

React

◆ state

정수아

Contents

01 state란?

02 함수형 컴포넌트의 useState

03 state와 이벤트 연동

04 state 값 변경하기

05 클래스형 컴포넌트의 state

06 state 끌어올리기



01

state란?

state란?

❖ state

- 값을 저장하거나 변경할 수 있는 객체
- 컴포넌트 내부에서 바뀔 수 있는 값을 의미
- 주로 버튼 클릭과 같은 이벤트와 함께 사용함
- props와 차이점
 - props는 부모 컴포넌트가 설정한 값을 전달받아 읽기 전용으로만 사용할 수 있음
 - props는 컴포넌트 내부에서는 값을 직접 변경할 수 없음

[실습] props 값 변경해보기

❖ App.js

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

function App() {
  return (
    <>
      <ChangePropsValue name="S00A"/>
    </>
  );
};

export default App;
```

[실습] props 값 변경해보기

❖ ChangePropsValue.js

```
import React from 'react';

function ChangePropsValue(props) {
  let name = props.name;

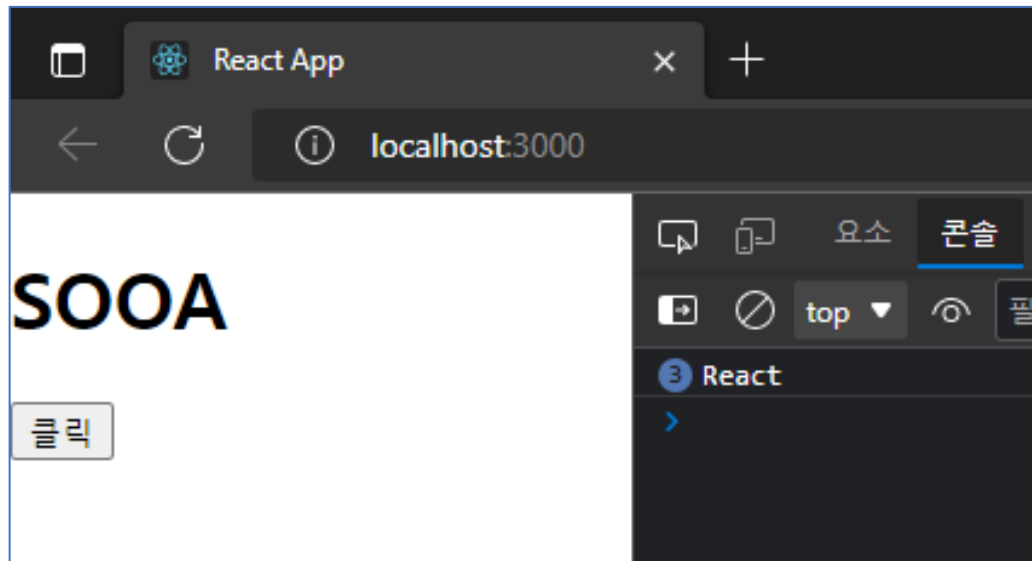
  function changeName(){
    name = "React";
  }

  return (
    <div>
      <h1>{props.name}</h1>
      <button onClick={changeName}>클릭</button>
    </div>
  );
};

export default ChangePropsValue;
```

[실습] props 값 변경해보기

❖ 실행 결과



state란?

❖ state 종류

- 클래스형 컴포넌트의 state 속성
- 함수형 컴포넌트의 useState 함수



02

함수형 컴포넌트의 useState

useState 사용하기

❖ 배열 비구조화 할당

- 배열 안에 들어 있는 값을 쉽게 추출할 수 있도록 해주는 문법
- array 안에 있는 값을 변수 one과 two에 저장

```
const array = [1, 2];  
const one = array[0];  
const two = array[1];
```

- 배열 비구조화 할당을 사용하여 저장

```
const array = [1, 2];  
const [one, two] = array;
```

useState 사용하기

❖ useState 함수

```
const [value, setValue] = useState(값);
```

- 값의 형태
 - 숫자, 문자열, 객체, 배열
- 리턴 값
 - 배열
 - 첫 번째 원소 : 현재 상태
 - 두 번째 원소 : 상태를 바꾸어 주는 함수

useState 사용하기

❖ useState 함수

```
const [인사, 인사변경] = useState('안녕하세요');
```

- 값의 형태
 - 문자열
- 리턴 값
 - 배열
 - 첫 번째 원소 : '안녕하세요'
 - 두 번째 원소 : '안녕하세요'를 다른 값으로 바꾸어 주는 함수

[실습] useState 값 가져오기 - 문자열

❖ App.js

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

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

export default App;
```

[실습] useState 값 가져오기 - 문자열

❖ Hello.js

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

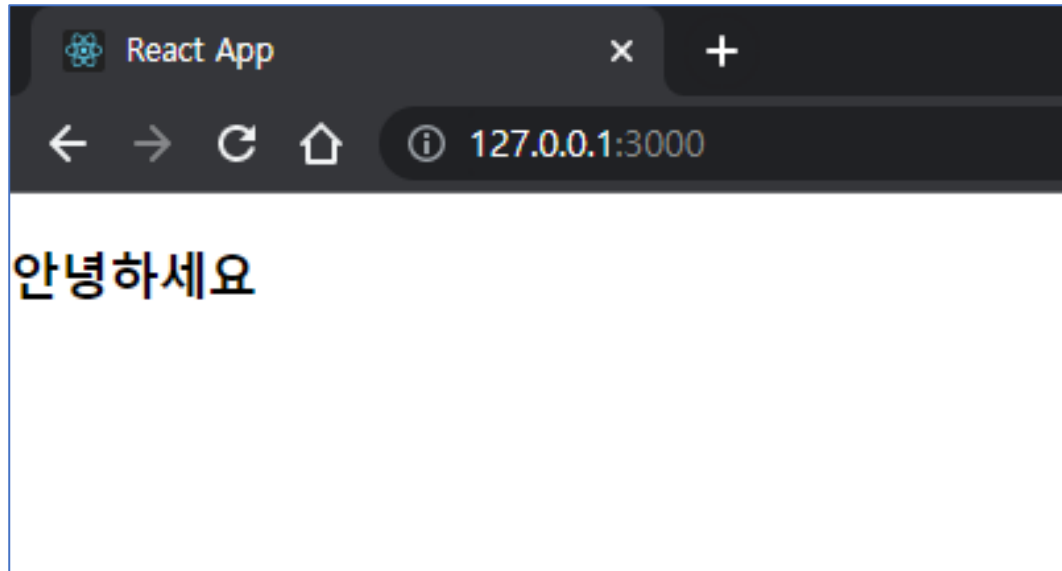
function Hello(){
  const [인사, 인사변경] = useState('안녕하세요');

