

3. 컴포넌트

작성일시: 2022년 5월 6일 오전 11:08 참고 도서: 리엑트를 다루는 기술

3.0 컴포넌트란?

컴포넌트 기능은 템플릿 이상입니다. 데이터가 주어졌을 때, 이에 맞춰 UI를 만들어주고, 라이프사이클API(함수형 컴포넌트에서는 Hook을 이용)를 이용하여 컴포넌트가 화면에서 나타날 때, 사라질 때, 변화가 일어날 때 주어진 작업들을 처리할 수 있으며, 임의의 메서드를 만들어 특별한 기능을 붙여줄 수 있습니다.

3.1 함수형 컴포넌트의 장점

1. 클래스형 컴포넌트보다 선언하기가 편하다.
2. 메모리 자원도 클래스형 컴포넌트보다 덜 사용한다.
3. 프로젝트를 배포할 때도 함수 컴포넌트를 사용하는 것이 결과물의 파일 크기가 더 작다.

리엑트 공식 메뉴얼에서는 함수 컴포넌트와 hooks을 사용하도록 권장한다.

3.2 첫 컴포넌트 생성

Note) ES6의 화살표 함수

- 화살표 함수(arrow function)는 ES6 문법에서 함수를 표현하는 새로운 방식입니다. 그렇다고 기존의 function을 이용하는 함수 선언 방식을 아예 대체하지 않고, 사용 용도가 조금 다르다. 이 문법은 함수를 파라미터로 전달할 때 유용합니다.

```
setTimeout(function(){
  console.log('hello world');
},1000);

setTimeout(()=>{
  console.log('hello world');
},1000);
```

- 이 문법이 기존 function을 대체할 수 없는 것은 용도가 다르기 때문입니다. 무엇보다 서로 가리키고 있는 this 값이 다르다.

```
function BlackDog() {
  this.name = '흰둥이';
  return{
    name:'검둥이',
    bark: function() {
      console.log(this.name + ': 멍멍!');
    }
  }
}
```

```
const blackDog = new BlackDog();
blackDog.bark(); //검둥이: 멍멍!

function WhiteDog() {
  this.name = '흰둥이';
  return{
    name: '검둥이',
    bark: () =>{
      console.log(this.name + ': 멍멍!')
    }
  }
}

const WhiteDog = new WhiteDog();
WhiteDog.bark(); //흰둥이: 멍멍!
```

함수 컴포넌트를 선언할 때 function 키워드를 사용하는 것과 화살표 함수 문법을 사용하는 것에 큰 차이가 없습니다.

3.2.2 컴포넌트 코드작성 및 모듈 내보내기

3.2.3.1 모듈 내보내기(export)

```
export default MyComponent;
```

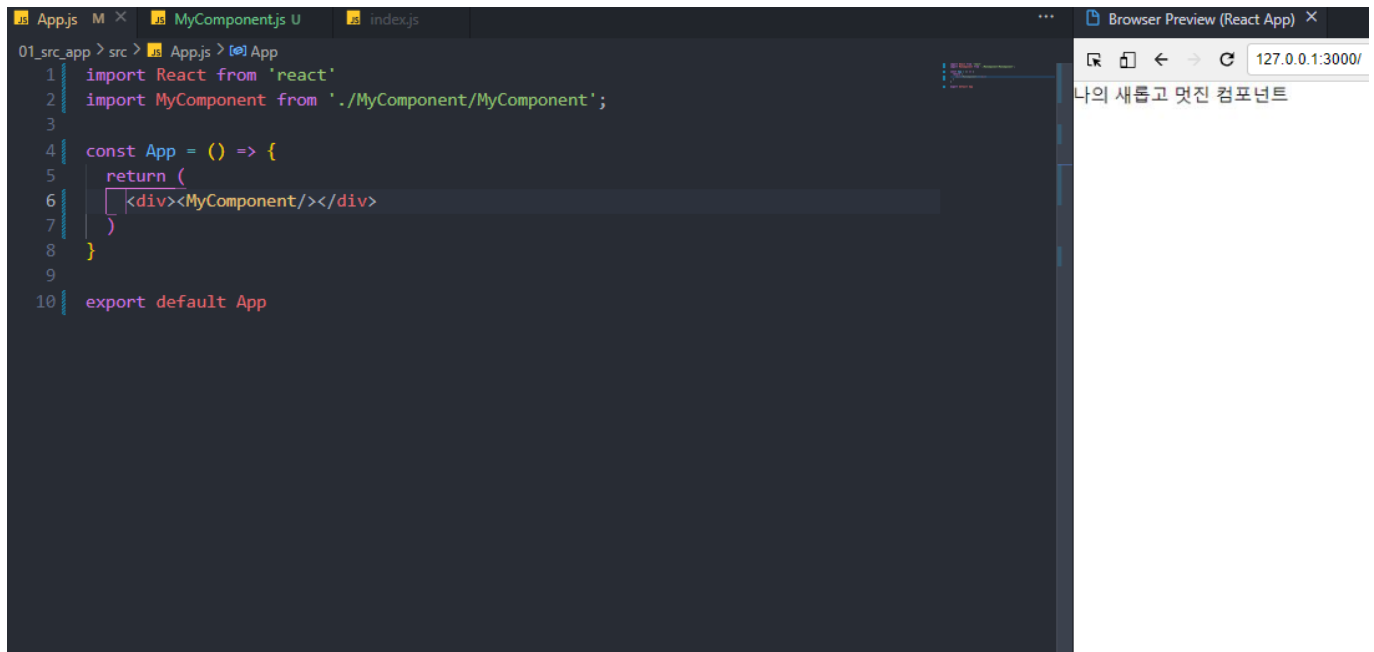
3.2.3 모듈 불러오기

App.js

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

const App = () => {
  return (
    <div><MyComponent/></div>
  );
};

export default App;
```



MyComponent.js

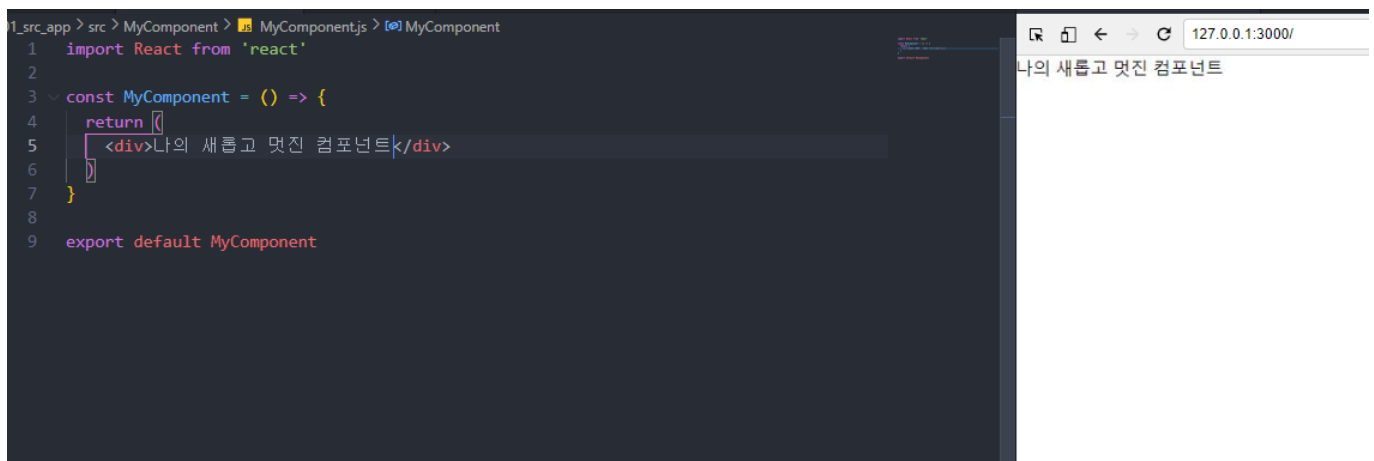
```

import React from 'react'

const MyComponent = () => { //화살표 함수를 사용
  return (
    <div>나의 새롭고 멋진 컴포넌트</div>
  );
};

export default MyComponent; // 모듈 내보내기

```



3.3 props(properties 약자)

컴포넌트 속성을 설정할 때 사용하는 요소로, props 값은 부모 컴포넌트에서 설정할 수 있습니다.

3.3.1 JSX 내부에서 props 렌더링

MyComponent.js

