useEffect의 의존성 배열에 객체를 넣으면 생기는 일

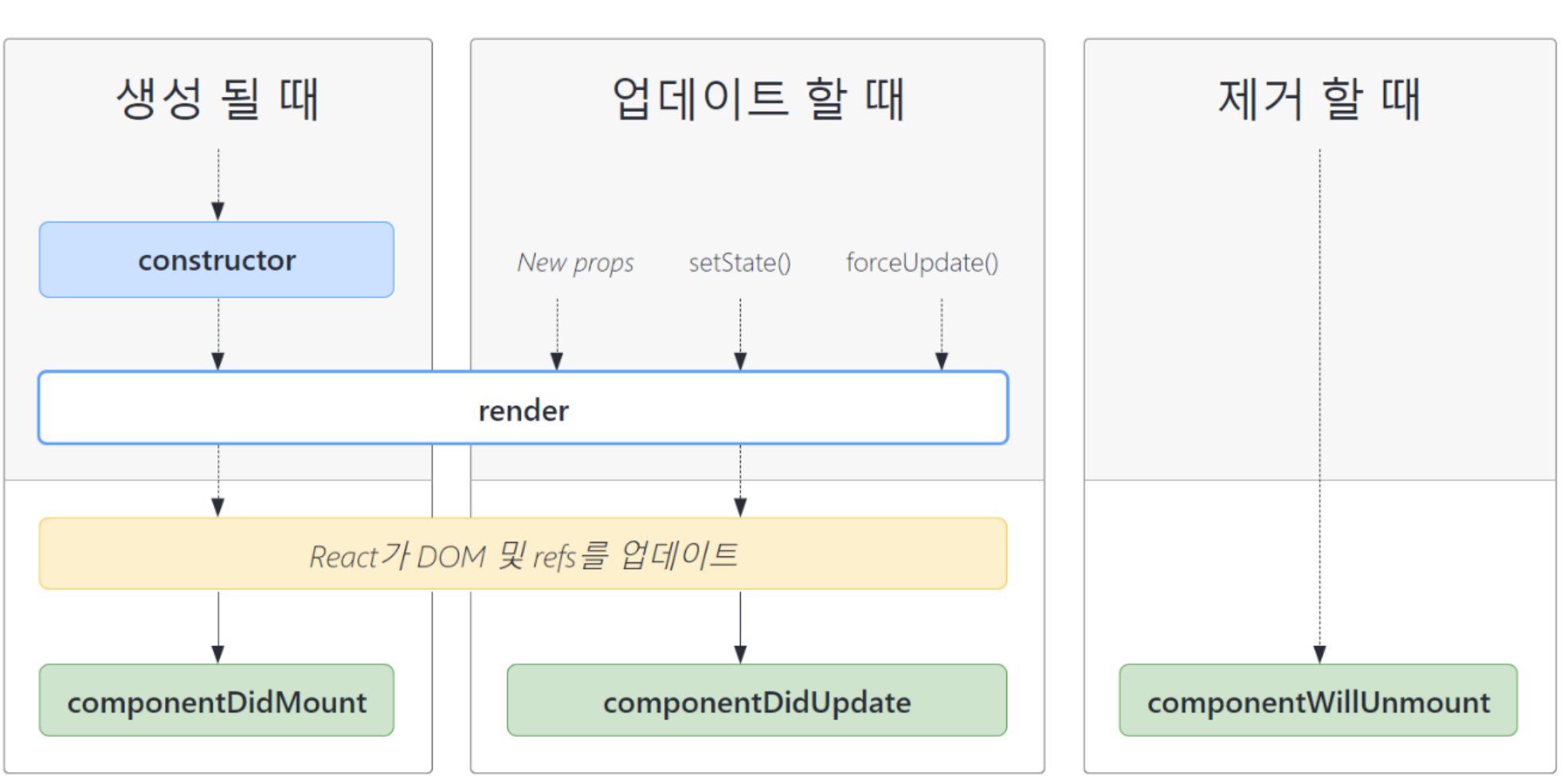
목차

useEffect가 뭘까?