  return (
    <div>
      <h3>{인사}</h3>
    </div>
  );
}

export default Hello;
```

[실습] useState 값 가져오기 - 문자열

❖ 실행 결과



[실습] useState 값 가져오기 - 배열

❖ Hello.js

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

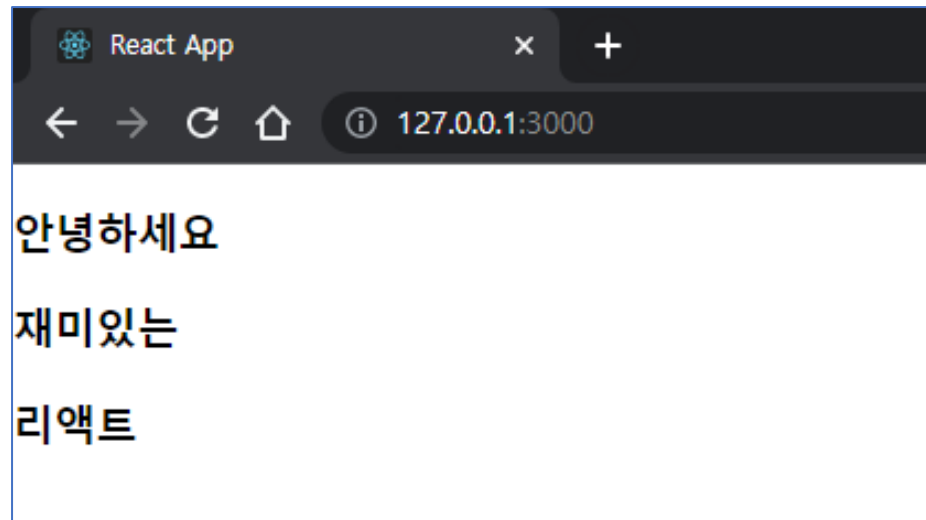
function Hello(){
  const [인사, 인사변경] = useState(['안녕하세요', '재미있는', '리액트']);

  return (
    <div>
      <h3>{인사[0]}</h3>
      <h3>{인사[1]}</h3>
      <h3>{인사[2]}</h3>
    </div>
  );
}

export default Hello;
```


[실습] useState 값 가져오기 - 배열

❖ 실행 결과



[실습] 한 컴포넌트에서 useState 여러 번 사용

❖ App.js

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

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

export default App;
```

[실습] 한 컴포넌트에서 useState 여러 번 사용

❖ ChangeFont.js

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

function ChangeFont(){
  const [value, setValue] = useState('안녕하세요');
  const [font, setFont] = useState('50px');

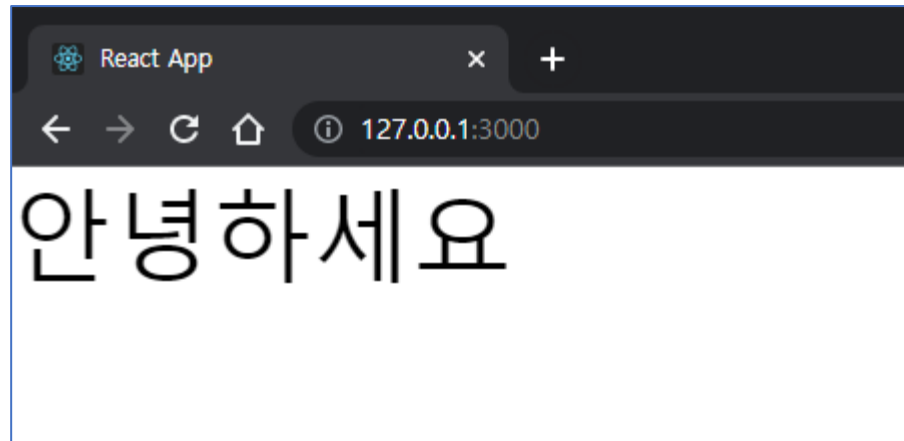
  const fstyle = { fontSize : font }

  return (
    <div>
      <div style={fstyle}>{value}</div>
    </div>
  );
};

export default ChangeFont;
```

