7. useState 를 통해 컴포넌트에서 바뀌는 값 관리하기

지금까지 우리가 리액트 컴포넌트를 만들 때는, 동적인 부분이 하나도 없었습니다. 값이 바뀌는 일이 없었죠. 이번에는 컴포넌트에서 보여줘야 하는 내용이 사용자 인터랙션에 따라 바뀌어야 할 때 어떻게 구현할 수 있는지에 대하여 알아보겠습니다.

리액트 16.8 이전 버전에서는 함수형 컴포넌트에서는 상태를 관리할 수 없었는데요, 리액트 16.8 에서 Hooks 라는 기능이 도입되면서 함수형 컴포넌트에서도 상태를 관리할 수 있게 되었습니다. 이번에는 useState 라는 함수를 사용해보게 되는데, 이게 바로 리액트의 Hooks 중 하나입니다.

정말 진부한 예제인, 버튼을 누르면 숫자가 바뀌는 Counter 컴포넌트를 만들어볼게요.

src 디렉터리에 Counter.js 를 다음과 같이 작성해보세요.

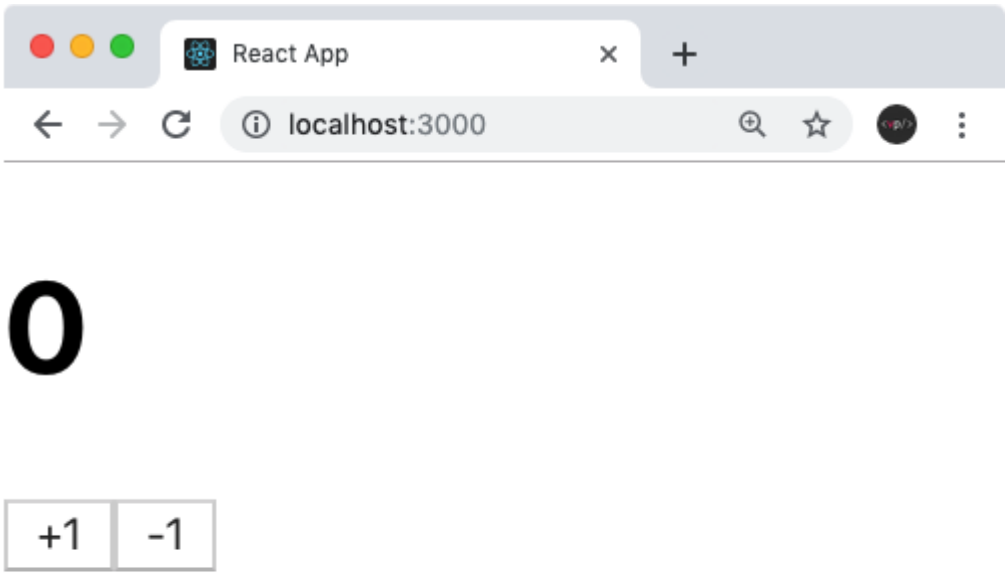
Counter.js

```
import React from 'react';

function Counter() {
  return (
    <div>
      <h1>0</h1>
      <button>+1</button>
      <button>-1</button>
    </div>
  );
}

export default Counter;
```

그 다음엔 App 에서 Counter 를 렌더링 해보세요.

App.js

```
import React from 'react';
import Counter from './Counter';

function App() {
  return (
    <Counter />
  );
}

export default App;
```



이런 UI 가 보여졌나요?

이벤트 설정

이제, Counter 에서 버튼이 클릭되는 이벤트가 발생 했을 때, 특정 함수가 호출되도록 설정을 해보겠습니다.

Counter 컴포넌트를 다음과 같이 수정해보세요.