함수형 컴포넌트에서도
부작용(Side Effect)을 처리할 수 있게 해주는 Hook

* 부작용 : 컴포넌트가 렌더링 되거나 상태가 업데이트 될 때 발생하는 작업

리액트의 Life Cycle



Life Cycle에 따라 부수 효과를 일으키는 생명 주기 메소드

시계를 만듭시다

index.js

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Clock />);
```

ReactDOM은 변경된 부분을 실제 DOM에 그려준다.

root.render에 Clock 컴포넌트 전달하면

Clock의 constructor 호출

Constructor 호출

```
export default class Clock extends React.Component {
  constructor (props) {
    super(props);
    this.state = {date: new Date()};
  }
```

```
export default class App extends Componen

  state = {
    date: new Date()
  };
```

constructor 에서는 state를 초기화 합니다.
(최신 버전 React는 클래스 필드 문법으로 초기화 가능)

시계는 시간 표시해야 하니까
현재 시각이 포함된 객체를 할당하여 지역 state를 초기화합니다.

render() 호출


```
render() {
  return (
    <div>
      <p>Hello, it is {this.state.date.toLocaleTimeString()}</p>
    </div>
  )
```

React는 Clock 컴포넌트의 render() 메서드를 호출합니다.
화면에 표시되어야 할 내용을 React가 알게 되었습니다.

이전 랜더링에서 생긴 DOM 트리와 현재 가상 DOM 트리를 비교하면서
바뀐 부분만 ReactDOM이 실제 DOM에 반영
Clock의 렌더링 출력값을 일치시키기 위해 DOM을 업데이트한다.

Hello, it is 오후 9:52:41

```
componentDidMount() {
  this.timerId = setInterval(() => this.tick(), 1000);
}
```

컴포넌트가 mount 되면

React는 componentDidMount() 생명주기 메서드를 호출

그 안에서는

setInterval로 브라우저한테 1초마다 tick() 실행하라고 시킴