[실습] 한 컴포넌트에서 useState 여러 번 사용

❖ 실행 결과





03

state와 이벤트 연동

state와 클릭 이벤트 연동

❖ state와 클릭 이벤트 연동

```
<button onClick={클릭할때_실행할_함수}>클릭</button>
```

- 함수 구현

```
function 클릭할때_실행할_함수(){  
  // 실행할 내용  
}
```

[실습] state와 클릭 이벤트 연동

❖ App.js

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

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

export default App;
```

[실습] state와 클릭 이벤트 연동

❖ EventClick.js

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

function EventClick(){
  const [value, setValue] = useState('안녕하세요');

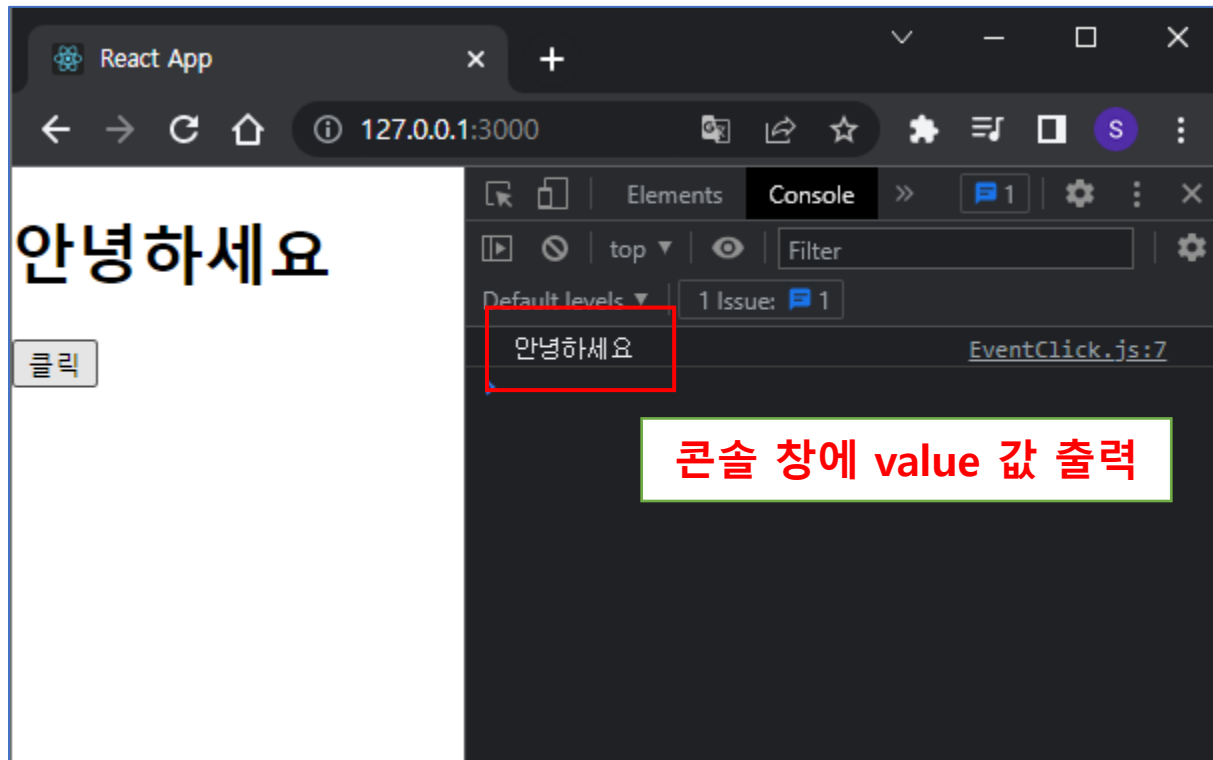
  function printConsole(){
    console.log(value);
  }

  return (
    <div>
      <h1>{value}</h1>
      <button onClick={printConsole}>클릭</button>
    </div>
  );
};

export default EventClick;
```


[실습] state와 클릭 이벤트 연동

❖ 실행 결과





04

state 값 변경하기

state 값 변경하기

❖ state 값 변경

- useState 함수 리턴값의 두 번째 원소를 이용

```
const [value, setValue] = useState('안녕하세요');
```

- 아래와 같은 방식으로 값 변경 불가능

```
value = '반가워요';
```

- setValue() 함수를 이용하여 값 변경

```
setValue('반가워요');
```

[실습] state 값 변경하기

❖ App.js

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

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

export default App;
```

[실습] state 값 변경하기

❖ ChangeValue.js

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

function ChangeValue(){
  const [value, setValue] = useState('안녕하세요');

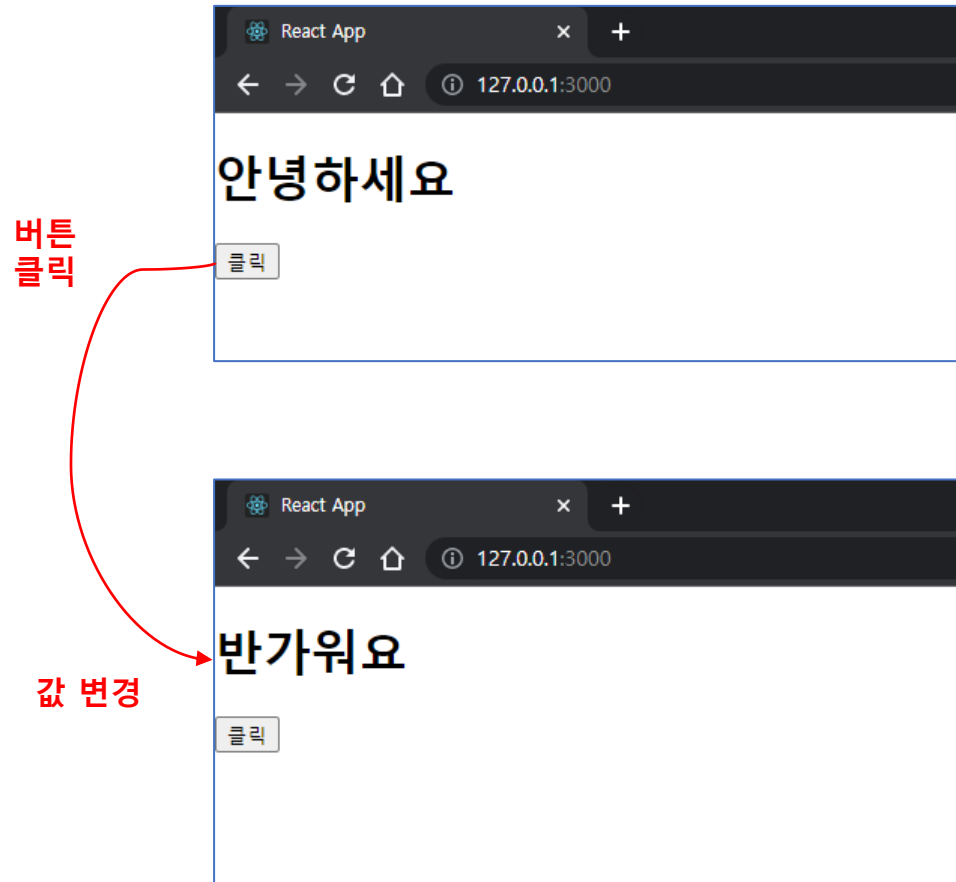
  function chageValue(){
    setValue('반가워요');
  }

  return (
    <div>
      <h1>{value}</h1>
      <button onClick={chageValue}>클릭</button>
    </div>
  );
};

export default ChangeValue;
```

[실습] state 값 변경하기

❖ 실행 결과



state 값 변경하기

❖ 객체 또는 배열의 state 값 변경

- 객체 또는 배열의 복사본을 만들어 값을 업데이트한 후, 복사본의 상태를 useState() 함수를 통해 업데이트

```
const [value, setValue] = useState({a:1, b:1});
```