```
import React from 'react'

const MyComponent = props => { //부모컴포넌트<MyComponent name='ooo' />을 props로 불러옴.
  return <div>안녕하세요. 제이름은{props.name}</div>; //불러온 값을 {props.name}에 대입
};

export default MyComponent;
```

The screenshot shows a code editor with the following content in `MyComponent.js`:

```
1 import React from 'react'
2
3 const MyComponent = props => {
4   return <div>안녕하세요. 제이름은{props.name}</div>;
5 }
6
7 export default MyComponent
```

To the right, a browser window at `127.0.0.1:3000/` displays the rendered output: `안녕하세요. 제이름은React`.

3.3.2 컴포넌트를 사용할 때 props 값 지정하기

App.js

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

const App = () => {
  return <MyComponent name="React"/>;
  //MyComponent 부모 컴포넌트인 App.js에서 props값 지정
};

export default App
```

The screenshot shows a code editor with the following content in `App.js`:

```
1 import React from 'react'
2 import MyComponent from './MyComponent/MyComponent';
3
4 const App = () => {
5   return <MyComponent name="React"/>;
6 }
7
8
9 export default App
```

To the right, a browser window at `127.0.0.1:3000/` displays the rendered output: `안녕하세요. 제이름은React`.

3.3.3 props 기본값 설정: defaultProps

App.js

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

const App = () => {
  return <MyComponent/>;
};

export default App;
```

```
01_src_app > src > App.js > ...
1 import React from 'react'
2 import MyComponent from './MyComponent/MyComponent';
3
4 const App = () => {
5   return <MyComponent/>;
6 }
7 };
8
9 export default App;
```

127.0.0.1:3000/

안녕하세요, 제이름은기본이름

MyComponent.js

```
import React from 'react'

const MyComponent = props => {
  return <div>안녕하세요. 제이름은{props.name}</div>;
};

MyComponent.defaultProps = {
  name: '기본이름'
};

export default MyComponent;
```

```
01_src_app > src > MyComponent > MyComponent.js > ...
1 import React from 'react'
2
3 const MyComponent = props => {
4   return <div>안녕하세요. 제이름은{props.name}</div>;
5 };
6
7 MyComponent.defaultProps = {
8   name: '기본이름'
9 };
10
11 export default MyComponent;
```

127.0.0.1:3000/

안녕하세요, 제이름은기본이름

3.3.4 태그 사이의 내용을 보여주는 children

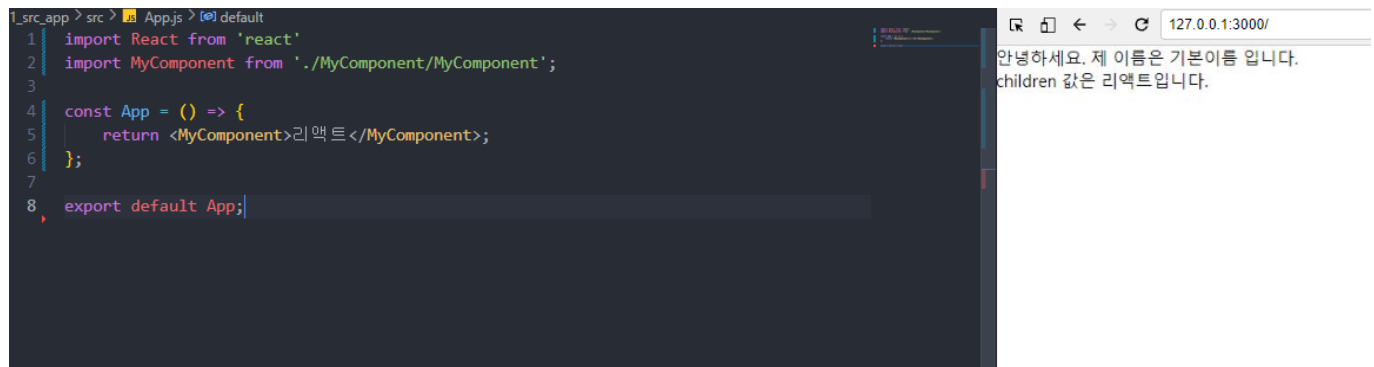
리액트 컴포넌트 사용 시, 컴포넌트 태그 사이의 내용을 보여주는 props가 children이다.

App.js

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

const App = () => {
  return <MyComponent>리액트</MyComponent>; //<MyComponent>리액트</MyComponent>값
  을 전달
};

export default App;
```



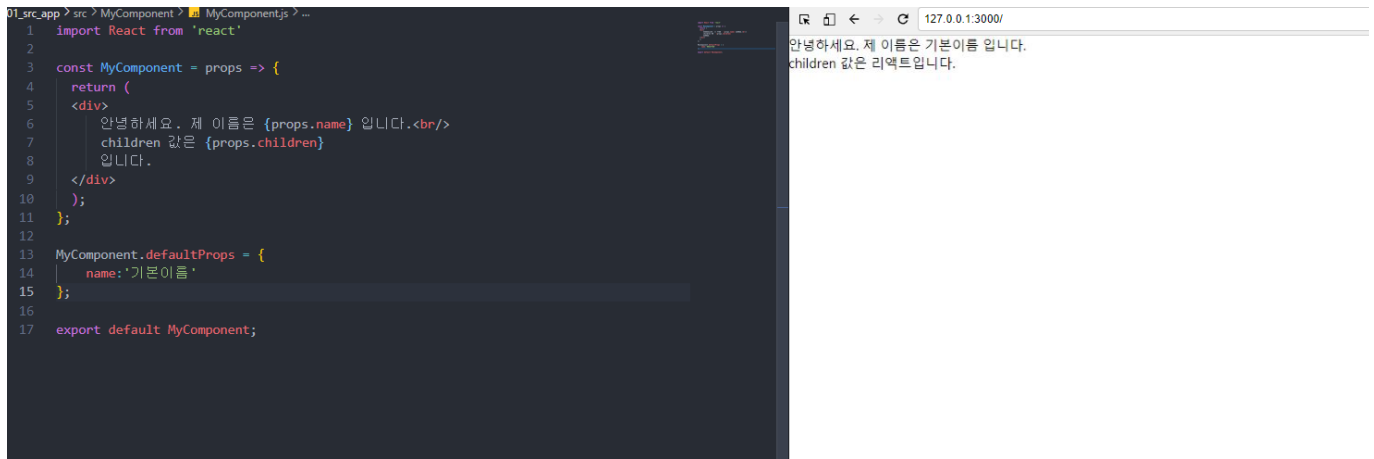
MyComponent.js

```
import React from 'react'

const MyComponent = props => {
  return (
    <div>
      안녕하세요. 제 이름은 {props.name} 입니다.<br/>
      children 값은 {props.children} {/*전달된 props 값이 {props.children}에서 출력
      됨*/}
      입니다.
    </div>
  );
};

MyComponent.defaultProps = {
  name: '기본이름'
};

export default MyComponent;
```