```
tick() {
  this.setState({
    date: new Date()
  })
}
```

componentDidMount()

Called immediately after a component is mounted.
Setting state here will trigger re-rendering.

컴포넌트가 마운트된 직후 즉시 호출됨
여기에 setState 쓰면 리렌더링 될 듯 조심하셈

setState() 호출 덕분에 React는 state가 변경된 것을 인지한 뒤,
화면에 표시될 내용을 알아내기 위해
해당 컴포넌트의 render() 메서드를 다시 호출합니다.

render() 메서드 안에 변경된 state 값이 들어가고,
React는 이에 따라 DOM을 업데이트한다.

componentDidUpdate()

Called immediately after updating occurs.

Not called for the initial render.

업데이트 일어난 뒤에 즉시 호출됨

최초 렌더링에선 호출되지 않음

The snapshot is only present

if getSnapshotBeforeUpdate is present and returns non-null.

컴포넌트의 생명주기 메소드 중 하나인 getSnapshotBeforeUpdate가 존재하고

이 메소드가 null이 아닌 값 반환할 때만 스냅샷이 존재한다.

useEffect

어려운 메소드 그만 쓰자

useEffect

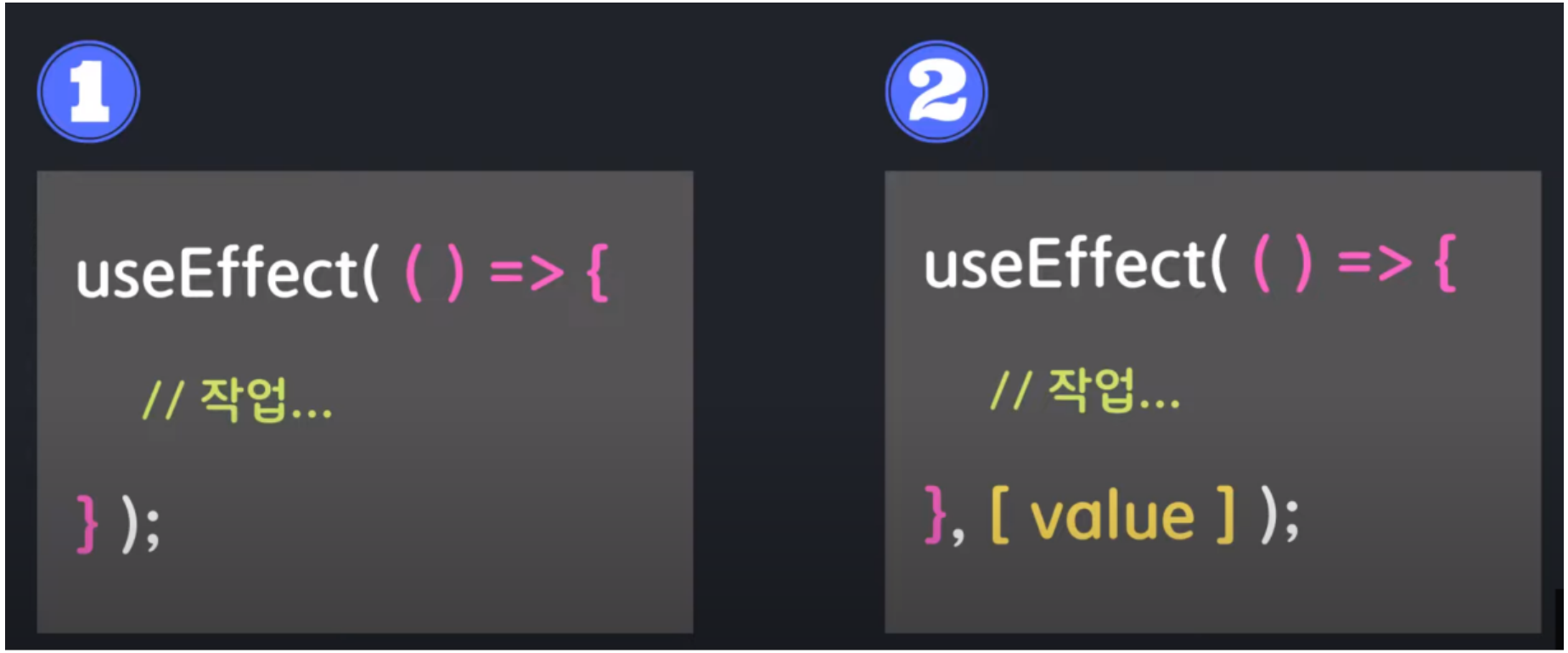
React 16.8버전에 새로 추가

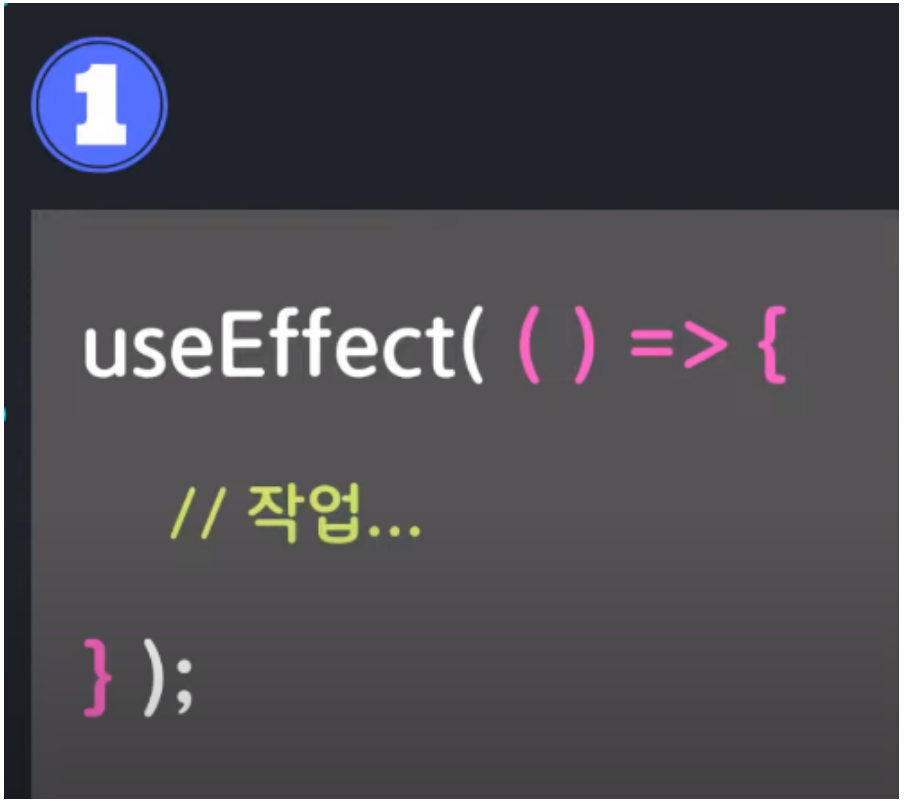
클래스형 컴포넌트를 작성하지 않아도