Counter.js

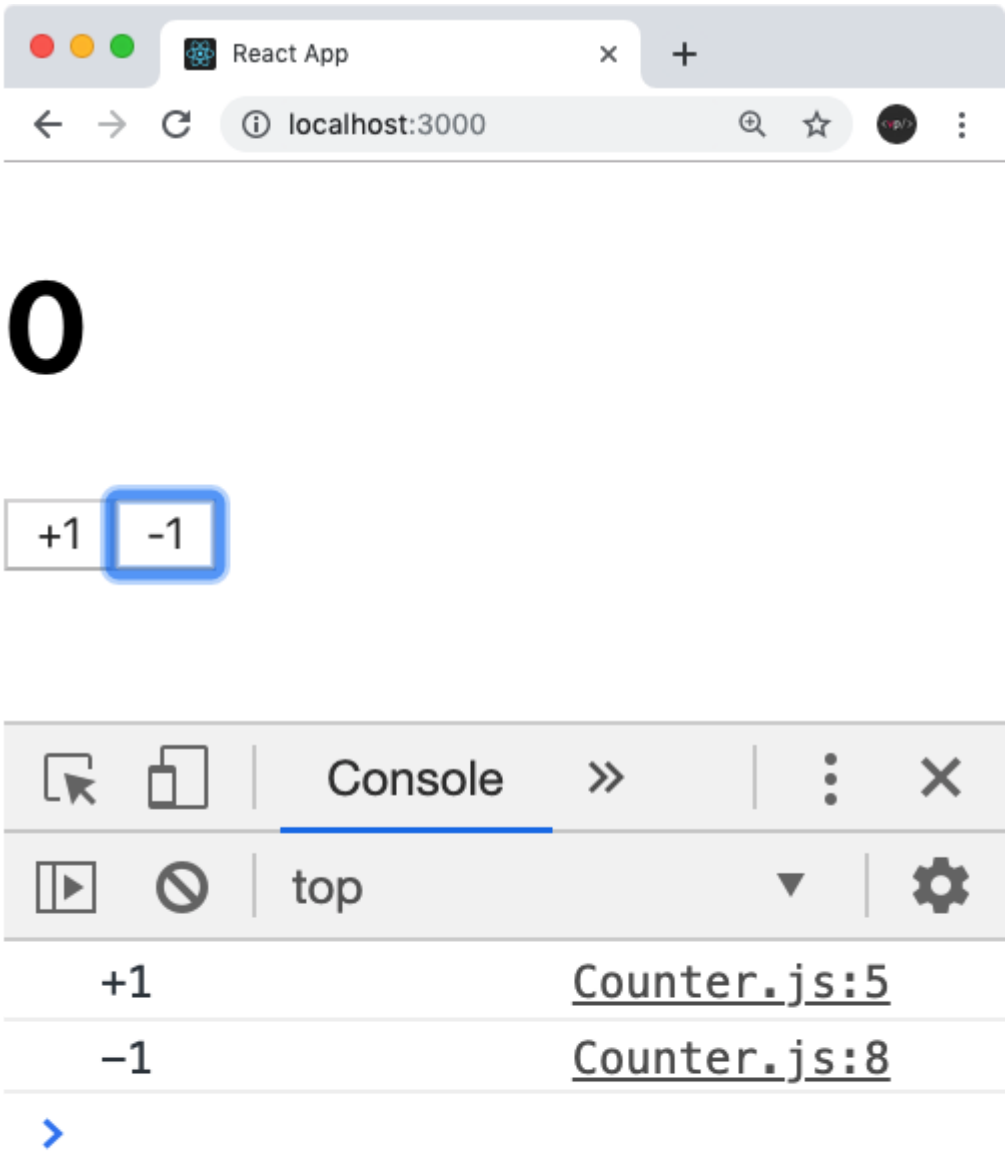
```
import React from 'react';

function Counter() {
  const onIncrease = () => {
    console.log('+1')
  }
  const onDecrease = () => {
    console.log('-1');
  }
  return (
    <div>
      <h1>0</h1>
      <button onClick={onIncrease}>+1</button>
      <button onClick={onDecrease}>-1</button>
    </div>
  );
}

export default Counter;
```

여기서 `onIncrease` 와 `onDecrease` 는 화살표 함수를 사용하여 구현을 해주었는데요, 화살표 함수에 대해서 잘 모르신다면 이 링크 를 참고하세요.
함수를 만들고, button 의 `onClick` 으로 각 함수를 연결해주었습니다. 리액트에서 엘리먼트에 이벤트를 설정해줄때에는 `on 이벤트이름={실행하고싶은함수}` 형태로 설정해주어야 합니다.
여기서 주의하셔야 하는 점은, 함수형태를 넣어주어야 하지, 함수를 다음과 같이 실행하시면 안됩니다.

```
onClick={onIncrease()}
```

이렇게 하면 렌더링되는 시점에서 함수가 호출되버리기 때문입니다. 이벤트를 설정할때에는 함수타입의 값을 넣어주어야 한다는 것, 주의해주세요.

코드를 저장 후 버튼들을 눌러보세요. 콘솔에 메시지들이 잘 출력되나요?



동적인 값 끼얹기, useState

컴포넌트에서 동적인 값을 상태(state)라고 부릅니다. 리액트에 `useState` 라는 함수가 있는데요, 이것을 사용하면 컴포넌트에서 상태를 관리 할 수 있습니다. Counter 컴포넌트를 다음과 같이 수정해보세요.

