

Ultimate Guide to `useEffect()` in React

Art By
Cameron Mark



What is useEffect ?

useEffect() is a React Hook which let us run side Effects in our components



But what are Side Effects?

Side effects are those actions performed by the component to affect something outside of the component itself like updating state, calling an API, working with caches, modifying DOM etc

in Simple words , any action which connects the component to the outside world is a Side Effect

Syntax:- useEffect hook takes 2 arguments



```
useEffect(() => {  
  //code here  
}, [dependencies])
```

The diagram shows a code editor window with three dots in the top left corner. Inside, the code `useEffect(() => {
 //code here
}, [dependencies])` is displayed. A curved arrow points from the text 'First argument is a function which you want to execute whenever useEffect runs' to the function `() => {`. Another curved arrow points from the text 'Second argument is an array of dependencies' to the array `[dependencies]`.

*First argument is a function
which you want to execute
whenever useEffect runs*

Second argument is an array of dependencies

But when useEffect() actually executes?

- So , useEffect() always executes when the component is rendered for the first time (when component mounts)
- useEffect() also executes whenever the second argument (dependency array) changes which cause a re-render of the component
- Remember useEffect (side effects) only runs after the component renders or re-renders , not before or not along the render but only after

Case 1:- Passing array of dependencies

```
useEffect(()=>{  
  console.log("useEffect() in the house")  
},[test])
```

→ In this , we have passed test (state variable) as a dependency to the useEffect hook

Order of execution

- *"useEffect() in the house"* will console out because component renders for the first time so useEffect executes
- Now , whenever the dependency changes which means test variable changes , useEffect will execute and console "useEffect() in the house"

Case 2:- Passing an empty array

```
useEffect(()=>{  
  console.log("useEffect() in the house")  
}, [])
```

Empty Dependency Array

Order of execution

- *"useEffect() in the house" will console out because component renders for the first time so useEffect executes*
- *now there is no variable inside our dependency array so useEffect will never run again*

Case 3:- Passing no array

```
useEffect(()=>{  
  console.log("useEffect() in the house")  
})
```

→ No Dependency Array

Order of execution

- *"useEffect() in the house" will console out because component renders for the first time so useEffect executes*
- *now there is no array available so useEffect will run everytime when the component re-renders no matter why or how the component re-renders*

Cleanup Function in useEffect()

- Cleanup Function in useEffect is an optional function that can be returned from the useEffect hook
- This function is used to perform necessary cleanups after the side effects executes

```
useEffect(()=>{  
  //Effect Code here  
  return ()=>{  
    //cleanUp Code Here  
  }  
})
```

This is how we return a Cleanup Function

Syntax and all is fine but i dont understand Cleanups?

lets see an example

Example

```
useEffect(() => {  
  // Perform side effect  
  const interval = setInterval(() => {  
    // Update state  
  }, 1000);  
  
  // Define cleanup function  
  return () => {  
    clearInterval(interval);  
  };  
}, [dependencies]);
```

- Here we are using `setInterval()` inside `useEffect` to update the state at every 1000ms or 1s
- And we are returning a function (cleanup) which clears the side effect or specifically here clears the Interval
- let's see Order of Execution




```
useEffect(() => {  
  // Perform side effect  
  const interval = setInterval(() => {  
    // Update state  
  }, 1000);  
  
  // Define cleanup function  
  return () => {  
    clearInterval(interval);  
  };  
}, [dependencies]);
```

- *first cleanUp function executes before the useEffect hook so it can clear the previous useEffect execution*
- *or in simple words!, cleanUp function executes everytime when the next useEffect(side effect) is executed*

Order of Execution

- *first useEffect runs on the first render and after that cleanUp executes*
- *Now , whenever the component re-renders first cleanUp function executes and then useEffect hook*

Follow for more Posts like this

@codebysid

**Multiple ways to
Style React**



**5 GitHub Repo
every coder should
know about**