3.3.5 비구조화 할당 문법을 통해 props 내부 값 추출하기

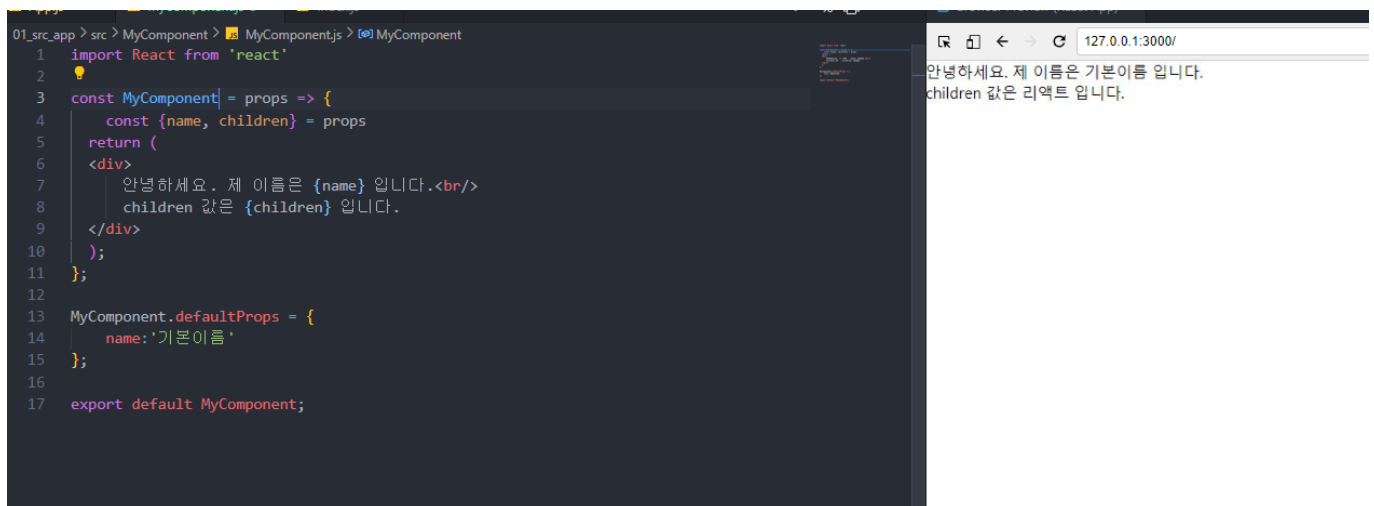
MyComponent.js

```
import React from 'react'

const MyComponent = props => {
  const {name, children} = props; //비구조 문법
  return (
    <div>
      안녕하세요. 제 이름은 {name} 입니다.<br/>
      children 값은 {children} 입니다. { /*변수로 뽑아서 사용 가능*/ }
    </div>
  );
};

MyComponent.defaultProps = {
  name: '기본이름'
};

export default MyComponent;
```



props 대신 객체를 사용한 MyComponent.js

```
import React from 'react'

const MyComponent = ({name, children}) => {
  return (
    <div>
      안녕하세요. 제 이름은 {name} 입니다.<br/>
      children 값은 {children} 입니다.
    </div>
  );
};

MyComponent.defaultProps = {
  name: '기본이름'
};

export default MyComponent;
```



The screenshot shows a code editor with the following code in `MyComponent.js`:

```
1 import React from 'react'
2
3 const MyComponent = ({name, children}) => {
4   return (
5     <div>
6       안녕하세요. 제 이름은 {name} 입니다.<br/>
7       children 값은 {children} 입니다.
8     </div>
9   );
10 };
11
12 MyComponent.defaultProps = {
13   name: '기본이름'
14 };
15
16 export default MyComponent;
```

To the right, a browser window at `127.0.0.1:3000/` displays the rendered output:

```
안녕하세요. 제 이름은 기본이름 입니다.
children 값은 리액트 입니다.
```

3.3.6 propTypes를 통한 props검증

MyComponent.js

```
import React from 'react'
import PropTypes from 'prop-types';
//컴포넌트의 필수 props를 지정하거나 props의 타입(type)을 지정할 때는 propTypes를 사
용.

const MyComponent = ({name, children}) => {
  return (
    <div>
      안녕하세요. 제 이름은 {name} 입니다.<br/>
      children 값은 {children} 입니다.
    </div>
  );
};

MyComponent.defaultProps = {
  name: '기본이름'
};
```



```
};

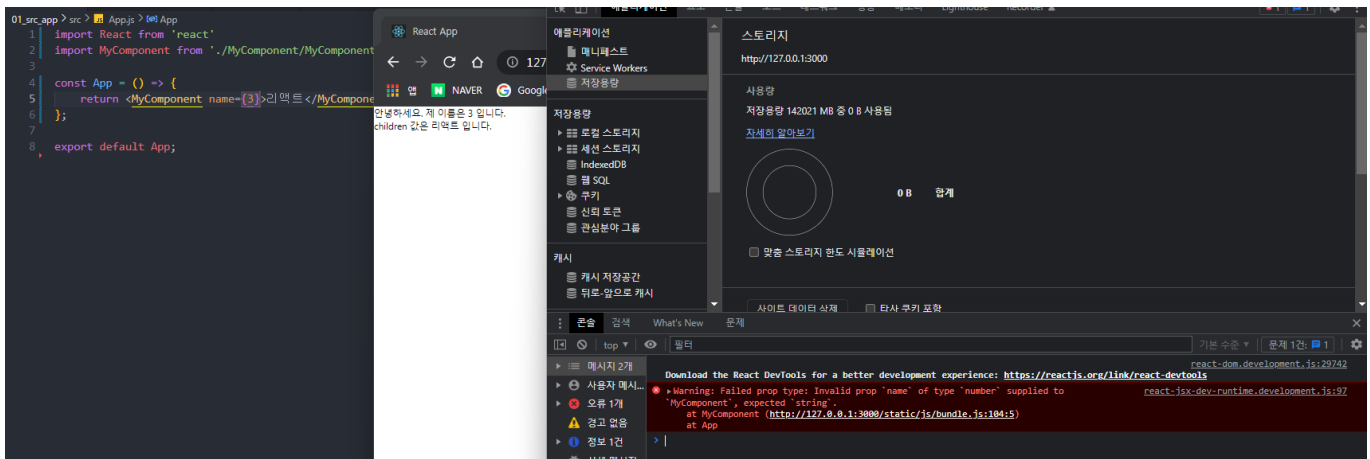
MyComponent.propTypes = {
  name: PropTypes.string
};
// 이렇게 설정하면 name값은 무조건 문자열(string) 형태로 전달해야한다.
export default MyComponent;
```

App.js

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

const App = () => {
  return <MyComponent name={3}>리액트</MyComponent>;
};

export default App;
```

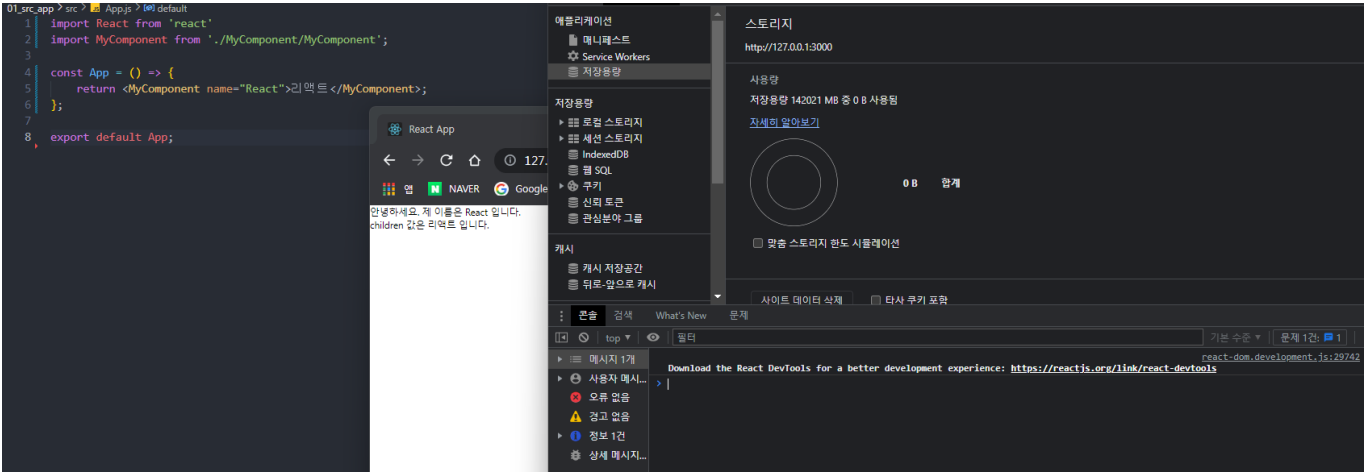


App.js

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

const App = () => {
  return <MyComponent name="React">리액트</MyComponent>;
};

export default App;
```