- 아래와 같은 방식으로 값 변경하면 안됨

```
value.b = 2;
```

state 값 변경하기

❖ 객체/배열에 대한 사본 만들기

- spread 연산자(...) 사용
 - 객체/배열의 기존 내용을 변경하지 않고도 새로운 객체/배열을 생성할 수 있음
- 예) 객체 사본 만들기

```
const object = { a : 1, b : 2, c : 3 };

// object 객체의 사본 생성 후, b값만 변경
const copyObject = { ...object, b : 50 };

console.log(copyObject); // { a : 1, b : 50, c : 3 }
```


[실습] spread 연산자

❖ Spread.js

```
import React from 'react';

function Spread(){
  const person1 = { name : 'sooa' };
  const person2 = { name : 'sooa', age : 20 };
  const person3 = { name : 'sooa', age : 20, region : 'seoul' };

  return (
    <div>
      <h1>{JSON.stringify(person1)}</h1>
      <h1>{JSON.stringify(person2)}</h1>
      <h1>{JSON.stringify(person3)}</h1>
    </div>
  );
}

export default Spread;
```

[실습] spread 연산자

person1 →

name : 'sooa'

person2 →

name : 'sooa'

age : 20

person3 →

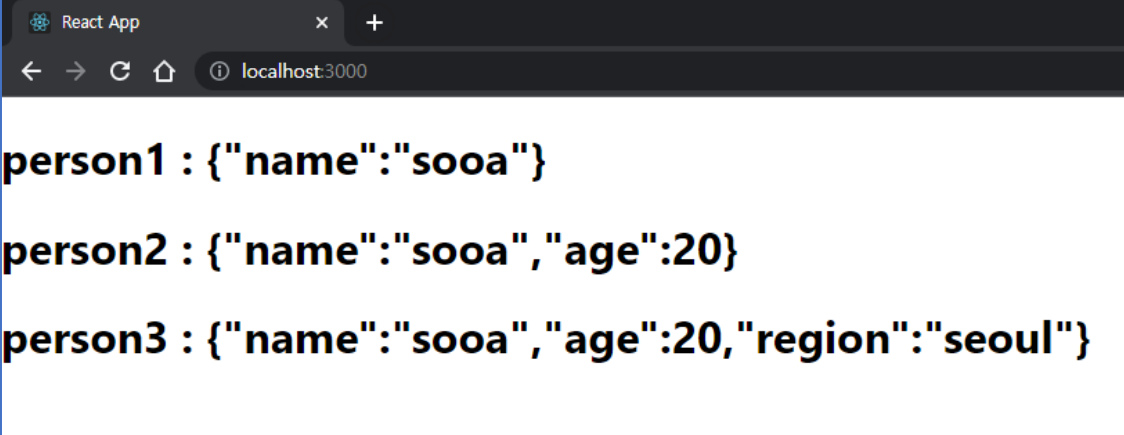
name : 'sooa'

age : 20

region : 'seoul'

[실습] spread 연산자

❖ 실행 결과



```
person1 : {"name":"sooa"}  
person2 : {"name":"sooa","age":20}  
person3 : {"name":"sooa","age":20,"region":"seoul"}
```

[실습] spread 연산자

❖ Spread.js 수정

```
import React from 'react';

function Spread(){
  const person1 = { name : 'sooa' };

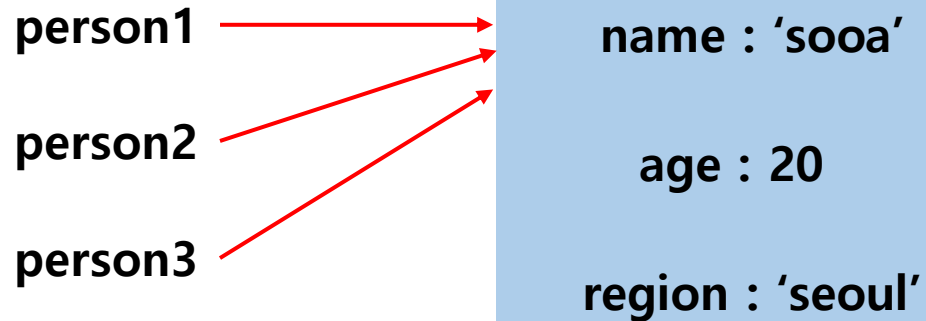
  const person2 = person1;
  person2.age = 20;

  const person3 = person2;
  person3.region = 'seoul';

  return (
    <div>
      <h1>{JSON.stringify(person1)}</h1>
      <h1>{JSON.stringify(person2)}</h1>
      <h1>{JSON.stringify(person3)}</h1>
    </div>
  );
}

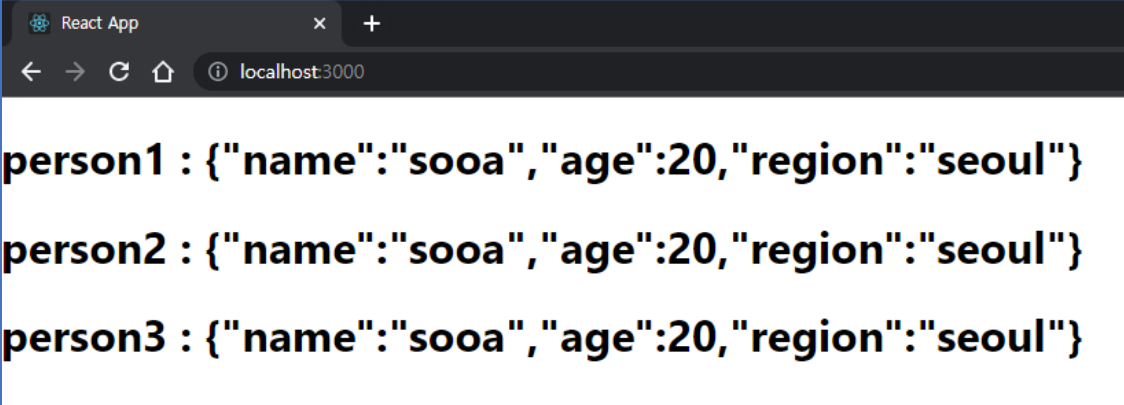
export default Spread;
```

[실습] spread 연산자



[실습] spread 연산자

❖ 실행 결과



```
person1 : {"name":"sooa","age":20,"region":"seoul"}
person2 : {"name":"sooa","age":20,"region":"seoul"}
person3 : {"name":"sooa","age":20,"region":"seoul"}
```

