2.4.5 undefined를 렌더링 하는 경우 (최신버젼 변경사항)

리액트 컴포넌트에서는 함수에서 undefined만 반환하여 렌더링 하는 경우 (오류가 나던 상황) 화면에 아무것도 표시되지 않음

3장 컴포넌트

- 데이터가 만들어졌을 때 UI를 생성
- 라이프사이클(함수형 : hook)을 이용하여 컴포넌트가 화면에서 나타날 때, 사라질 때, 변화가 일어날 때 주어진 작업을 처리
- 임의 메서드(함수)를 만들어 특별한 기능 부여 가능

3.1 클래스형 컴포넌트

- 클래스 컴포넌트 보다 함수형 컴포넌트이 좋은 점
  - 선언이 편리
  - 메모리 자원도 클래스형보다 적게 사용
  - 배포할 때에도 사이즈가 더 작음 -기존 함수형 컴포넌트의 경우 state와 라이프사이클 API 사용이 불가능 하였으나 hook을 사용하여 해결

3.2 첫 컴포넌트 생성

3.2.2 코드 작성하기

함수형 컴포넌트 선언시 화살표 함수 사용이 더 간결, 큰차이는 없음

> ReactJS Code Snippet(vscode shortcut)

- RSC : React JS Component
- RSCP : RSC + proptype

3.2.3.1 모듈 내보내기(export)

> export default MyComponent 다른 파일에서 MyComponent를 import 할때 . 위에서 선언한 MyComponent를 불러오도록 설정 를 불러오도록 설정함

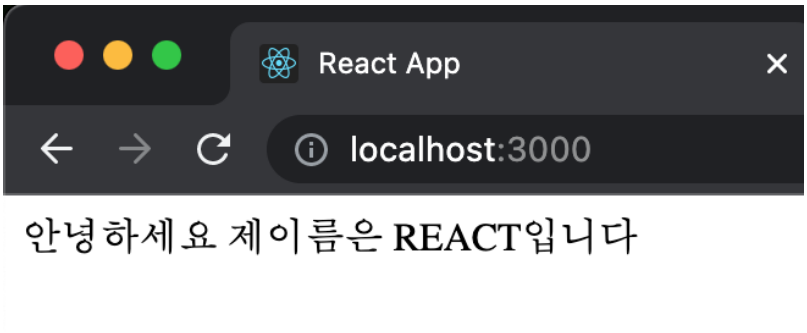
3.2.3.2 모듈 불러오기(import)

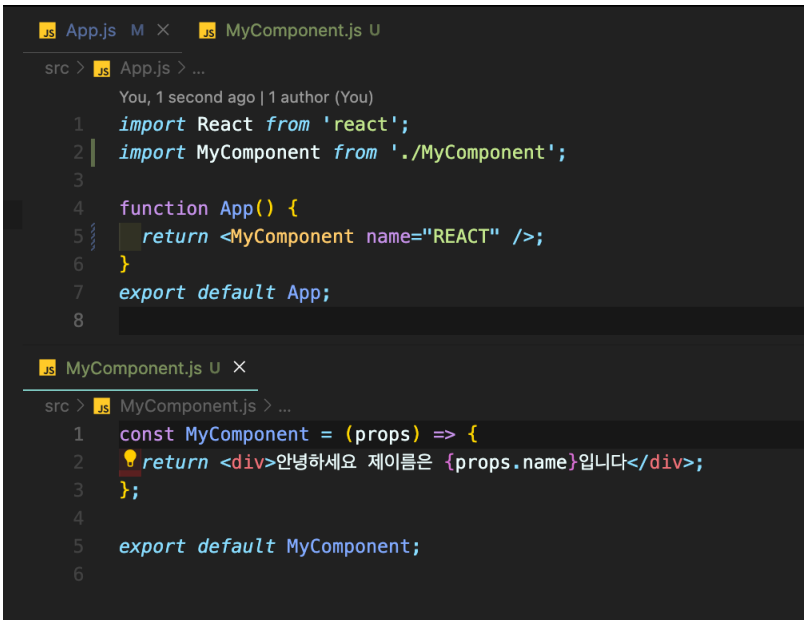
> import (...) from (..파일 위치..) 컴포넌트 최상단에 작성하며, 다른 컴포넌트를 불러온다