3.3.6.1 isRequired를 사용하여 필수 propTypes 설정

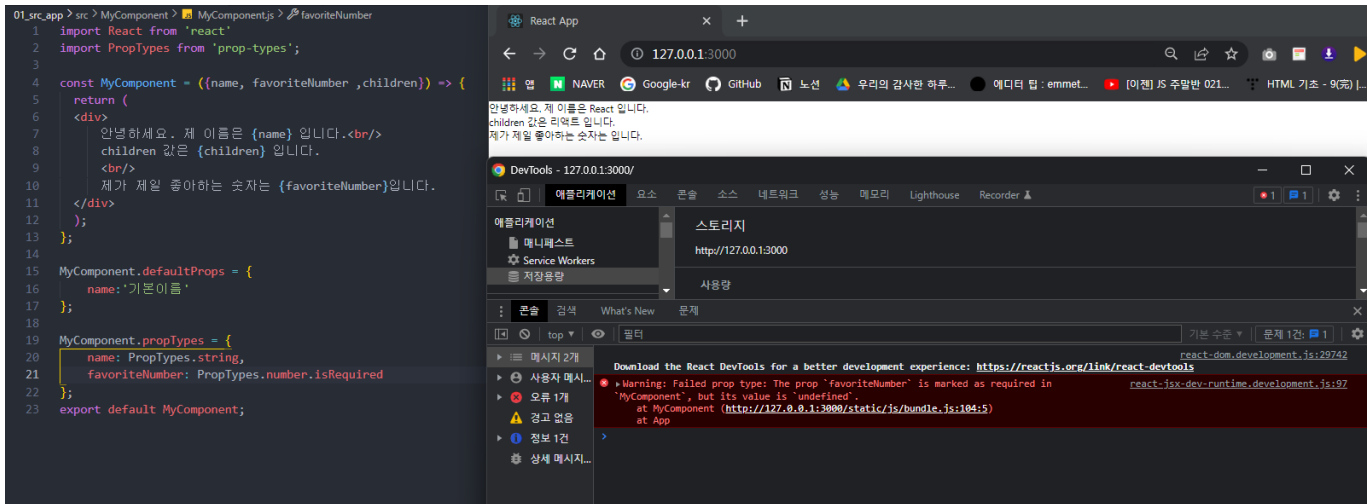
MyComponent.js

```
import React from 'react'
import PropTypes from 'prop-types';

const MyComponent = ({name, favoriteNumber ,children}) => {
  return (
    <div>
      안녕하세요. 제 이름은 {name} 입니다.<br/>
      children 값은 {children} 입니다.
      <br/>
      제가 제일 좋아하는 숫자는 {favoriteNumber}입니다.
    </div>
  );
};

MyComponent.defaultProps = {
  name: '기본이름'
};

MyComponent.propTypes = {
  name: PropTypes.string,
  favoriteNumber: PropTypes.number.isRequired //숫자형을 필수 값으로 지정
};
export default MyComponent;
```

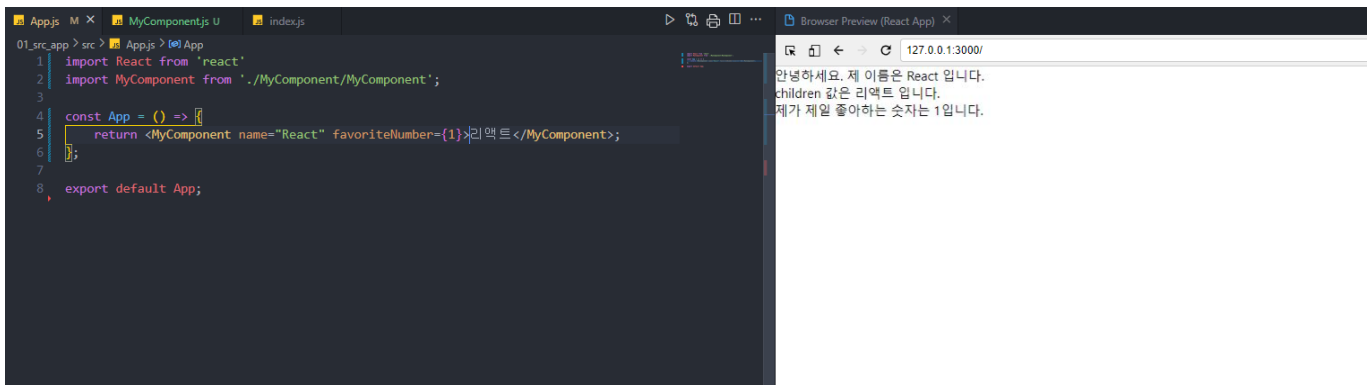


App.js

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

const App = () => {
  return <MyComponent name="React" favoriteNumber={1}>리액트</MyComponent>;
};

export default App;
```



3.3.6.2 더 많은 PropTypes 종류

array	배열
bool	true 혹은 false 값
func	함수
number	숫자
object	객체
string	문자열
instanceOf(클래스)	특정 클래스의 인스턴스(예:instanceOf(MyClass))

array	배열
oneOf(['값1','값2'])	주어진 배열 요소 중 값 하나
oneOfType([React.PropTypes.string, PropTypes.number])	주어진 배열 안의 종류 중 하나(값 제한으로 값1,과 값2 중 하나만 선택 가능)
any	아무 종류(타입지정 안함)

3.4 state

리액트에서 state는 컴포넌트 내부에서 바뀔 수 있는 값을 의미함. props는 컴포넌트가 사용되는 과정에서 부모 컴포넌트가 설정하는 값이다. 컴포넌트 자신은 해당 props를 읽기 전용으로만 사용한다. props를 변경하려면 부모 컴포넌트에서 바꿔줘야한다.

3.4.2 함수 컴포넌트에서 useState 사용하기(HOOK기능)

3.4.2.1 배열 비구조화 할당

배열 비구조화 할당은 객체 비구조화 할당과 유사한다. 즉, 배열 안에 들어 있는 값을 쉽게 추출할 수 있도록 해주는 문법이다.

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

// 아래는 배열 비구조화 할당을 사용한 예제입니다.
const array =[1,2];
const [one,two] = array;
```

####3.4.2.2 useState 사용하기

useState란? useState 함수의 인자()에는 상태의 초기값을 넣어줘야한다. 값의 형태(숫자, 문자열, 객체, 배열)는 자유이다. useState함수를 호출하면 배열이 반환된다. const [원소1, 원소2] 배열의 첫 번째 원소는 현재 상태, 두 번째 원소는 상태를 바꿔주는 세터(setter) 함수이다. 배열 비구조화 할당을 통해 이름을 자유롭게 정해줄 수 있다.