[실습] spread 연산자

❖ Spread.js 수정

```
import React from 'react';

function Spread(){
  const person1 = { name : 'sooa' };
  const person2 = { ...person1, age : 20 };
  const person3 = { ...person2, region : 'seoul' };

  return (
    <div>
      <h1>{JSON.stringify(person1)}</h1>
      <h1>{JSON.stringify(person2)}</h1>
      <h1>{JSON.stringify(person3)}</h1>
    </div>
  );
}

export default Spread;
```

[실습] spread 연산자

person1 →

name : 'sooa'

person2 →

name : 'sooa'

age : 20

person3 →

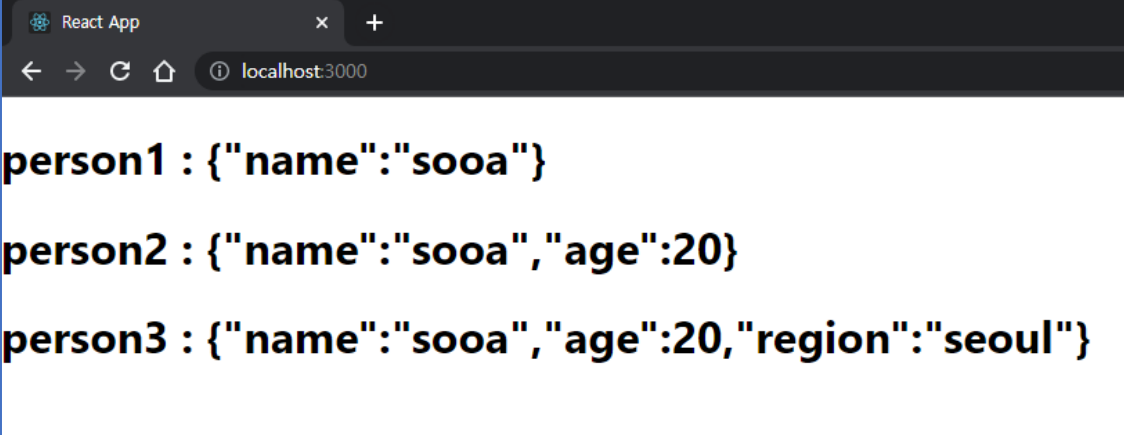
name : 'sooa'

age : 20

region : 'seoul'

[실습] spread 연산자

❖ 실행 결과



```
person1 : {"name":"sooa"}  
person2 : {"name":"sooa","age":20}  
person3 : {"name":"sooa","age":20,"region":"seoul"}
```

[실습] spread 연산자

❖ Spread.js 수정

```
import React from 'react';

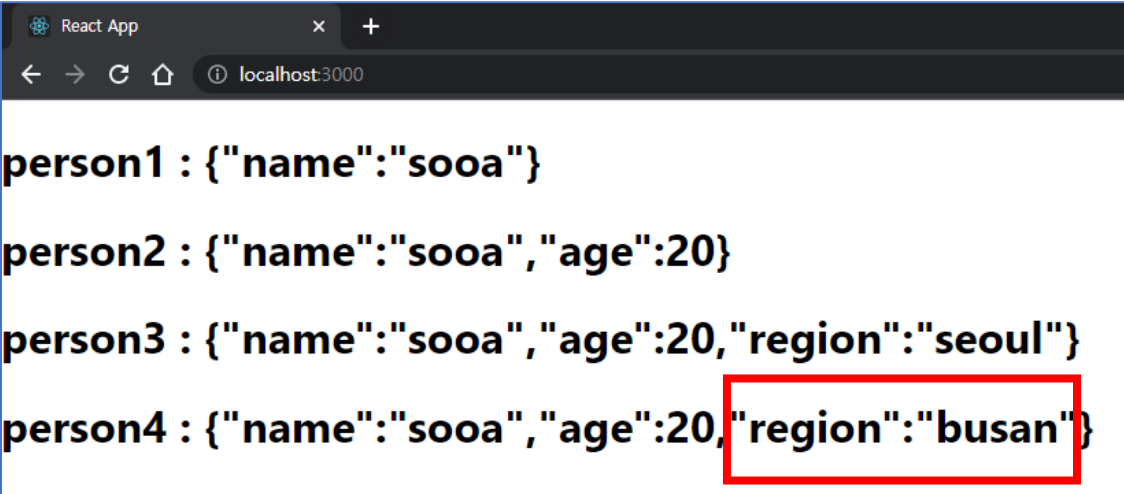
function Spread(){
  const person1 = { name : 'sooa' };
  const person2 = { ...person1, age : 20 };
  const person3 = { ...person2, region : 'seoul' };
  const person4 = { ...person3, region : 'busan' };

  return (
    <div>
      <h1>{JSON.stringify(person1)}</h1>
      <h1>{JSON.stringify(person2)}</h1>
      <h1>{JSON.stringify(person3)}</h1>
      <h1>{JSON.stringify(person4)}</h1>
    </div>
  );
}

export default Spread;
```

[실습] spread 연산자

❖ 실행 결과



A screenshot of a web browser window titled "React App" showing a JSON array of four person objects. The browser's address bar shows "localhost:3000". The JSON data is as follows:

```
person1 : {"name":"sooa"}  
person2 : {"name":"sooa","age":20}  
person3 : {"name":"sooa","age":20,"region":"seoul"}  
person4 : {"name":"sooa","age":20,"region":"busan"}
```

The "region" property value "busan" for person4 is highlighted with a red rectangular box.