3.3 props

- 컴포넌트 속성을 설정
- props 값은 부모 컴포넌트가 자식 컴포넌트를 불러올 때 설정함

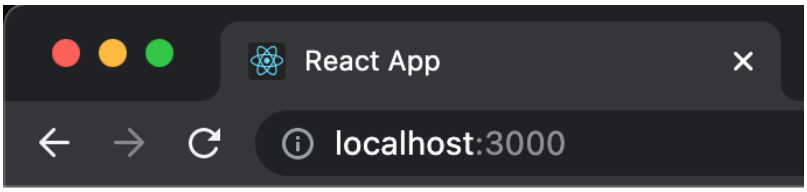
3.3.1 JSX 내부에서 props 렌더링





3.3.3 props 기본값 설정: defaultProps

props값을 따로 지정하지 않았을 때 기본값으로 사용될 값을 설정하는 방법 컴포넌트 작성 후 컴포넌트명.defaultProps = { json형식 } 으로 지정



안녕하세요 제이름은 기본이름입니다

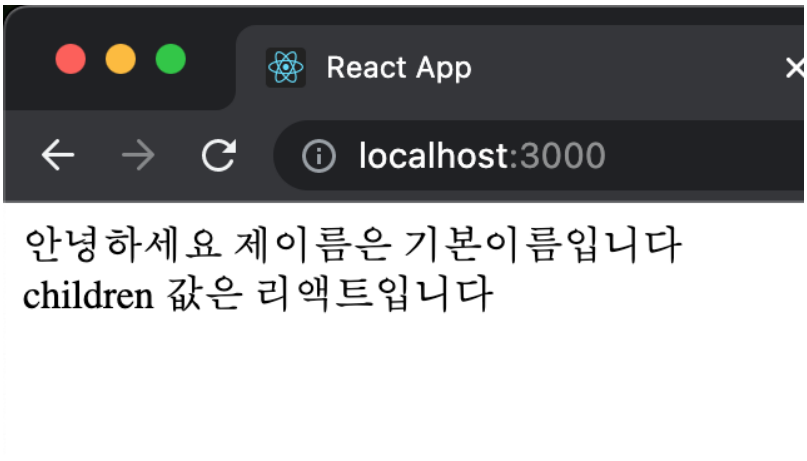
App.js M
MyComponent.js U
src > App.js > App
You, 12 seconds ago | 1 author (You)

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

function App() {
  return <MyComponent />;
}
export default App;
```

MyComponent.js U
src > MyComponent.js > name

```jsx
const MyComponent = (props) => {
  return <div>안녕하세요 제이름은 {props.name}입니다</div>;
};

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

export default MyComponent;
```

3.3.4 태그 사이의 내용을 보여주는 children




```
src > App.js > ...
You, 17 seconds ago | 1 author (You)
1  import React from 'react';
2  import MyComponent from './MyComponent';
3
4  function App() {
5    return <MyComponent>리액트</MyComponent>;
6  }
7  export default App;
8
```

```
MyComponent.js U
src > MyComponent.js > MyComponent
1  const MyComponent = (props) => {
2    return (
3      <div>
4        안녕하세요 제이름은 {props.name}입니다
5        <br />
6        children 값은 {props.children}입니다참
7      </div>
8    );
9  };
10
11 MyComponent.defaultProps = {
12   name: '기본이름',
13 };
14
15 export default MyComponent;
16
```

3.3.5 비구조화 할당 문법을 통해 props 내부 값 추출하기

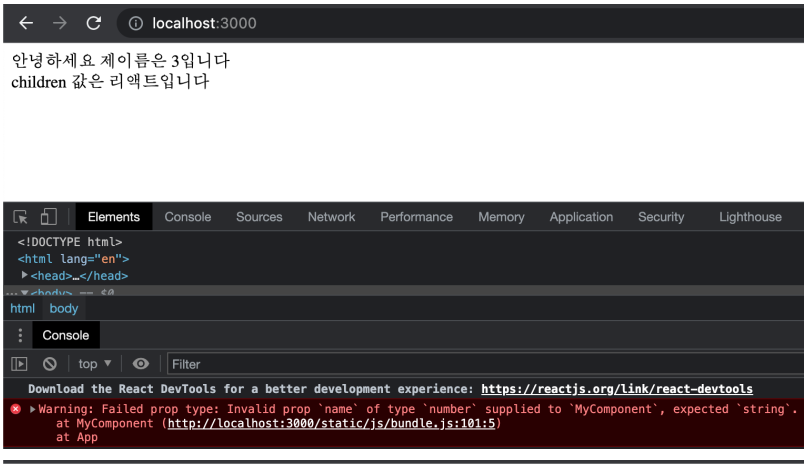
```
const json = { a: 1, b: 2, c: 3 };

const a = json.a;
const b = json.b;
const c = json.c;

