

Skills
Network

**Side-effects Management
and Custom Hook**

IBM

What you will learn



Define the role of `useEffect` and side effects in React



Describe how `useEffect` and side effects work in React



Describe the various dependencies in `useEffect`



Describe a custom hook in React

What is useEffect?

A React hook



Allows side effects

```
11 - 0100
11001-1
0-0-0
1100-01
```

- Any operation to execute on page upload
- Asynchronous actions
- Affect the application's state

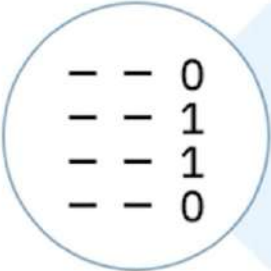
useEffect and side effects example

```
import React, { useState, useEffect } from 'react';
function SideEffect() {
  const [foods, setFoods] = useState([]);
  useEffect(() => {
    fetch('https://api.npoint.io/d542b9ad99f501ab3dbf')
      .then(response => response.json())
      .then(data => {
        console.log(data);
        setFoods(data);
      })
      .catch(error => console.error('Error fetching users:', error));
  }, []); // Empty dependency array means this effect runs only once when
  the component mounts
  return (
    </div> //return statement completed on next slide
  );
}
```

useEffect and side effects example

```
//return statement continued from previous screen
<h1>Food List</h1>
<ul>
  {foods.map((food)=>{
    return (<>
      <li><h1>{food.name}</h1></li>
      <p>food.description</p> <p>food.price</p>
      <p>food.category</p> <p>food.ingredients</p>
      <img src={food.image_url} alt="" height='100px' width='100px' />
    </> )
  })}
</ul>
</div>
);
}
export default SideEffect;
```

Dependencies in useEffect



-- 0
-- 1
-- 1
-- 0

Variables or values used to determine when useEffect should run



Specified as an array argument

Dependencies in useEffect

Empty dependency array ([]):

```
useEffect(() => {  
  fetch('https://api.npoint.io/d542b9ad99f501ab3dbf')  
    .then(response => response.json())  
    .then(data => {  
      console.log(data);  
      setFoods(data);  
    })  
    .catch(error => console.error  
      ('Error fetching users:', error));  
}, []); // Empty dependency array means this effect runs  
        //only once when the component mounts
```

Dependency array with values

```
const [count, setCount] = useState(0);  
useEffect(() => {  
  console.log(`Count updated: ${count}`);  
}, [count]); // Runs only when count changes  
// State update function  
const increment = () => setCount(count + 1);
```


Omitting dependency array

Effect runs on each render

```
useEffect(() => {  
  fetch('https://api.npoint.io/d542b9ad99f501ab3dbf')  
    .then(response => response.json())  
    .then(data => {  
      console.log(data);  
      setFoods(data);  
    })  
    .catch(error => console.error('Error fetching users:',  
      error));  
});
```

No array: execution before and after the component re-renders

Custom hook example

Reuse logic across components

```
function ToggleButton() {  
  const [isToggled, toggle] = UseToggle(false);  
  return (  
    <div>  
      <h1>Toggle Button</h1>  
      <button onClick={toggle}>  
        {isToggled ? 'ON' : 'OFF'}  
      </button>  
    </div>  
  );  
}  
export default ToggleButton;
```

Custom hook: UseToggle

```
function UseToggle(initialValue = false) {  
  const [value, setValue] = useState(initialValue);  
  const toggle = () => { setValue(!value);  
    };  
  return [value, toggle];  
}  
export default UseToggle
```

Custom hook: UseToggle

```
function ToggleButton() {  
  const [isToggled, toggle] =  
    UseToggle(false);  
  return (  
    <div>  
      <h1>Toggle Button</h1>  
      <button onClick={toggle}>  
        {isToggled ? 'ON' : 'OFF'}  
      </button>  
    </div>  
  );  
}  
export default ToggleButton;
```

```
function UseToggle(initialValue = false) {  
  const [value, setValue] =  
    useState(initialValue);  
  const toggle = () => {  
    setValue(!value);  
  };  
  return [value, toggle];  
}  
export default UseToggle
```

Custom hook

Before clicking on the button:

Toggle Button
☐ OFF

After clicking on the button:

Toggle Button
☒ ON

Recap

In this video, you learned that:

- `useEffect` is a React hook that allows you to perform side effects in functional components
- Side effects are actions that occur asynchronously
- In React's `useEffect` hook, dependencies refer to variables or values that determine when `useEffect` should run
- Custom hooks allow you to abstract complex logic that you can easily reuse across different components in your application