[실습] spread 연산자

❖ Spread.js 수정

```
import React from 'react';

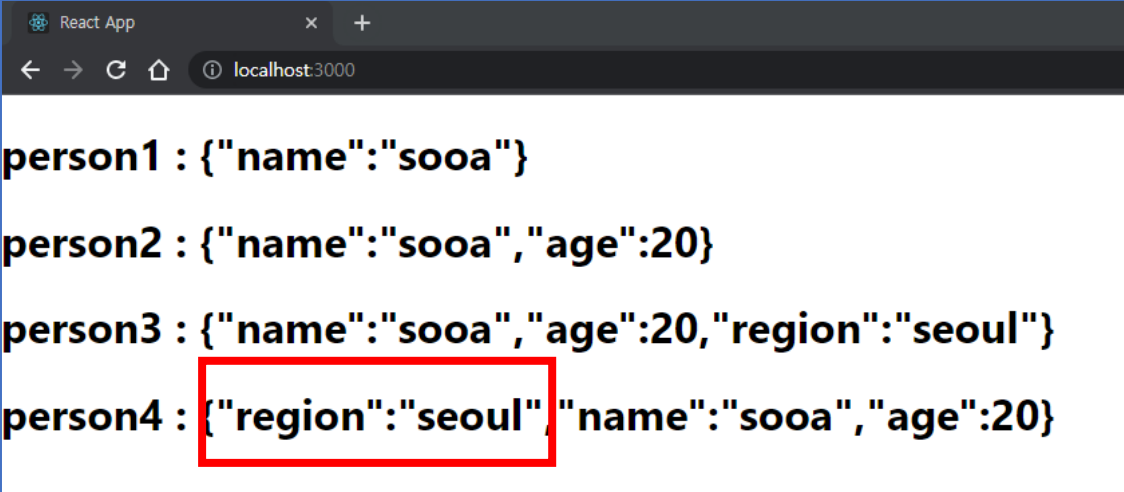
function Spread(){
  const person1 = { name : 'sooa' };
  const person2 = { ...person1, age : 20 };
  const person3 = { ...person2, region : 'seoul' };
  const person4 = { region : 'busan', ...person3 };

  return (
    <div>
      <h1>{JSON.stringify(person1)}</h1>
      <h1>{JSON.stringify(person2)}</h1>
      <h1>{JSON.stringify(person3)}</h1>
      <h1>{JSON.stringify(person4)}</h1>
    </div>
  );
}

export default Spread;
```

[실습] spread 연산자

❖ 실행 결과



```
person1 : {"name":"sooa"}  
person2 : {"name":"sooa","age":20}  
person3 : {"name":"sooa","age":20,"region":"seoul"}  
person4 : {"region":"seoul","name":"sooa","age":20}
```

[실습] state 값 변경하기 - 배열

❖ App.js

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

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

export default App;
```

[실습] state 값 변경하기 - 배열

❖ ChangeArray.js

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

function ChangeArray() {
  const [value, setValue] = useState(['안녕', '하이']);

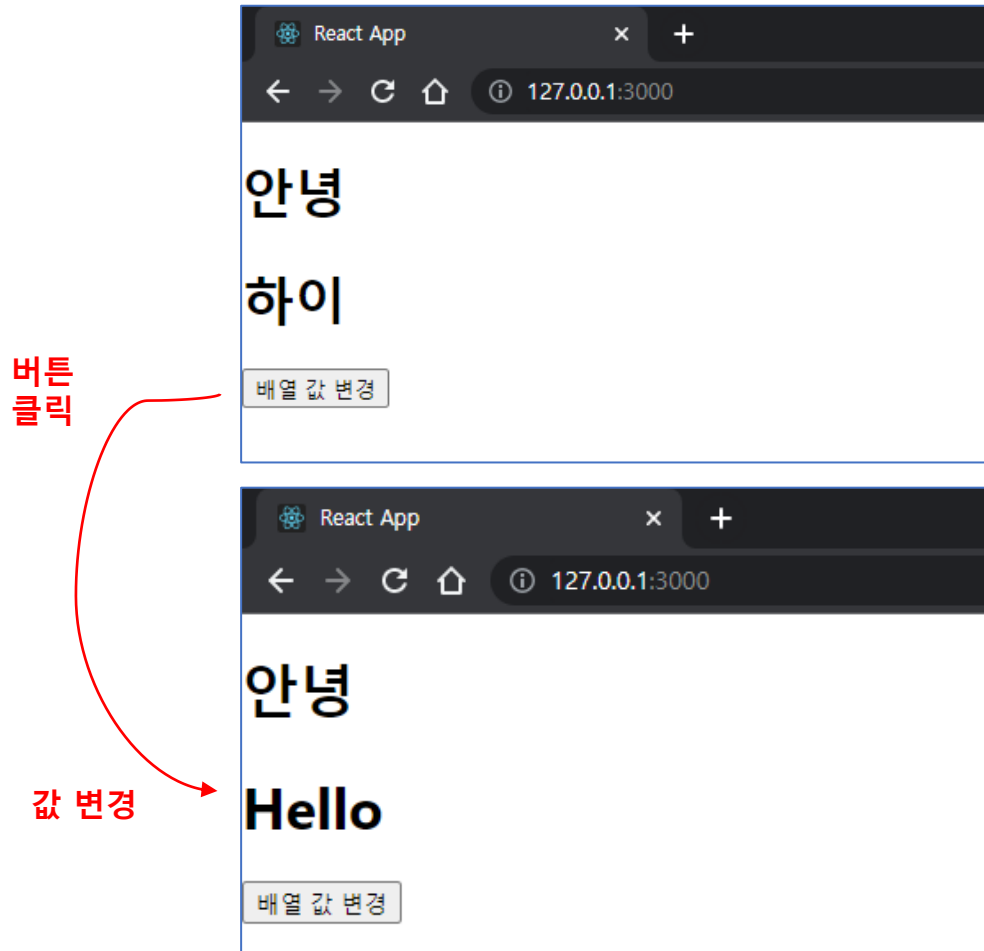
  function changeArr(){
    var cValue = [...value];
    cValue[1] = "Hello";
    setValue(cValue);
  }

  return (
    <div>
      <h1>{value[0]}</h1>
      <h1>{value[1]}</h1>
      <button onClick={changeArr}>배열 값 변경</button>
    </div>
  );
};

export default ChangeArray;
```