Say.js

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

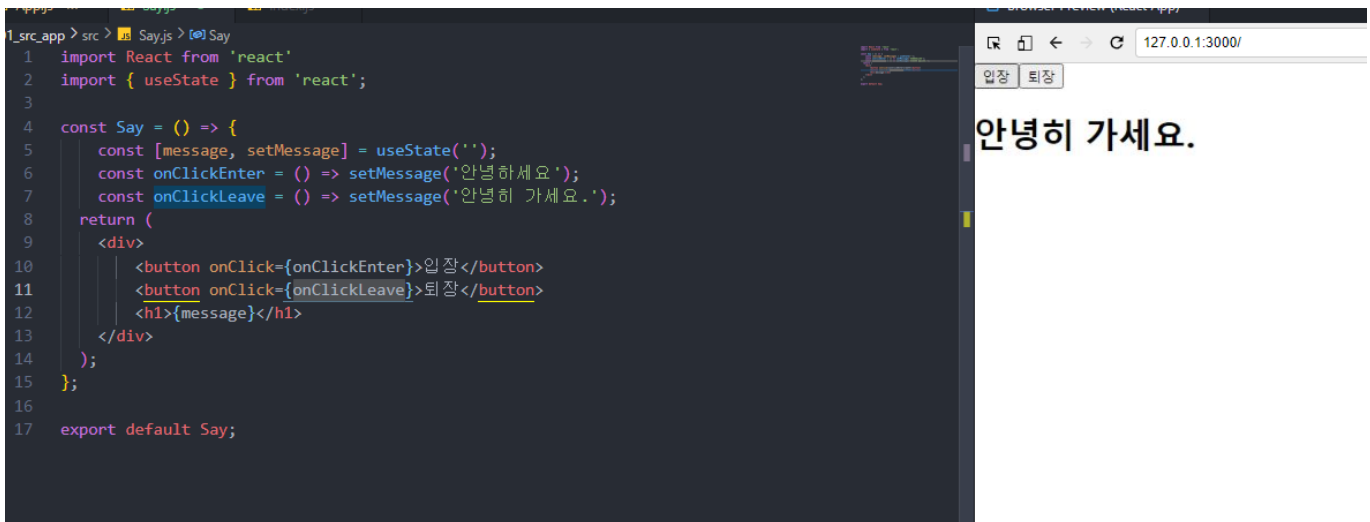
const Say = () => {
  const [message, setMessage] = useState(''); //비구조 문법 => 배열을 리턴한다.
  // message=변수, setMessage=변수에 대한 setter, ( )=변수의 초기값
  const onClickEnter = () => setMessage('안녕하세요');
  const onClickLeave = () => setMessage('안녕히 가세요. ');
  return (
    <div>
```

```

    <button onClick={onClickEnter}>입장</button>
    <button onClick={onClickLeave}>퇴장</button>
    <h1>{message}</h1>
  /*{message}은 setter에 의해 값이 변경되며 실시간으로 화면이 갱신된다.*/
  </div>
);
};

export default Say;

```



App.js

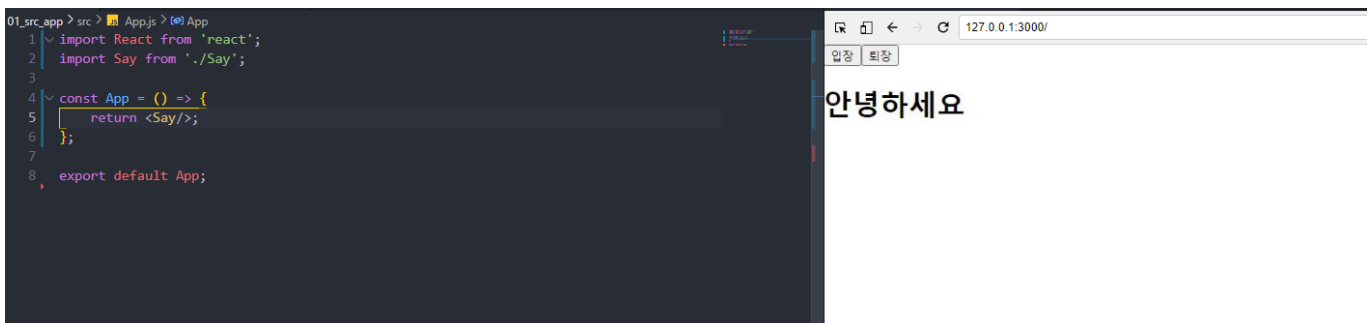
```

import React from 'react';
import Say from './Say';

const App = () => {
  return <Say/>;
};

export default App;

```



3.4.2.3 한 컴포넌트에서 useState 여러 번 사용하기

useState는 한 컴포넌트에서 여러 번 사용해도 상관없다.

Say.js

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

const Say = () => {
  const [message, setMessage] = useState('');
  const onClickEnter = () => setMessage('안녕하세요');
  const onClickLeave = () => setMessage('안녕히 가세요. ');
  const [color, setColor] = useState('black');
  //ES6부터는 변수이름과 key이름이 동일하다면, key이름은 변수이름으로 자동 생성 된다.

  return (
    <div>
      <button onClick={onClickEnter}>입장</button>
      <button onClick={onClickLeave}>퇴장</button>
      <h1 style={{color}}>{message}</h1>

      <button style={{color:'red'}} onClick={()=> setColor('red')}}>
        빨간색
      </button>
      <button style={{color:'green'}} onClick={()=> setColor('green')}}>
        초록색
      </button>
      <button style={{color:'blue'}} onClick={()=> setColor('blue')}}>
        파란색
      </button>
    </div>
  );
};

export default Say;
```

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

const Say = () => {
  const [message, setMessage] = useState('');
  const onClickEnter = () => setMessage('안녕하세요');
  const onClickLeave = () => setMessage('안녕히 가세요. ');
  const [color, setColor] = useState('black');
  //ES6부터는 변수이름과 key이름이 동일하다면, key이름은 변수이름으로 자동 생성 된다.

  return (
    <div>
      <button onClick={onClickEnter}>입장</button>
      <button onClick={onClickLeave}>퇴장</button>
      <h1 style={{color}}>{message}</h1>

      <button style={{color:'red'}} onClick={()=> setColor('red')}}>
        빨간색
      </button>
      <button style={{color:'green'}} onClick={()=> setColor('green')}}>
        초록색
      </button>
      <button style={{color:'blue'}} onClick={()=> setColor('blue')}}>
        파란색
      </button>
    </div>
  );
};

export default Say;
```

입장 퇴장

안녕하세요

빨간색 초록색 파란색



3.5 state를 사용할 때 주의사항

state값을 바꿔야 할 때는 `setState` 혹은 `useState`를 통해 전달(리턴)받은 세터 함수를 사용해야한다.

//객체다루기 예제

```
const object = {a:1, b:2, c:3}; //{a:1, b:2, c:3}은 상태값을 가정한 객체
const nextObject = { ...object b:2 } //사본을 만들어서 b값만 덮어 쓰기