const {a, b, c} = json;
```

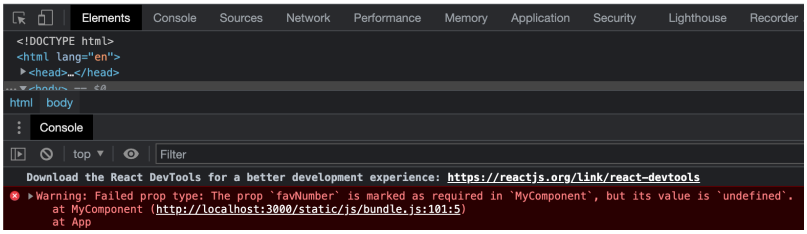
props로 전달되는 값을 비구조 할당으로 받으면 편리하다. 실습결과

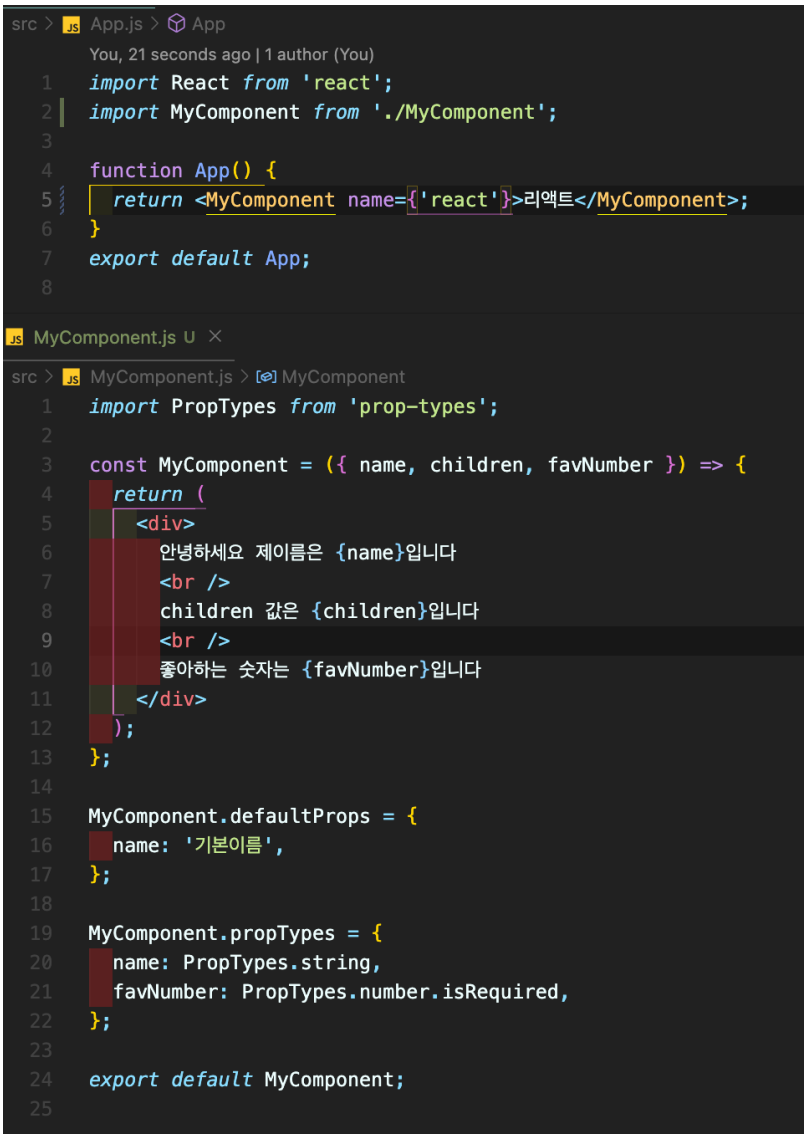
3.3.6 propTypes를 통한 props 검증

컴포넌트의 필수 props를 지정할 때나 데이터타입을 지정할 때 사용 propTypes의 경우 import 구문을 통해 불러와야 사용이 가능 import PropTypes from 'prop-types';

- 컴포넌트 하단에 컴포넌트명.propTypes = { name: PropTypes.string } 의 형식으로 사용
- 설정한 type과 다른 type의 props가 전달될 경우 Console탭에 에러를 띄움
- json으로 전달하는 propTypes의 value 끝에 .isRequired를 붙일 경우, props로 전달되지 않을 때 경고를 출력







```
src > App.js > App
You, 21 seconds ago | 1 author (You)
import React from 'react';
import MyComponent from './MyComponent';

function App() {
  return <MyComponent name={'react'}>리액트</MyComponent>;
}
export default App;
```

