



## Courses



# useEffect Hook

Manage with your component's lifecycle with the useEffect hook

React is known for its powerful state management process that doesn't require any page reload. One of the tools used for managing the state is the useEffect hook, that deals with the



component's lifecycle. Read on to learn how to use the `useEffect` hook.

## CodeSandbox link

You can find the full code for this tutorial at <https://codesandbox.io/s/useeffect-yuwht>.

## Understanding the component lifecycle

Using the `useEffect` hook means understanding the lifecycle of the component. This lifecycle consists of three main parts: mounting, updating, and unmounting. If you're familiar with the old way to deal with the component lifecycle, these three parts correspond to **`ComponentDidMount`**, **`ComponentDidUpdate`**, **`ComponentWillUnmount`**.

To understand better a component's lifecycle, we can look at the lifecycle of a flower. When the flower's seed is planted, it is **mounted**. Then, as it grows, it is **updating** and, unfortunately, it will die (or **unmount**) someday. The component's lifecycle is exactly the same: when the user lands on the page, the component is mounted, then as the states update, the component is also updated, and, finally, the component is unmounted when the user leaves the page.

## Anatomy of the `useEffect` hook

The component lifecycle is represented in three different parts of the `useEffect` hook, as shown in the code snippet below.



```
    return () => {  
      // Cleanup function  
    }  
  }, [//Updating])
```

The first part is the mounting part. That's when the component is initially rendered when the user lands on the page. The return function is the cleanup function, or when the user leaves the page and the component will unmount. The array is the last part, and it is where you put the states that will update throughout the component's lifecycle. You can discard the array if your component won't update during its lifecycle.

## When to use the useEffect hook?

The useEffect hook is useful when you wish to run some functions during the component's lifecycle. For example, if you want to update the UI when a state changes, the useEffect hook is the way to go. You can also define a state on first load (when `componentDidMount`), and also clean the state when the component is unmounting (`componentWillUnmount`).

Let's look at an example below. We want to update the page title every time the user clicks on a button on the page, using the `useEffect` and `useState` hooks.

## Create your component

Start by creating your component. We'll reuse the button component we created in the [useState hook section](#) and incorporate the `useEffect` hook as well.

```
import React, { useState } from "react"
```



```
Button = () => {  
  const [count, setCount] = useState(0)
```

```
    return <button onClick={() => setCount(count + 1)}>You clicked {count} times</button>
  }

  export default Button
```

## Import useEffect

In order to use **useEffect**, you'll need to import it at the top of the file. Don't forget to keep **React** and **useState** as we still need them.

```
import React, { useState, useEffect } from "react"
```