[실습] state 값 변경하기 - 배열

❖ 실행 결과





05

클래스형 컴포넌트의 state

state 속성 사용하기

❖ App.js

```
import React, { Component } from 'react';
import ClassState from './ClassState';

class App extends Component {
  render() {
    return (
      <div>
        <ClassState />
      </div>
    );
  }
}

export default App;
```

state 속성 사용하기

❖ ClassState.js

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

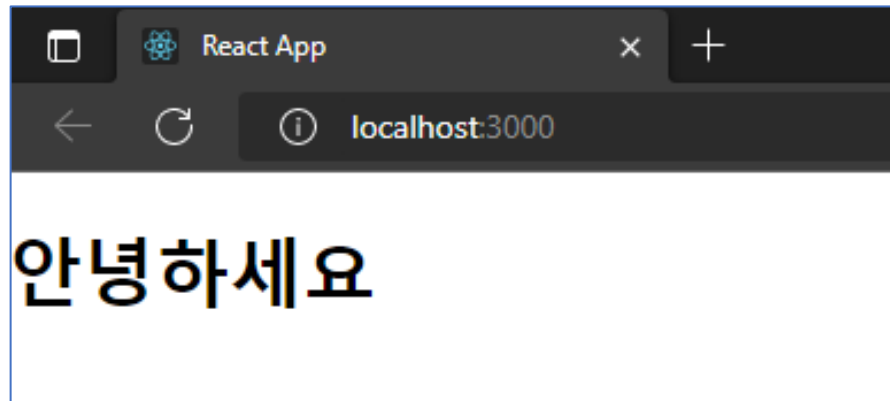
class ClassState extends Component {
  constructor(props){
    super(props);

    this.state = {
      인사 : '안녕하세요'
    };
  }
  render() {
    return (
      <div>
        <h1>{this.state.인사}</h1>
      </div>
    );
  }
}

export default ClassState;
```

state 속성 사용하기

❖ 실행 결과



state 값 변경하기

❖ ClassState.js

```
// 생략
class ClassState extends Component {

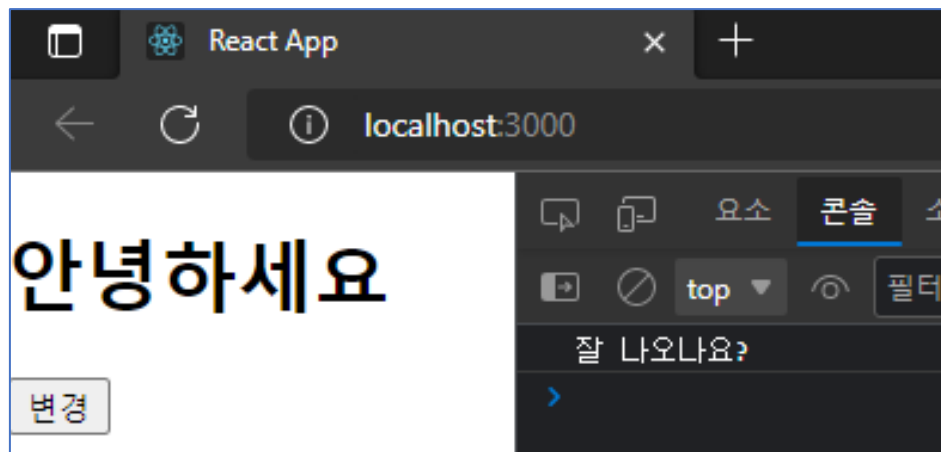
  // 생략

  changeMsg = () => {
    console.log("잘 나오나요?");
  }

  render() {
    return (
      <div>
        <h1>{this.state.인사}</h1>
        <button onClick={this.changeMsg}>변경</button>
      </div>
    );
  }
}
// 생략
```

state 값 변경하기

❖ 실행 결과



state 값 변경하기

❖ ClassState.js

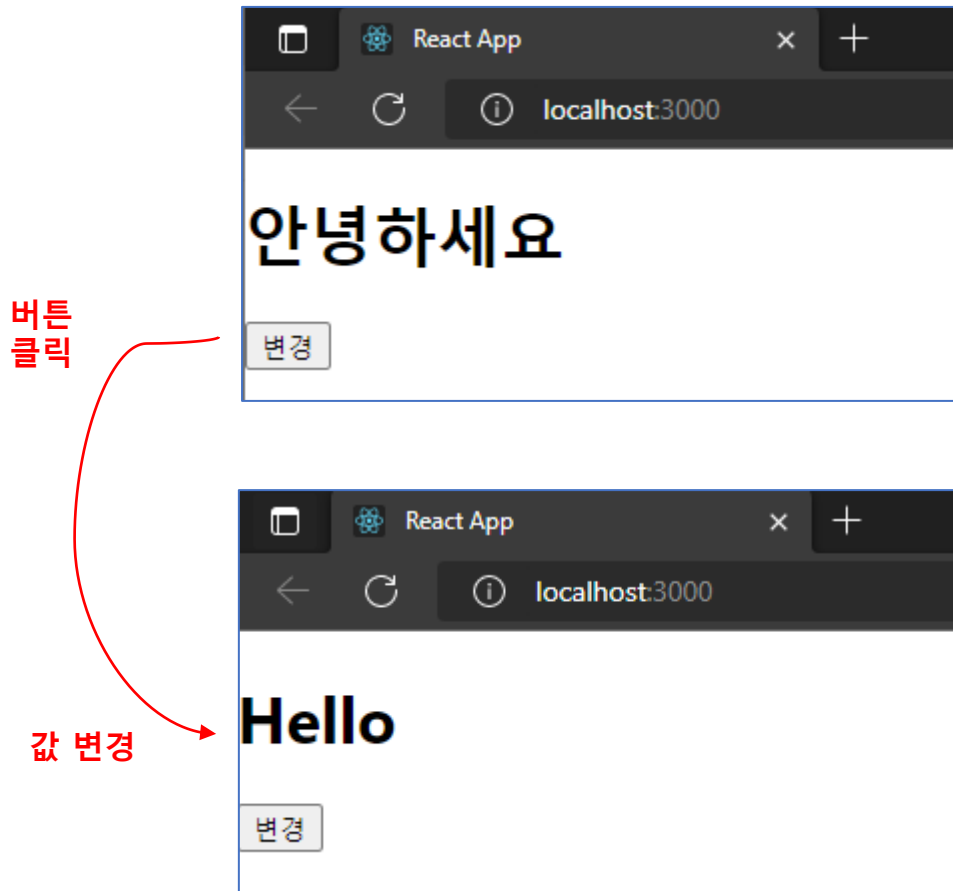
```
// 생략
class ClassState extends Component {
  state = {
    인사 : '안녕하세요'
  };