함수형 컴포넌트에서 side effect를 수행할 수 있다.

'useEffect를 componentDidMount와 componentDidUpdate,
componentWillUnmount가 합쳐진 것으로 생각해도 좋다.'

---

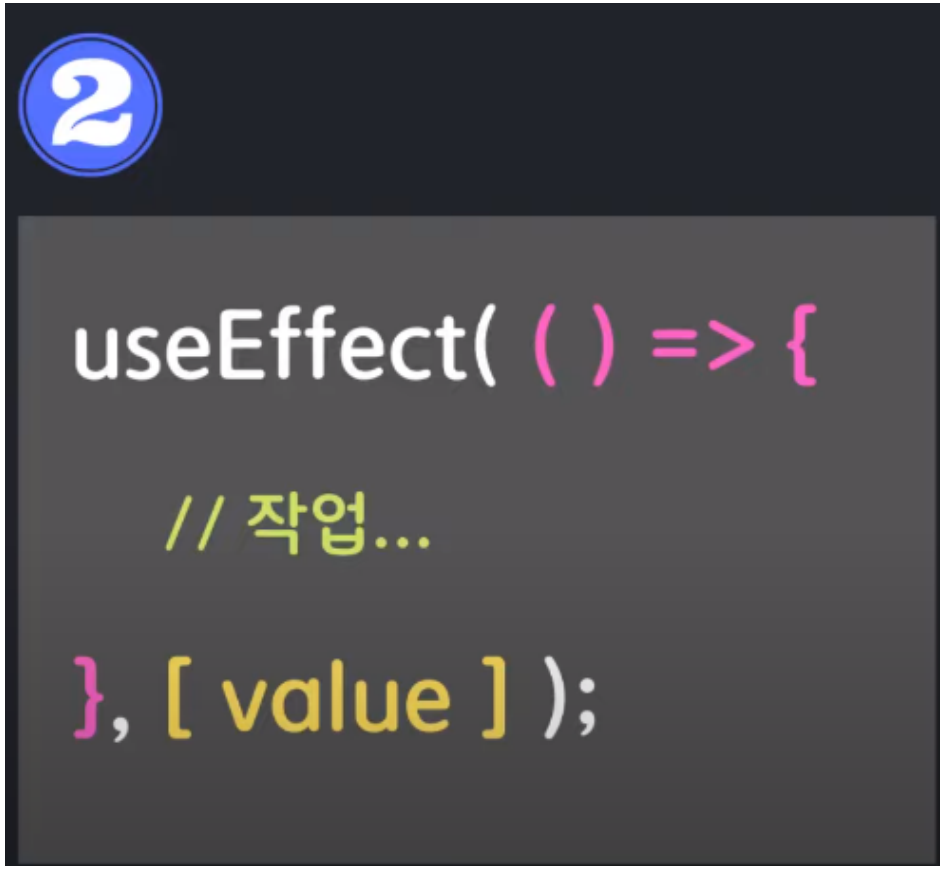
useEffect의 두가지 형태





렌더링 될 때마다 콜백함수 실행

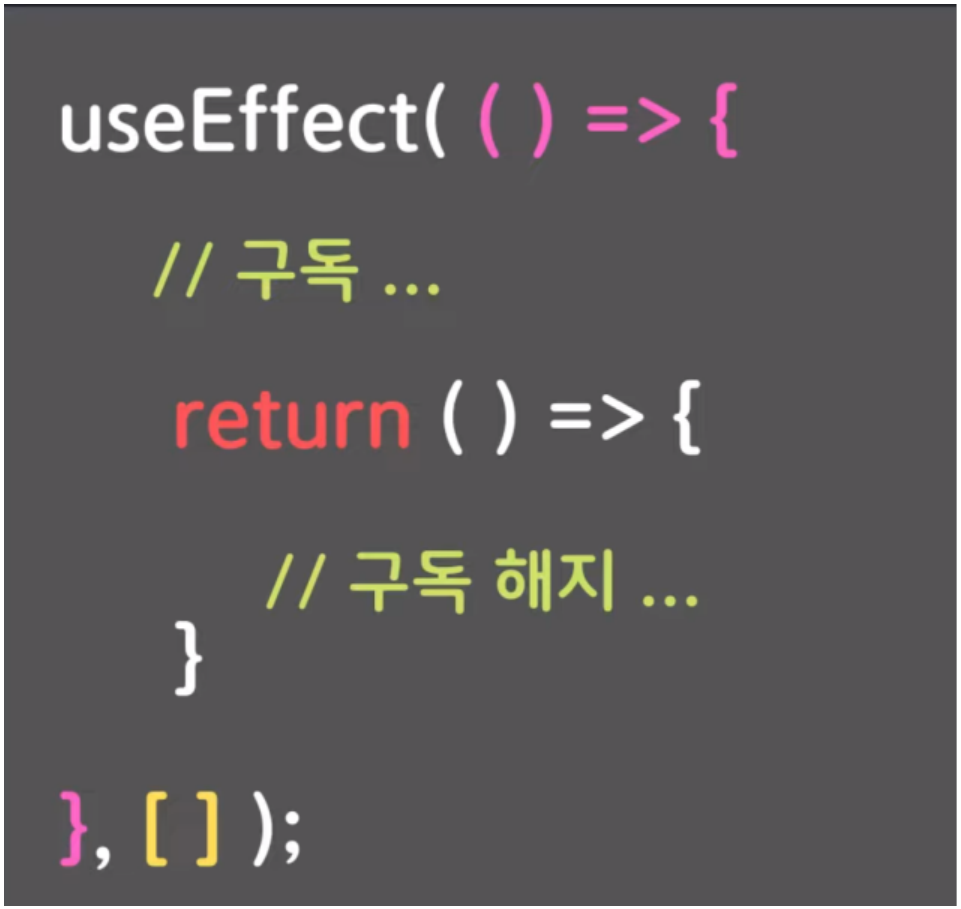
* 콜백함수: 다른 함수의 인자로 전달된 함수



의존성 배열
(dependency array)

초기 마운트 시에 실행되고

의존성 배열 안에 들어있는 값이
변할 때 콜백함수가 실행된다.



useEffect의 return 값으로
함수를 넣어주면
해당 컴포넌트가
unmount 될 때, 혹은
다음 렌더링 시 불릴
useEffect가 실행되기 이전에
실행됨