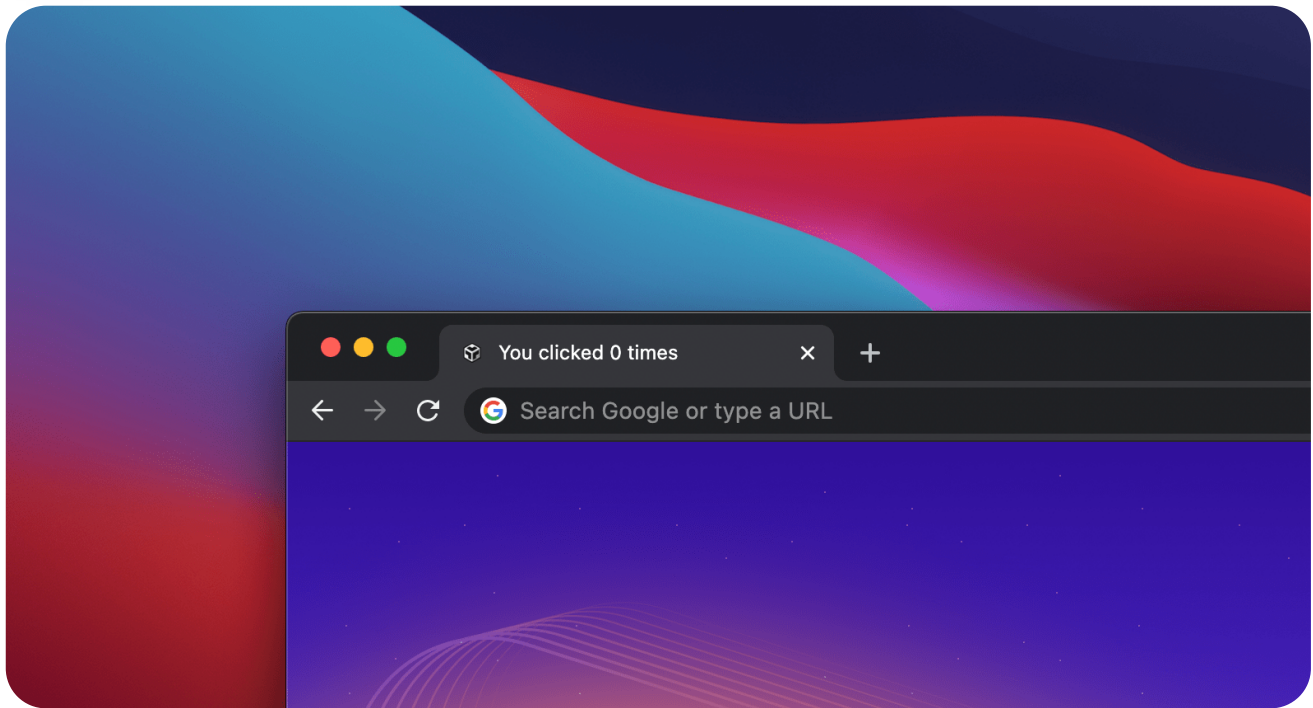
## Define useEffect

Next, create a new **useEffect** just above the return statement in your component. We want to update the document title every time the user clicks on the button, so let's set it in the mounting part of the **useEffect** hook.

```
useEffect(() => {
  document.title = `You clicked ${count} times.` // This is the mounting part
}, [])
```

You'll see the title of the tab change to **You clicked 0 times**, since the value of **count** is initially set to 0.