// 배열 다루기
const array=[
  {id:1, value: true},
  {id:2, value: true},
  {id:3, value: false},
];
let nextArray = array.concat({id:4}); //새 항목 추가
nextArray.filter(item => item.id !==2); //id가 2인 항목 제거
nextArray.map(item => (item.id ===1 ? {...item, value:false} : item)); //id가 1인
항목의 value를 false로 설정
```

3.6 컴포넌트 정리

1. props는 부모 컴포넌트가 설정하고, state는 컴포넌트 자체적으로 지닌 값으로 내부에서 값을 업데이트 할 수 있다.

2. 부모 컴포넌트 state를 자식 컴포넌트의 props로 전달, 자식 컴포넌트에서 특정 이벤트가 발생할 때 부모 컴포넌트의 메서드를 호출하면 props도 유동적으로 사용할 수 있다.

4. 이벤트 핸들링

작성일시: 2022년 5월 6일 오후 2:32 참고 도서: 리엑트를 다루는 기술

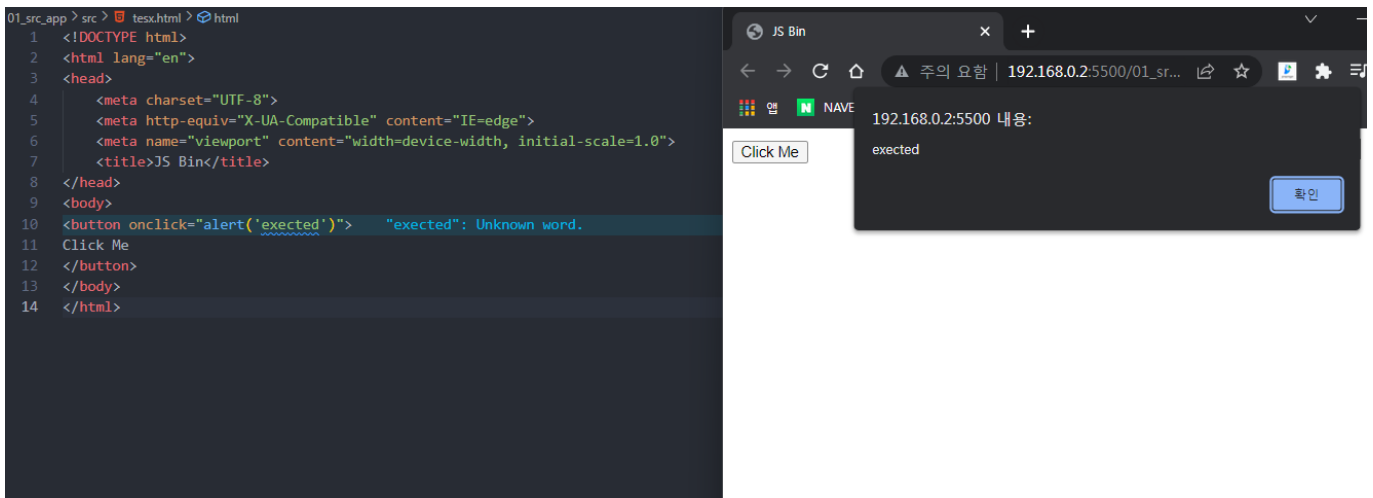
4.0 HTML에서 DOM 요소에 이벤트 설정

이벤트란?

사용자가 웹 브라우저에서 DOM 요소들과 상호 작용을 하는 것을 이벤트라고 한다.

test.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <metaname="viewport" content="width=device-width, initial-scale=1.0">
  <title>JS Bin</title>
</head>
<body>
  <button onclick="alert('exected')"> //addEventLister로 전환될 수 있음
    Click Me
  </button>
</body>
</html>
```



4.1 리엑트의 이벤트 시스템

리엑트의 이벤트 시스템은 웹 브라우저의 HTML 이벤트와 호출 방법이 동일하기 때문에 사용법이 비슷하다.

Say.js

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

const Say = () => {
  const [message, setMessage] = useState('');
  const onClickEnter = () => setMessage('안녕하세요');
  const onClickLeave = () => setMessage('안녕히 가세요. ');
  const [color, setColor] = useState('black');
  //ES6부터는 변수이름과 key이름이 동일하다면, key이름은 변수이름으로 자동 생성 된다.

  return (
    <div>
      <button onClick={onClickEnter}>입장</button>
      { /*버튼에 클릭이벤트를 걸어서 onClickEnter와 onClickLeave 지정한 변수값 호출 */ }
      <button onClick={onClickLeave}>퇴장</button>
    </div>
  )
}
```

4.1.1 이벤트 사용시 주의사항

1. 이벤트 이름은 카멜 표기법으로 작성되어야함 ⇒ ex)onClick
2. 이벤트에 실행할 함수 형태의 값을 전달해야함. ⇒ HTML과 값 전달 방법이 다르다.
3. DOM 요소에만 이벤트 설정을 할 수 있다.
 - div, button, input, form과 같은 DOM 요소에는 이벤트 설정을 할 수 있지만, 직접 만든 컴포넌트에는 이벤트 설정이 불가능하다.

4.2 예제로 이벤트 핸들링 익히기

4.2.1 컴포넌트 생성 및 불러오기

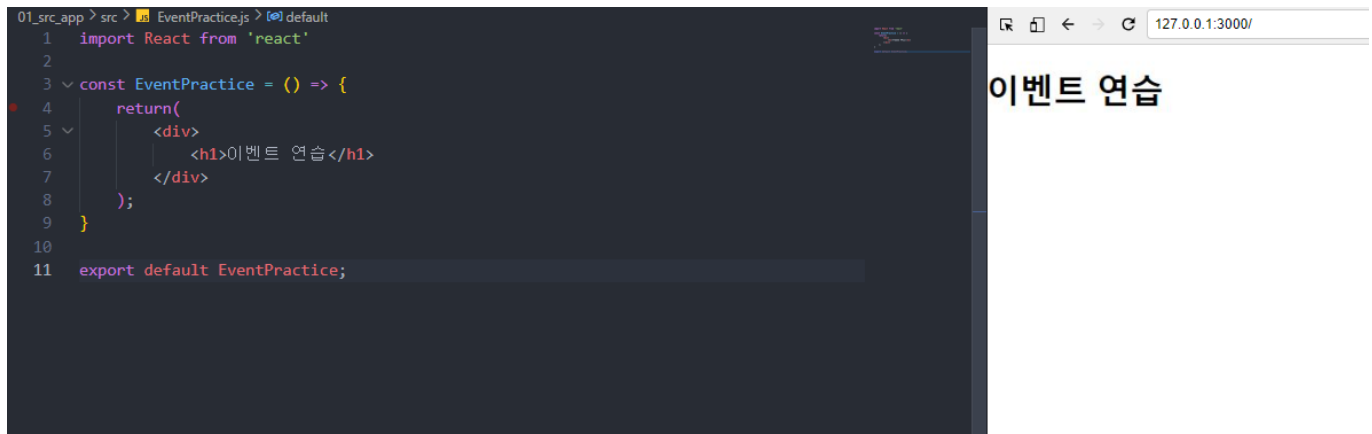
4.2.1.1 컴포넌트 생성

EventPractice.js

```
import React from 'react'

const EventPractice = () => {
  return(
    <div>
      <h1>이벤트 연습</h1>
    </div>
  );
}

export default EventPractice;
```



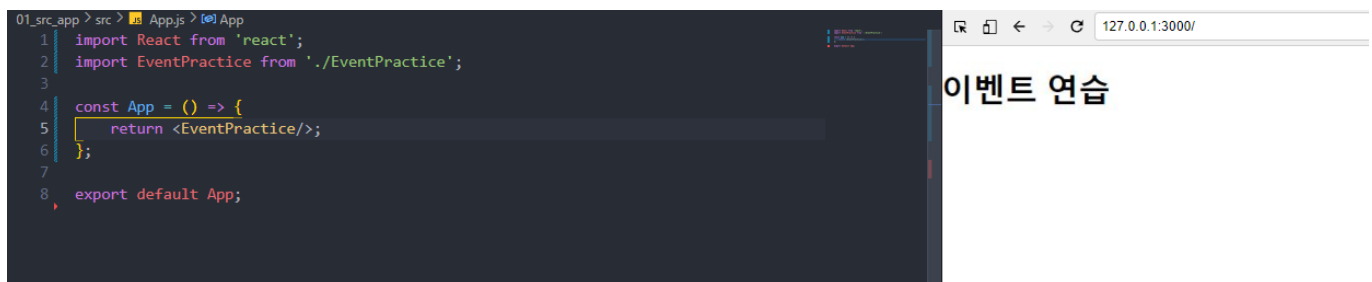
```
01_src_app > src > EventPractice.js > default
1  import React from 'react'
2
3  const EventPractice = () => {
4    return(
5      <div>
6        <h1>이벤트 연습</h1>
7      </div>
8    );
9  }
10
11  export default EventPractice;
```

이벤트 연습

App.js