  changeMsg = () => {
    this.setState({ 인사 : 'Hello' });
  }

  render() {
    return (
      <div>
        <h1>{this.state.인사}</h1>
        <button onClick={this.changeMsg}>변경</button>
      </div>
    );
  }
}
// 생략
```

state 값 변경하기

❖ 실행 결과





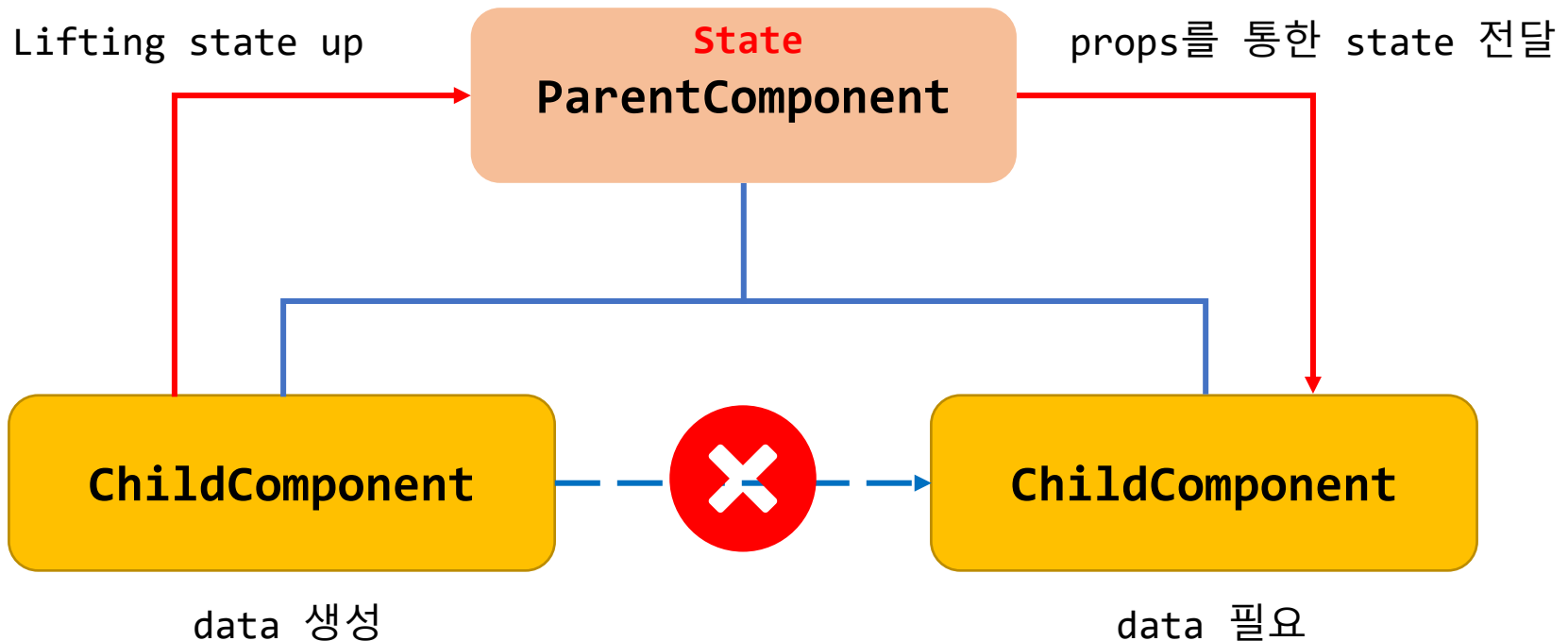
06

state 끌어올리기

state 끌어올리기(Lifting State Up)

❖ 형제 컴포넌트 간 데이터 공유

- 해당 값을 필요로 하는 컴포넌트 간의 가장 가까운 공통 조상으로 state를 끌어올려 공유



state 끌어올리기(Lifting State Up)

❖ 방법

- 상위 컴포넌트의 '상태를 변경하는 함수' 그 자체를 하위 컴포넌트로 전달(props)
- 전달된 함수를 하위 컴포넌트가 실행

[실습] state 끌어올리기

❖ App.js

```
import React, { useState } from 'react';
import ChildComponent from './ChildComponent';

function App() {
  const [value, setValue] = useState('');

  function addDataHandler(data){
    setValue(data);
  }

  return (
    <div>
      <h3>ChildComponent로부터 전달받은 데이터 : {value} </h3>
      <ChildComponent onAddData={addDataHandler}/>
    </div>
  );
};

export default App;
```

[실습] state 끌어올리기

❖ ChildComponent.js

```
import React from 'react';

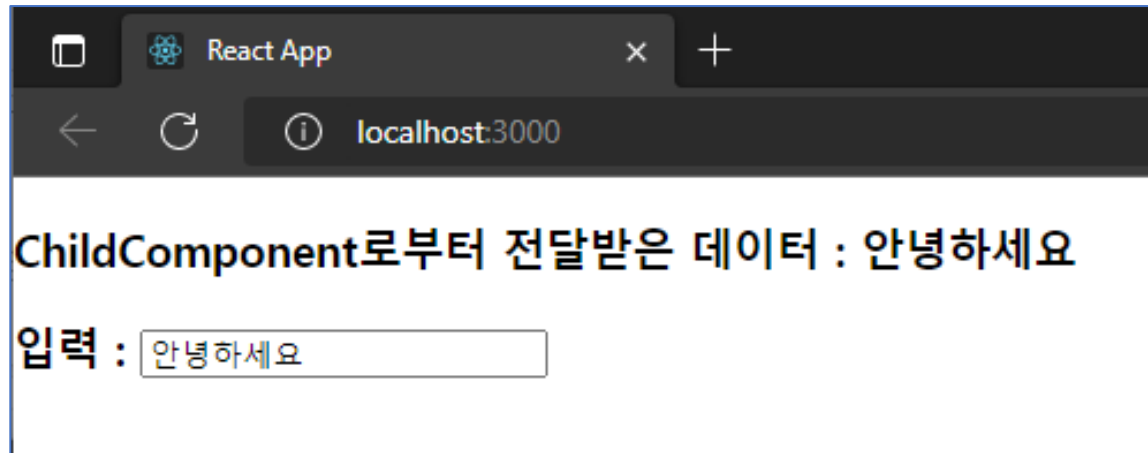
function ChildComponent(props) {
  function updateValue(e){
    props.onAddData(e.target.value);
  }

  return (
    <div>
      <h3>입력 : <input type="text" onChange={updateValue}/></h3>
    </div>
  );
};

export default ChildComponent;
```

[실습] state 끌어올리기

❖ 실행 결과



THANK 😊 YOU