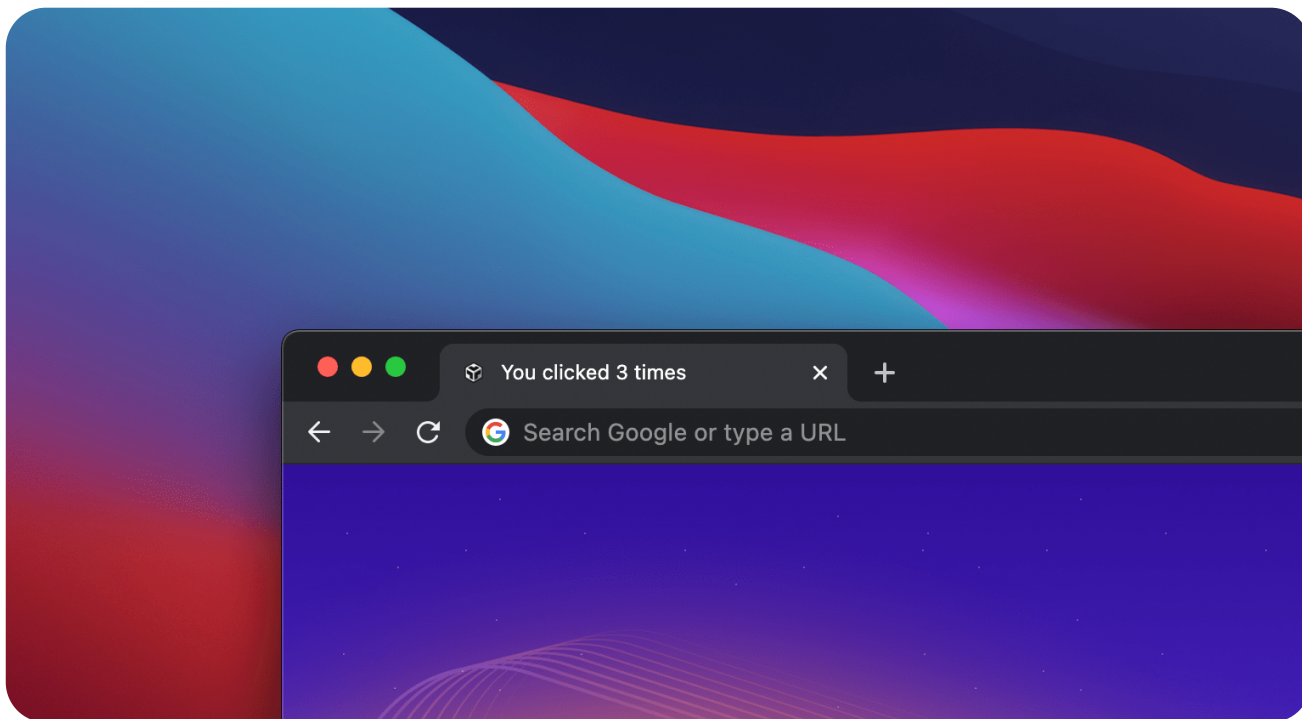


However, when you click on the button, the text on the button updates but not the title of the tab. Therefore, we need to add the count state in the update array of the `useEffect` hook. Every time the **count** is updated, the `useEffect` will be re-run, updating the document title in our case.

```
useEffect(() => {  
  document.title = `You clicked ${count} times.`  
}, [count]) // Add the count state in the array here
```

Now, every time you click on the button, the title of the tab will update to reflect the number of times the user clicked on the button.





## Final code with styling

You can style your components by using the [styled-components](#) package in your project. Learn more about it in [the Styling in React section](#) of this handbook.

The final code incorporating `useEffect` as well as `useState` and `styled-components` will look like this:

```
import React, { useState, useEffect } from "react";
import styled from "styled-components";
```

```
const MyButton = () => {
  const [count, setCount] = useState(0);
```

```
  useEffect(() => {
    document.title = `You clicked ${count} times`;
    setCount(count);
```



```
    return (  
      <StyledButton onClick={() => setCount(count + 1)}>  
        You clicked {count} times  
      </StyledButton>  
    );  
  };  
  
export default MyButton;  
  
const StyledButton = styled.button`  
  background: linear-gradient(91.4deg, #2fb8ff 0%, #9eecd9 100%);  
  padding: 12px 0;  
  width: 200px;  
  border: none;  
  border-radius: 30px;  
  color: white;  
  font-weight: bold;  
  font-family: Segoe UI, sans-serif;  
`;  
;
```

**Learn with videos and source files. Available to Pro subscribers only.**



**GET PRO ACCESS**

\$19 per month



Purchase includes access to 30+ courses,  
100+ premium tutorials, 120+ hours of videos,  
source files and certificates.

BACK TO  
**useState Hook**




READ NEXT  
**useRef Hook**



## TEMPLATES AND SOURCE CODE

# Download source files

Download the videos and assets to refer and learn offline without interruption.

-  **Design** template
-  **Source code** for all sections
-  **Video files**, ePub and subtitles


Videos

ePub

Assets





- |   |   |      |    |   |      |
|---|---|------|----|---|------|
| 1   | <b>Intro to React Hooks</b><br>An overview of React Hooks and the frameworks you can use to...        | 3:39 | 2  | <b>Create your first React app</b><br>Create your first React project from the Terminal and save it on... | 4:23 |
| 3   | <b>React Component</b><br>Create your first JSX component using React                                 | 2:54 | 4  | <b>Styling in React</b><br>How to style your React components using inline styling,...                    | 5:06 |
| 5   | <b>Styles and Props</b><br>Render different styles depending on different properties passed to...     | 2:22 | 6  | <b>Understanding Hooks</b><br>Learn about the basics of React Hooks, which introduced at Reac...          | 3:21 |
| 7   | <b>useState Hook</b><br>Use the useState hook to manage local state in your React...                  | 2:54 | 8  | <b>useEffect Hook</b><br>Manage with your component's lifecycle with the useEffect hook                   | 3:41 |
| 9   | <b>useRef Hook</b><br>Learn about the useRef hook, which replaces the JavaScript...                   | 3:00 | 10 | <b>Props</b><br>Learn about props in React to pass data from parent to child...                           | 3:11 |
| 11  | <b>Conditional Rendering</b><br>Render different UIs depending on different conditions and states     | 4:21 | 12 | <b>Load Local Data</b><br>Load local JSON data into your React application                                | 4:04 |
|  | <b>Fetch Data from an API</b><br>Learn the basics of asynchronous functions and promises by fetchi... | 5:40 | 14 | <b>Toggle a state</b><br>Learn how to toggle a state from true to false and back again                    | 4:05 |