```
import React from 'react';
import EventPractice from './EventPractice'; // import 설정은 함수형과 클래스형 모두 동일
const App = () => {
  return <EventPractice/>;
};

export default App;
```



```
01_src_app > src > App.js > App
1  import React from 'react';
2  import EventPractice from './EventPractice';
3
4  const App = () => {
5    return <EventPractice/>;
6  };
7
8  export default App;
```

이벤트 연습

4.2.2 onChange 이벤트 핸들링하기

4.2.2.1 onChange 이벤트 설정

```
import React from 'react'

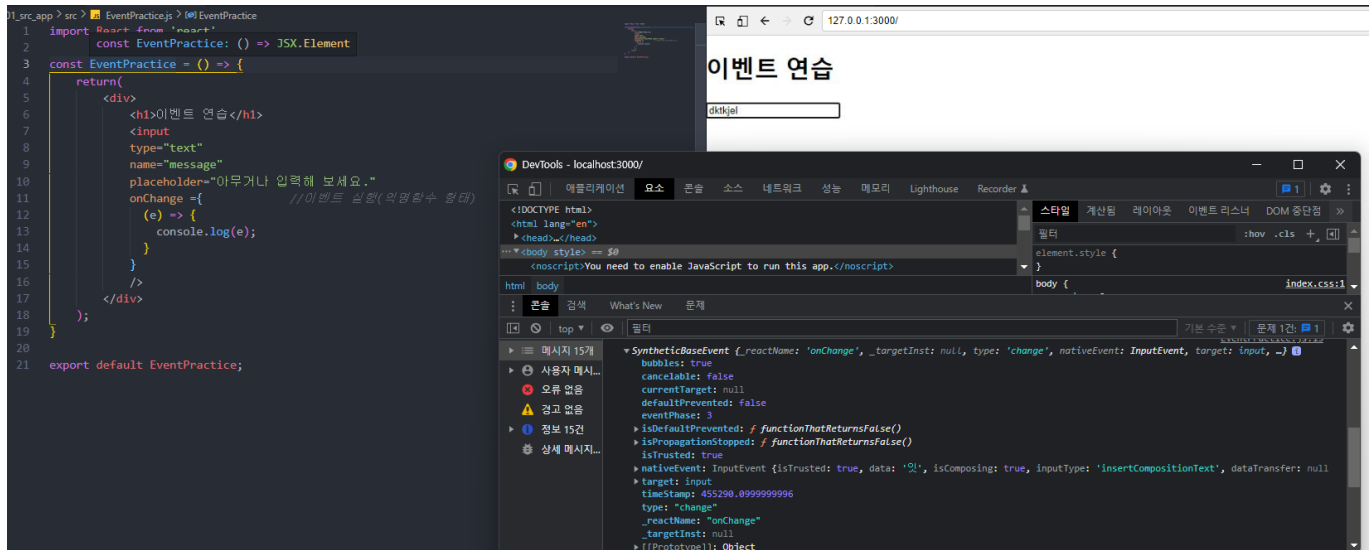
const EventPractice = () => {
  return(
    <div>
      <h1>이벤트 연습</h1>
      <input
        type="text"
        name="message"
        placeholder="아무거나 입력해 보세요."
        onChange ={ //이벤트 실행(익명함수 형태)
          (e) => {
            console.log(e);
          }
        }
      />
    </div>
  );
};
```

```

    }
  />
</div>
);
}

export default EventPractice;

```



EventPractice.js

```

import React from 'react'

const EventPractice = () => {
  return(
    <div>
      <h1>이벤트 연습</h1>
      <input
        type="text"
        name="message"
        placeholder="아무거나 입력해 보세요."
        onChange ={ //onChange가 발생했을 때, setMessage값이 업데이트가
                      됩.
                      (e) => {
                        console.log(e.target.value);
                      }
                      />
      </div>
    );
  }

  export default EventPractice;

```



4.2.2.2 state에 input 값 담기

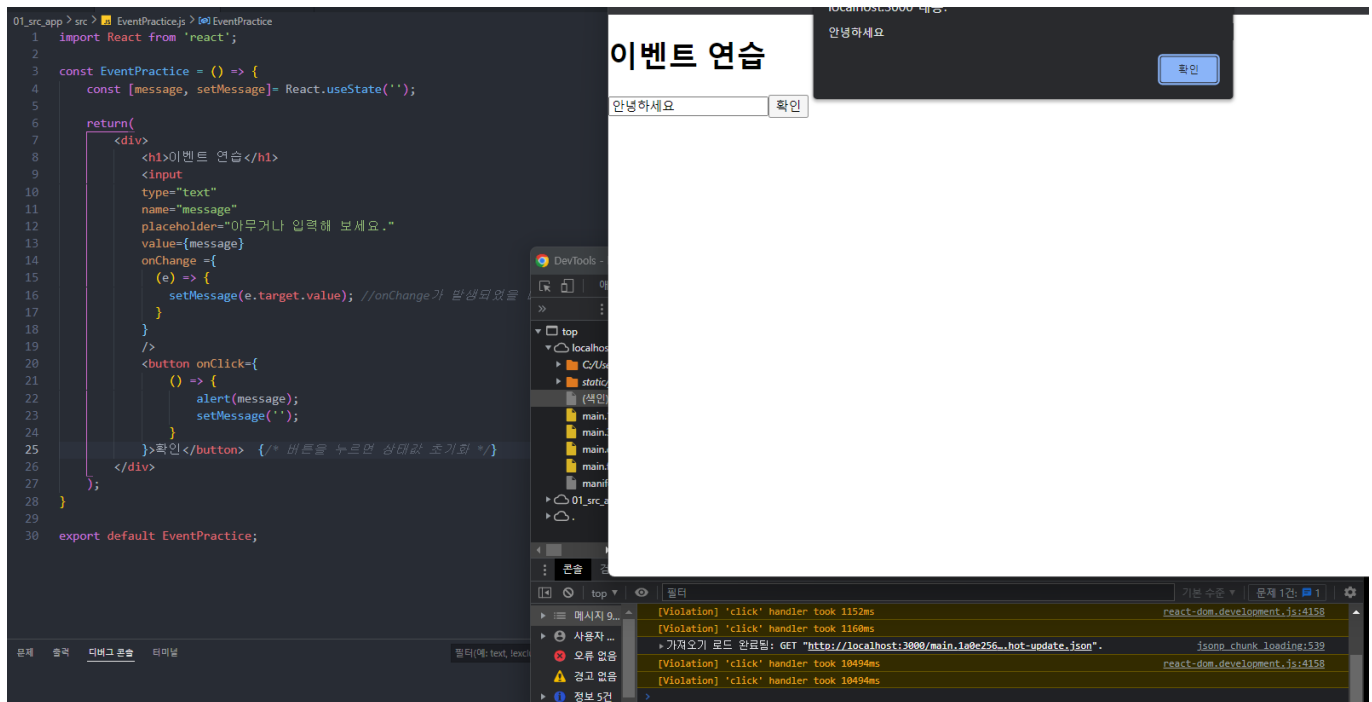
```

import React from 'react';

const EventPractice = () => {
  const [message, setMessage]= React.useState('');
  // 변경된 setMessage 값을 message로 전달하고=> 그 값을 input태그 value에 전달
  return(
    <div>
      <h1>이벤트 연습</h1>
      <input
        type="text"
        name="message"
        placeholder="아무거나 입력해 보세요."
        value={message}
        onChange ={
          (e) => {
            setMessage(e.target.value);
          }
        }
      </input>
      <button onClick={
        () => {
          alert(message);
          setMessage(''); /*변경된 setter 값을 변수에 대한 setter에 대입
        */
      }
      >확인</button>  /* 버튼을 누르면 상태값 초기화 */
    </div>
  );
}

```