```jsx
import { useState, useEffect } from 'react'

const App = () => {
  const [point, setPoint] = useState(0);

  useEffect(() => {
    console.log("오디세이 이륙");
    setPoint((prev) => prev + 1);

    return () => {
      console.log("ㅂ2");
    }
  }, [point]);

  return (
    <div>
      <button>{point}</button>
    </div>
  )
}

export default App;
```

useEffect 사용해보기

```
오디세이 이륙    App.js:7
ㅂ2    App.js:11
오디세이 이륙    main.fa1ba4d22569dec…64.hot-update.js:28
ㅂ2    main.fa1ba4d22569dec…64.hot-update.js:31
오디세이 이륙    main.fa1ba4d22569dec…64.hot-update.js:28
ㅂ2    main.fa1ba4d22569dec…64.hot-update.js:31
오디세이 이륙    main.fa1ba4d22569dec…64.hot-update.js:28
ㅂ2    main.fa1ba4d22569dec…64.hot-update.js:31
오디세이 이륙    main.fa1ba4d22569dec…64.hot-update.js:28
ㅂ2    main.fa1ba4d22569dec…64.hot-update.js:31
오디세이 이륙    main.fa1ba4d22569dec…64.hot-update.js:28
ㅂ2    main.fa1ba4d22569dec…64.hot-update.js:31
오디세이 이륙    main.fa1ba4d22569dec…64.hot-update.js:28
ㅂ2    main.fa1ba4d22569dec…64.hot-update.js:31
오디세이 이륙    main.fa1ba4d22569dec…64.hot-update.js:28
ㅂ2    main.fa1ba4d22569dec…64.hot-update.js:31
오디세이 이륙    main.fa1ba4d22569dec…64.hot-update.js:28
ㅂ2    main.fa1ba4d22569dec…64.hot-update.js:31
오디세이 이륙    main.fa1ba4d22569dec…64.hot-update.js:28
ㅂ2    main.fa1ba4d22569dec…64.hot-update.js:31
오디세이 이륙    main.fa1ba4d22569dec…64.hot-update.js:28
ㅂ2    main.fa1ba4d22569dec…64.hot-update.js:31
오디세이 이륙    main.fa1ba4d22569dec…64.hot-update.js:28
ㅂ2    main.fa1ba4d22569dec…64.hot-update.js:31
```

useEffect의 의존성 배열에
원시 값이 아닌
객체를 집어넣으면?

자바스크립트에서 객체는 **힙(Heap)**에 저장됨
따라서 객체를 state로 등록하면
해당 객체의 **힙 메모리 주소를 참조**하게 된다.

setState를 통해 state를 변경해야
useEffect의 **의존성 배열에 설정된 값의 변화**를 감지하고,
useEffect가 실행된다.

setState는 새로운 상태를 생성하고 해당 상태로
업데이트하는 메소드인데,

setState로 객체를 변경하면
새로운 객체의 힙의 메모리 주소가 할당된다.

이렇게 복사해도 힙의 메모리 주소가 달라짐 => useEffect() 실행

```
function App() {
  const [obj, setObj] = useState({ name: 'John', age: 25

  const updateName = () => {
    setObj({ ...obj });
  };

  useEffect(() => {
    console.log("변경됨");
  }, [obj])

  return (
    <div>
      <button onClick={updateName}>Update Name</button>
    </div>
  );
}
```

해당 객체의 속성 변경이 아닌, 새로운 객체 생성

```
function App() {
  const [obj, setObj] = useState({ name: 'ssafy', age: 25 });

  const updateName = () => {
    setObj({ ...obj, age : 26});
  };

  useEffect(() => {
    console.log(" obj 변경됨");
  }, [obj])

  return (
    <div>
      <button onClick={updateName}>Update Name</button>
    </div>
  );
}
```

이것도 마찬가지

```
function App() {
  const [obj, setObj] = useState({ name: 'ssafy', age: 25 });

  const updateName = () => {
    setObj(Object.assign({}, obj));
  };

  useEffect(() => {
    console.log(" obj 변경됨");
  }, [obj])

  return (
    <div>
      <button onClick={updateName}>Update Name</button>
    </div>
  );
}
```

단, 그대로 obj를 set 하는 경우에는 useEffect가 실행되지 않음

```
function App() {
  const [obj, setObj] = useState({ name: 'ssafy', age: 25 });

  const updateName = () => {
    setObj(obj);
  };

  useEffect(() => {
    console.log(" obj 변경됨");
  }, [obj])

  return (
    <div>
      <button onClick={updateName}>Update Name</button>
    </div>
  );
}
```

참고문헌

React] useEffect( )로 Lifecycle이해하기 (feat.예제)

https://velog.io/@fltxld3/React-useEffect-로-Lifecycle이해하기-feat.일기장

React 공식문서

https://ko.legacy.reactjs.org/docs/react-component.html#componentdidupdate

React useEffect 의 dependency array

https://sgwanlee.medium.com/useeffect%EC%9D%98-dependency-array-ebd15f35403a