- |  |   |
|--|---|
| <p><b>15</b>     <b>useInput Hook</b>     6:04</p> <p>Create a hook to get the value and the onChange event of input fields</p>            | <p><b>16</b>     <b>Intro to Firebase</b>     6:59</p> <p>Learn about Firebase and set it up in your React project</p>                    |
| <p><b>17</b>     <b>Firebase Auth</b>     11:59</p> <p>Set up authentication in your React application using Firebase Auth</p>             | <p><b>18</b>     <b>Firestore</b>     10:51</p> <p>Enable Firestore as your database in your React application</p>                        |
| <p><b>19</b>     <b>Firebase Storage</b>     6:40</p> <p>Enable Firebase Storage in your application to store photos or...</p>             | <p><b>20</b>     <b>Gatsby and React</b>     6:44</p> <p>Create a static content-oriented website using React on Gatsby</p>               |
| <p><b>21</b>     <b>NextJS and React</b>     5:24</p> <p>Create your first NextJS React application</p>                                    | <p><b>22</b>     <b>React TypeScript Par...</b>     8:19</p> <p>Learn how to create a React TypeScript application using the...</p>       |
| <p><b>23</b>     <b>React TypeScript Par...</b>     7:35</p> <p>Learn the basics of TypeScript and how to use TypeScript in a React...</p> | <p><b>24</b>     <b>useScrollPosition Hook</b>     4:26</p> <p>Create a custom hook to listen to the current window position of th...</p> |
| <p><b>25</b>     <b>useOnScreen hook</b>     8:08</p> <p>Create a custom hook to listen to when an element is visible on...</p>            | <p><b>26</b>     <b>useContext Hook</b>     8:32</p> <p>Manage global states throughout the entire application</p>                        |
| <p><b>27</b>     <b>Fragments</b>     2:43</p> <p>Group multiple children together with React Fragments</p>                                | <p><b>28</b>     <b>Lazy Loading</b>     4:05</p> <p>Lazy Load heavy components to improve performance</p>                                |
| <p><b>29</b>     <b>React Suspense</b>     3:13</p> <p>Wait for data with React Suspense and React.lazy</p>                                | <p><b>30</b>     <b>Environment Variables</b>     4:43</p> <p>Make environment variables secret with a .env file</p>                      |
| <p><b>31</b>     <b>Reach Router</b>     5:31</p> <p>Create a multiple-pages React application with Reach Router</p>                       | <p><b>32</b>     <b>URL Params</b>     4:04</p> <p>Create unique URL with URL Params</p>  |
| <p><b>SEO and Metadata</b>     6:47</p> <p>Optimize a React application for search engines with React Helmet</p>                           | <p><b>34</b>     <b>Favicon</b>     3:03</p> <p>Add an icon to a React website</p>  |



- |   |  |
|---|--|
| <p><b>35</b>   <b>Dynamic Favicon</b>   2:14</p> <p>Change the favicon's fill color depending on the user's system...</p> | <p><b>36</b>   <b>PropTypes</b>   3:54</p> <p>Implement props type-checking with PropTypes</p>                 |
| <p><b>37</b>   <b>Custom PropTypes</b>   3:58</p> <p>Create a custom PropTypes using a validator function</p>             | <p><b>38</b>   <b>useMemo</b>   4:05</p> <p>Prevent unnecessary re-renders when the component stays the...</p> |
| <p><b>39</b>   <b>Serverless Email Sending</b>   10:02</p> <p>Use EmailJS to send an email without backend</p>            |  |

## Meet the instructor

We all try to be consistent with our way of teaching step-by-step, providing source files and prioritizing design in our courses.

6 COURSES - 23 HOURS



**New Chapter**   4 hrs

This is the description of my new chapter



**SwiftUI Concurrency**   3 hrs

Concurrency, swipe actions, search feature,...



**SwiftUI Combine and Data**   3 hrs

Learn about Combine, the

### Stephanie Diep

FRONTEND AND WEB DEVELOPER

Developing web and mobile applications while learning new...



MVVM architecture, data,...



SwiftUI II Advanced

3 hrs

Home

Downloads

Courses

Search

Tutorials

Account

Pricing

Licenses

Updates



Site made with React, Gatsby,  
Netlify and Contentful. Learn **how**.

Community - **Discord Channel**

Design+Code © 2021

**Terms of Service - Privacy Policy**

Need help? **Contact Us**