```
export default EventPractice;
```



4.2.3 임의 메서드 만들기

4.2.3.1 기본방식

```
import React from 'react';

const EventPractice = () => {
  const [message, setMessage]= React.useState('');
  const handleChange = (e) =>{
    setMessage(e.target.value);
  }
  const handleClick=()=>{
    alert(message);
    setMessage('');
  }

  return(
    <div>
      <h1>이벤트 연습</h1>
      <input
        type="text"
        name="message"
        placeholder="아무거나 입력해 보세요."
        value={message}
        onChange ={handleChange}
      />
      <button onClick={handleClick}>확인</button>
    </div>
  );
};
```

}

export default EventPractice;

```

01_src_app > src > EventPractice.js > EventPractice
1  import React from 'react';
2
3  const EventPractice = () => {
4    const [message, setMessage] = React.useState('');
5    const handleChange = (e) =>{
6      setMessage(e.target.value);
7    }
8    const handleClick={() =>{
9      alert(message);
10     setMessage('');
11   }}
12
13   return(
14     <div>
15       <h1>이벤트 연습</h1>
16       <input
17         type="text"
18         name="message"
19         placeholder="아무거나 입력해 보세요."
20         value={message}
21         onChange={handleChange}
22       />
23       <button onClick={handleClick}>확인</button>
24     </div>
25   );
26 }
27
28 export default EventPractice;

```

이벤트 연습

이벤트 연습

확인

확인

4.3 함수 컴포넌트로 구현해 보기

EventPractice.js

```

import {useState} from 'react';

const EventPractice = () => {
  const [username, setUsername] = useState('');
  const [message, setMessage] = useState('');
  const onChangeUsername = e => setUsername(e.target.value);
  const onChangeMessage = e => setMessage(e.target.value);
  const onClick = () =>{
    alert(username + ':' + message);
    setUsername('');
    setMessage('');
  };
  const onKeyPress = e =>{
    if(e.key === 'Enter'){
      onClick();
    }
  };
  return (
    <div>
      <h1>이벤트 연습</h1>
      <input
        type="text"
        name="username"
        placeholder='사용자명'
        value={username}
        onChange={onChangeUsername}
      />

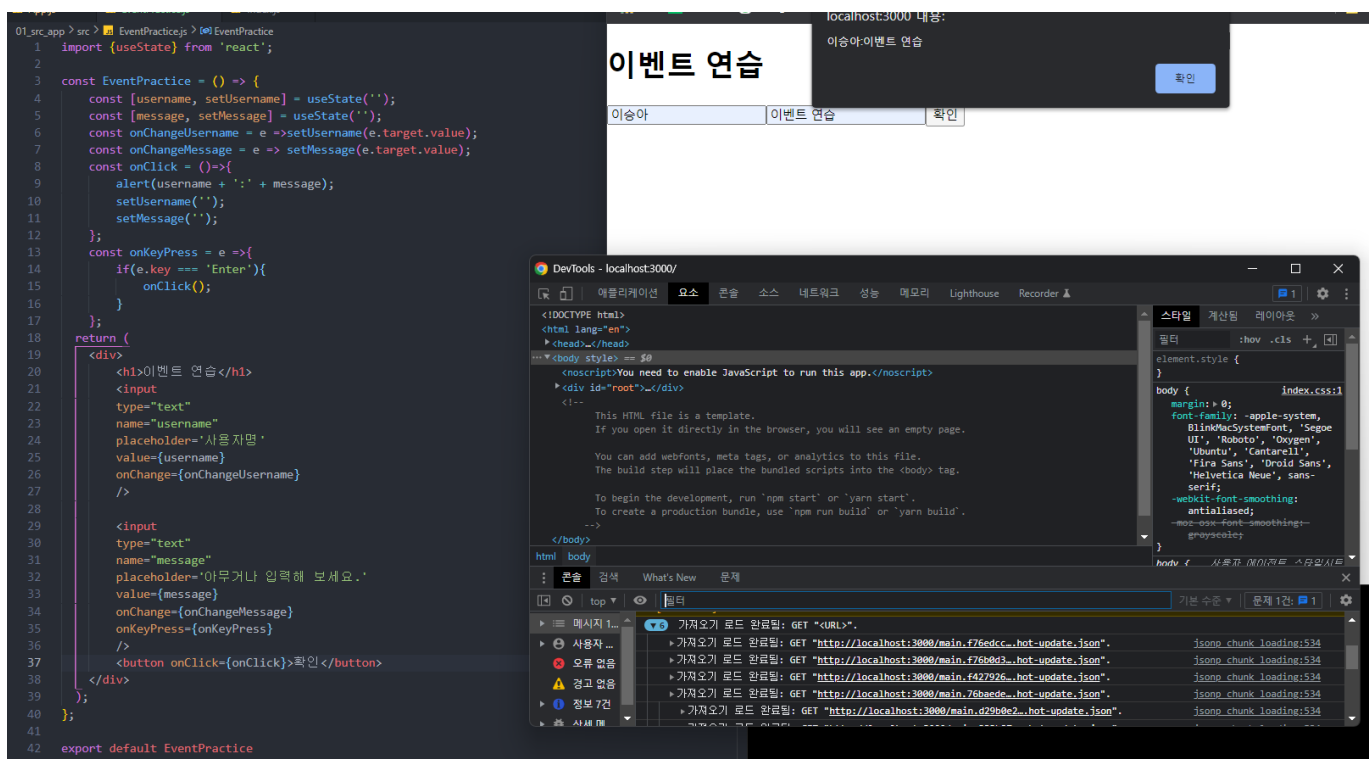
```

```

    <input
      type="text"
      name="message"
      placeholder='아무거나 입력해 보세요.'
      value={message}
      onChange={onChangeMessage}
      onKeyDown={onKeyPress}
    />
    <button onClick={onClick}>확인</button>
  </div>
);
};

export default EventPractice;

```



EventPractice.js

```

import {useState} from 'react';

/** const EventPractice = () => {
  const [form, setForm] = useState({
    username: '',
    message: ''
  });
  * 상태값이 JSON 형태로 구성되어있기 때문에,
  * 하나의 form에 값에 여러 하위 값을 포함 시킬 수 있다.
  */

  const EventPractice = () => {
    const [form, setForm] = useState({

```



```

        username: '',
        message: ''
    });
    const { username, message } = form;
    /**
     * name으로 값 지정된 두개의 값을 한번에 처리 할 수 있다.
     */
    const onChange= e =>{
        const nextForm ={
            ...form, //기존의 form 내용을 이 자리에 복사한 뒤
            [e.target.name]: e.target.value //원하는 값을 덮어 씌우기
        };
        setForm(nextForm);
    };
    * name으로 값 지정된 두개의 값을 한번에 처리 할 수 있다.
    */
    const onChange= e =>{
        const nextForm ={
            ...form, //기존의 form 내용을 이 자리에 복사한 뒤
            [e.target.name]: e.target.value //원하는 값을 덮어 씌우기
        };
        setForm(nextForm);
    };
    const onClick = ()=>{
        alert(username + ':' + message);
        setForm({
            username: '',
            message: ''
        });
    };
    const onKeyPress = e =>{
        if(e.key === 'Enter'){
            onClick();
        }
    };
    return (
        <div>
            <h1>이벤트 연습</h1>
            <input
                type="text"
                name="username"
                placeholder='사용자명'
                value={username}
                onChange={onChange}
            />

            <input
                type="text"
                name="message"
                placeholder='아무거나 입력해 보세요.'
                value={message}
                onChange={onChange}
                onKeyPress={onKeyPress}
            />
            <button onClick={onClick}>확인</button>
        </div>
    );

```

```

    });
  };

export default EventPractice;

```

