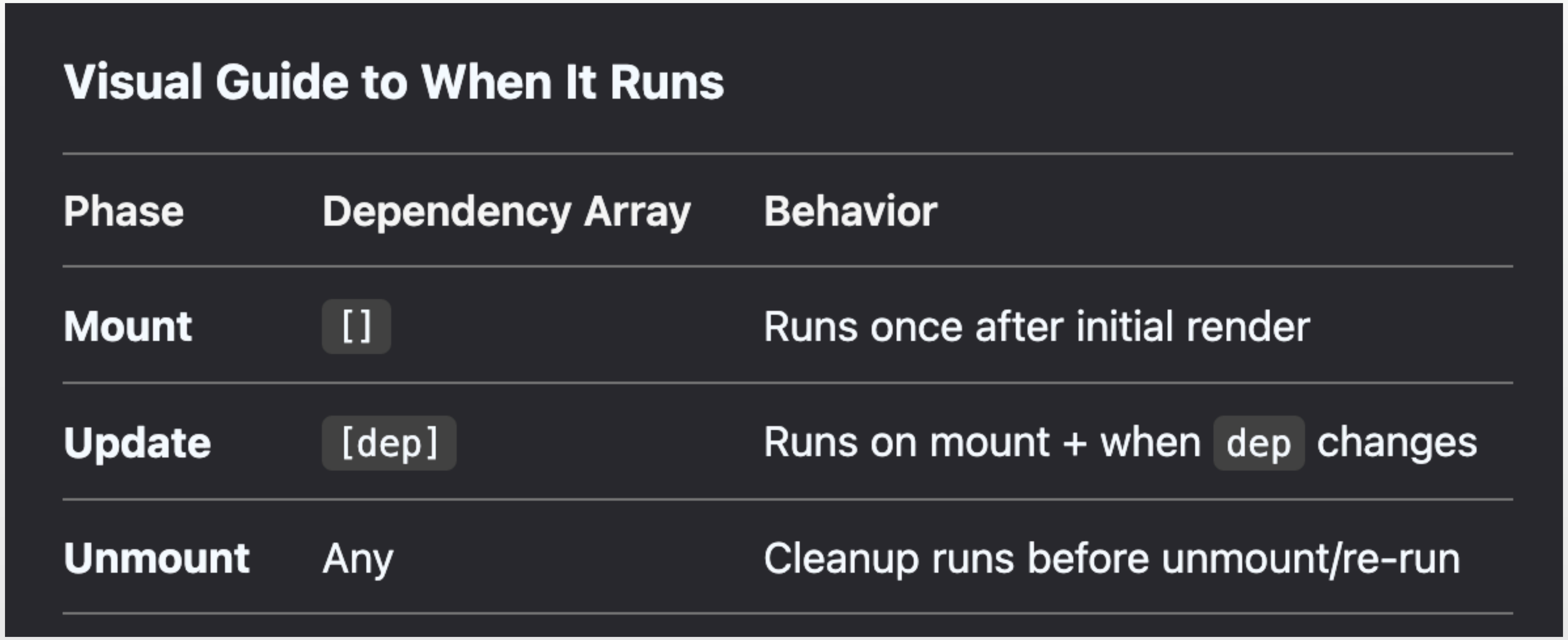
fn run when **val1** or **val2** changes

UseEffects()



```jsx
useEffect(() => {
  // Setup side effect here...
  return () => {
    // Clean it up here! (Unsubscribe, clear timers, remove listeners)
  };
}, [dependencies]); // Control when it re-runs
```

UseEffects()



Visual Guide to When It Runs

Phase	Dependency Array	Behavior
Mount	`[]`	Runs once after initial render
Update	`[dep]`	Runs on mount + when `dep` changes
Unmount	Any	Cleanup runs before unmount/re-run

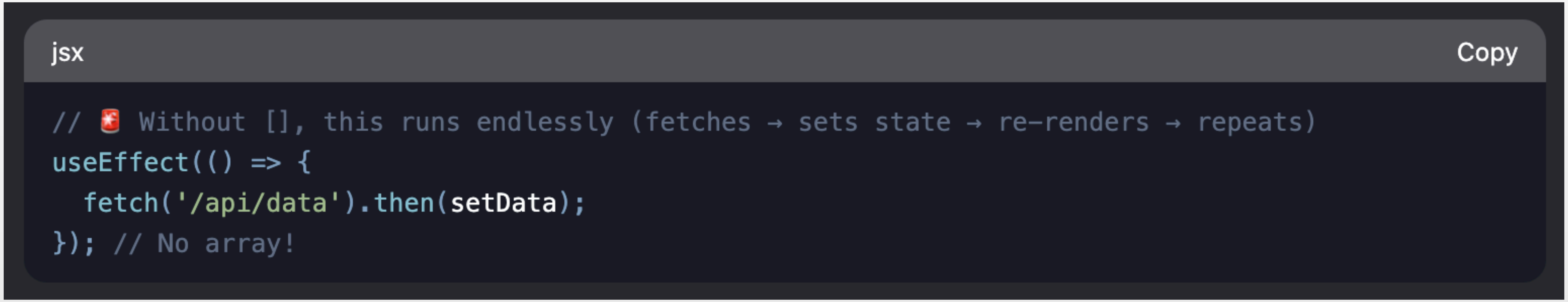
UseEffects() on Mounting

Useful for:

1. Fetching data
2. Analytic tools (Eg: Google) and other third-party libraries

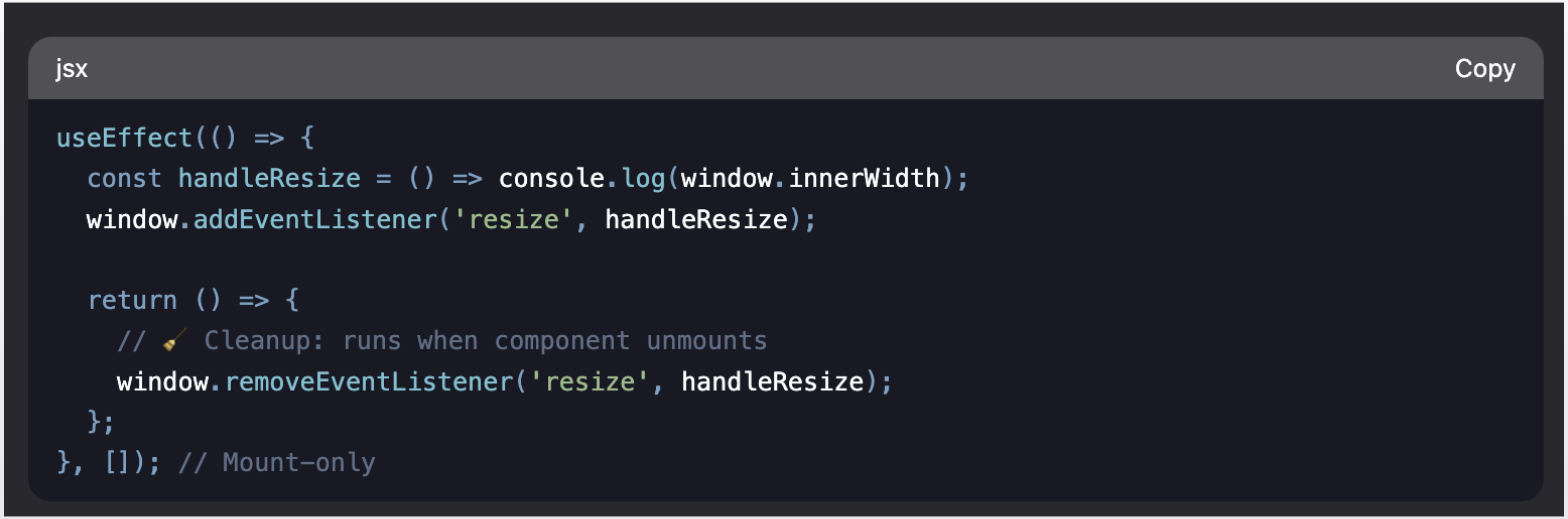
Avoids unnecessary re-runs, or re-fetch

UseEffects() - Example #1 - Any update


jsx Copy

```jsx
// 🚨 Without [], this runs endlessly (fetches → sets state → re-renders → repeats)
useEffect(() => {
  fetch('/api/data').then(setData);
}); // No array!
```

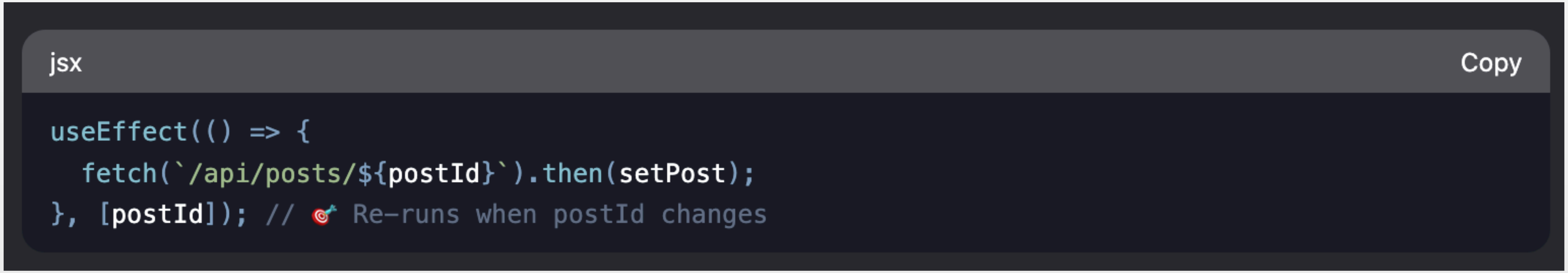
UseEffects() - Example #2 - Mount phase


jsx Copy

```jsx
useEffect(() => {
  const handleResize = () => console.log(window.innerWidth);
  window.addEventListener('resize', handleResize);

  return () => {
    // 🧹 Cleanup: runs when component unmounts
    window.removeEventListener('resize', handleResize);
  };
}, []); // Mount-only
```

UseEffects() - Example #3 - Update phase



jsx
Copy

```jsx
useEffect(() => {
  fetch(`/api/posts/${postId}`).then(setPost);
}, [postId]); // 🎯 Re-runs when postId changes
```

*Ps: **fetch** has been simplified here* 😉

UseEffects()

Now, we need to talk about **functional updates**