Slim

React js tips

How does useEffect work?

A Beginner's Guide

Slim toumi

Slim toumi

Slim

React js tips

🛠️ What is useEffect ?

useEffect is a React hook that lets you perform side effects in function components, like:

- Fetching data
- Updating the DOM
- Setting up subscriptions

🔍 Basic Syntax :

App.jsx

```
useEffect(() ⇒ {
  // Effect code here
  return () ⇒ {
    // Cleanup code here (optional)
  };
}, [dependencies]);
```

Slim toumi

Slim toumi

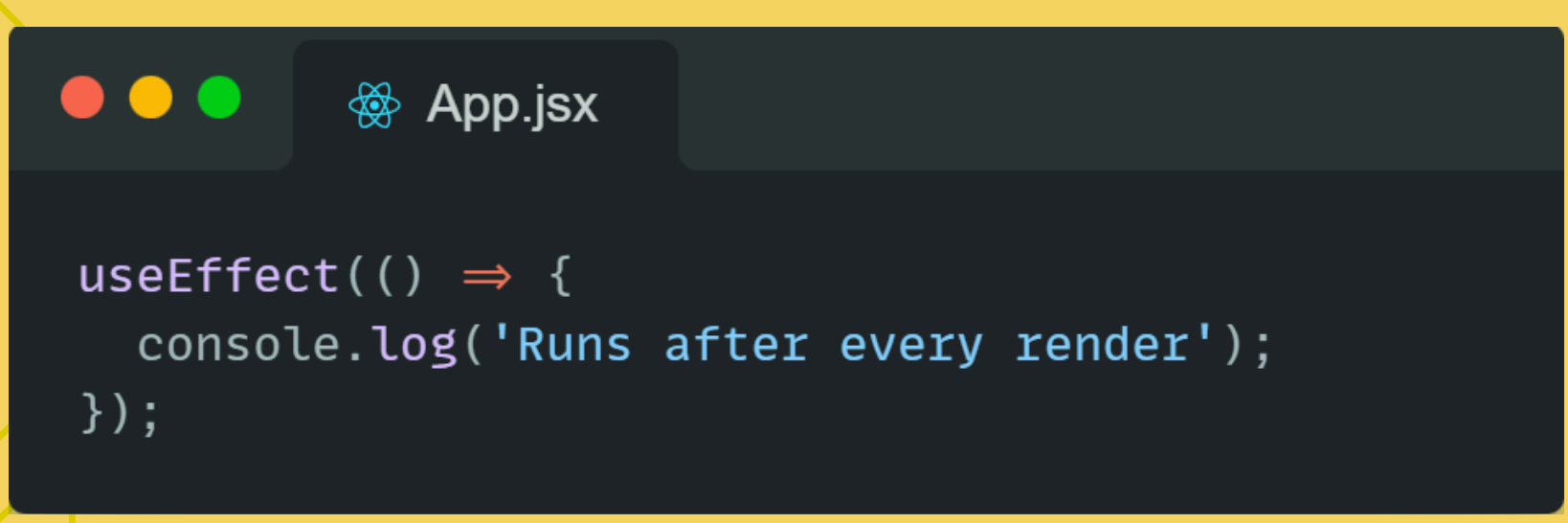
Slim

React js tips

🕒 When Does useEffect Run ?

1. After Every Render (No Dependency Array)

If you call useEffect without a dependency array (no [] at the end), the effect will run after every render of the component. This means that useEffect will execute its code every single time the component updates.



When This Runs:

- Runs after the component initially mounts.
- Runs after every update caused by changes in state or props.

Use Case:

This is useful if you want something to happen every time the component re-renders, but it's generally avoided in favor of more specific dependency triggers (to avoid performance issues).

Slim toumi

Slim toumi

2. On Mount and Unmount (Empty Dependency Array [])

If you provide an empty dependency array ([]), useEffect will run only once, right after the initial render, and then never again unless the component unmounts (in which case, it'll run any cleanup code).

App.jsx

```
useEffect(() ⇒ {
  console.log('Runs only once on mount');

  return () ⇒ console.log('Runs cleanup on unmount');
}, []);
```

When This Runs:

- **On Mount**: Right after the component mounts (first render).
- **On Unmount:** Any return function will run as cleanup when the component unmounts.

Use Case:

This is ideal for things you only want to set up once, like:

- Fetching initial data.
- Setting up a subscription or WebSocket connection.
- Initializing a timer or interval.

3. Dependencies Change (Specified Dependency Array)

If you provide a dependency array with specific values (like [count, user]), useEffect will only re-run if any of these dependencies change. React will compare each item in the dependency array to its previous value; if any value has changed, useEffect will run again.

App.jsx

```
useEffect(() => {
  console.log('Runs when `count` changes');
}, [count]);
```

When This Runs:

- Runs initially on mount (first render).
- Runs any time a dependency changes.

Use Case:

This approach allows you to control exactly when useEffect should re-run, which is efficient and powerful. Common uses include:

- Re-fetching data whenever a specific value changes (e.g., a userId or a page number).
- Re-calculating values when one of your dependencies updates.

Slim

React js tips

⚡ Side Effects Examples :

- **Data Fetching**

App.jsx

```
useEffect(() => {
  async function fetchData() {
    const data = await fetch('/api/data');
    // handle data
  }
  fetchData();
}, []); // Only on mount
```

- **Setting Timers**

App.jsx

```
useEffect(() => {
  const timer = setInterval(() => {
    console.log('Timer is running');
  }, 1000);

  return () => clearInterval(timer); // Cleanup on unmount
}, []);
```

Slim toumi

Slim toumi

Slim

React js tips

🧹 Cleanup Function :

- **Purpose:** Used to clear resources (like subscriptions or timers) when the component unmounts or dependencies change.
- **When to Use:** Useful for cleanup like unsubscribing from a data source, clearing timers, etc.

👀 Common Pitfalls :

- **Infinite Loops:** Avoid leaving out dependencies that should be included; otherwise, the effect could cause infinite re-renders.
- **Updating State in useEffect:** Be cautious with setting state in useEffect, as it can trigger additional renders.

Slim toumi
Slim toumi

Slim

React js tips

Hopefully You Found It Usefull!

Be sure to save this post so you can come back to it later

like

Comment

Share

Slim toumi

Slim toumi