Counter.js

```
import React, { useState } from 'react';
```

```
function Counter() {
  const [number, setNumber] = useState(0);

  const onIncrease = () => {
    setNumber(number + 1);
  }

  const onDecrease = () => {
    setNumber(number - 1);
  }

  return (
    <div>
      <h1>{number}</h1>
      <button onClick={onIncrease}>+1</button>
      <button onClick={onDecrease}>-1</button>
    </div>
  );
}

export default Counter;
```

```
import React, { useState } from 'react';
```

이 코드는 리액트 패키지에서 `useState` 라는 함수를 불러와줍니다.

```
const [number, setNumber] = useState(0);
```

`useState` 를 사용 할 때에는 상태의 기본값을 파라미터로 넣어서 호출해줍니다.
이 함수를 호출해주면 배열이 반환되는데요, 여기서 첫번째 원소는 현재 상태,
두번째 원소는 Setter 함수입니다.
원래는 다음과 같이 해야하지만,

```
const numberState = useState(0);
const number = numberState[0];
const setNumber = numberState[1];
```

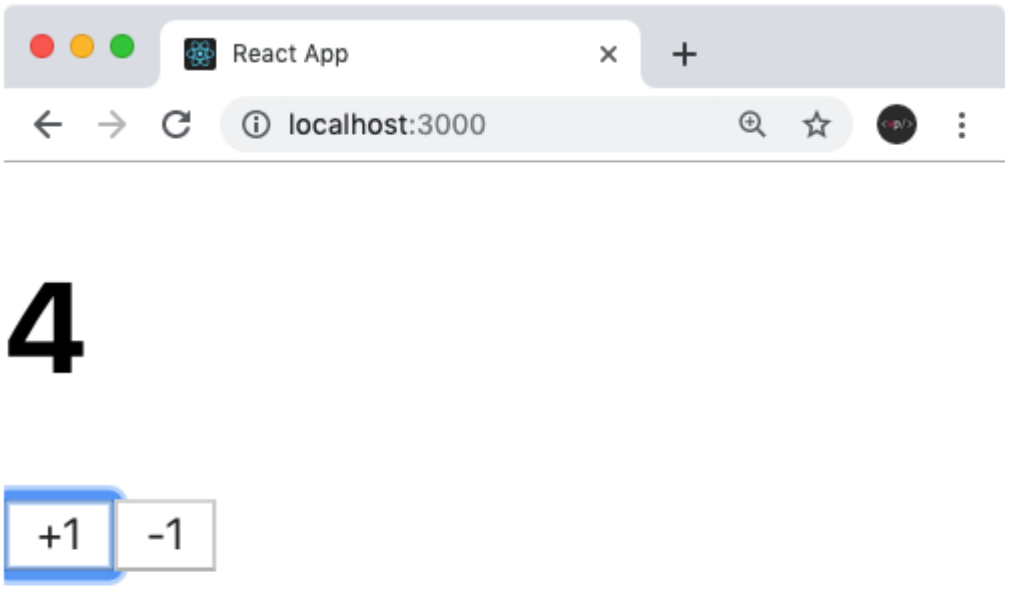
배열 비구조화 할당을 통하여 각 원소를 추출해준것입니다.

```
  const onIncrease = () => {
    setNumber(number + 1);
  }

  const onDecrease = () => {
    setNumber(number - 1);
  }
```

Setter 함수는 파라미터로 전달 받은 값을 최신 상태로 설정해줍니다.

```
    <h1>{number}</h1>
```

h1 태그에서는 이제 0 이 아닌 `{number}` 값을 보여주어야 합니다.
코드를 다 수정 후, 버튼들을 눌러보세요. 숫자가 잘 바뀌고 있나요?



함수형 업데이트

지금은 Setter 함수를 사용 할 때, 업데이트 하고 싶은 새로운 값을 파라미터로 넣어주고 있는데요, 그 대신에 기존 값을 어떻게 업데이트 할 지에 대한 함수를 등록하는 방식으로도 값을 업데이트 할 수 있습니다.

Counter 컴포넌트를 다음과 같이 수정해보세요.

Counter.js

```
import React, { useState } from 'react';

function Counter() {
  const [number, setNumber] = useState(0);

  const onIncrease = () => {
    setNumber(prevNumber => prevNumber + 1);
  }

  const onDecrease = () => {
    setNumber(prevNumber => prevNumber - 1);
  }

  return (
    <div>
      <h1>{number}</h1>
      <button onClick={onIncrease}>+1</button>
      <button onClick={onDecrease}>-1</button>
```

```
    </div>
  );
}

export default Counter;
```

`onIncrease` 와 `onDecrease` 에서 `setNumber` 를 사용 할 때 그 다음 상태를 파라미터로 넣어준것이 아니라, 값을 업데이트 하는 함수를 파라미터로 넣어주었습니다.

함수형 업데이트는 주로 나중에 컴포넌트를 최적화를 하게 될 때 사용하게 됩니다. 지금 당장은 함수형 업데이트란게 있는 것 정도만 이해해두시면 충분합니다. 이게 왜 최적화랑 관련이 되어있는지는 나중에 알아보도록 할게요.