```
MyComponent.js U
src > MyComponent.js > MyComponent
import PropTypes from 'prop-types';

const MyComponent = ({ name, children, favNumber }) => {
  return (
    <div>
      안녕하세요 제이름은 {name}입니다
      <br />
      children 값은 {children}입니다
      <br />
      좋아하는 숫자는 {favNumber}입니다
    </div>
  );
};

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

MyComponent.propTypes = {
  name: PropTypes.string,
  favNumber: PropTypes.number.isRequired,
};

export default MyComponent;
```

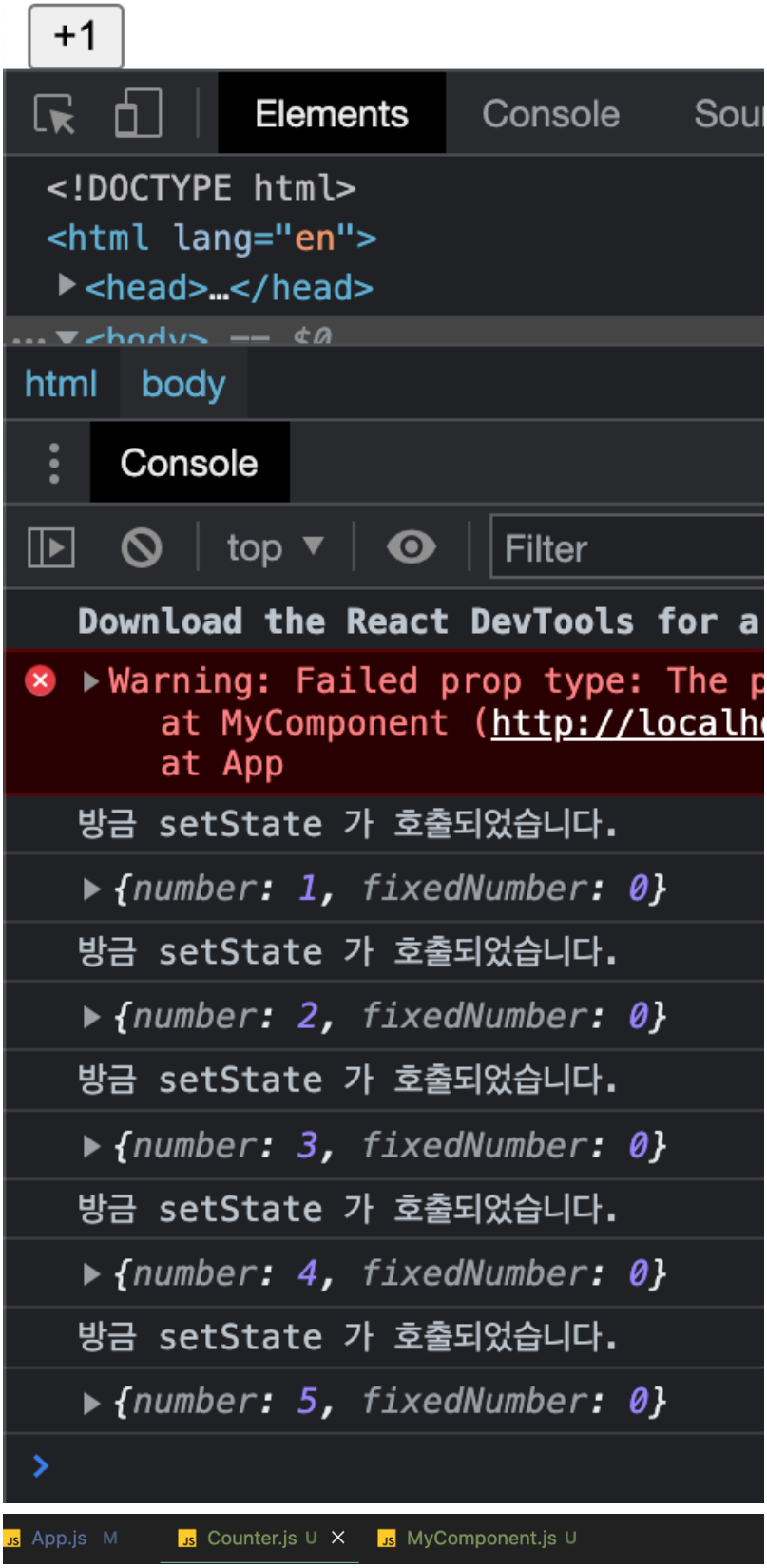
3.3.6.2 더많은 PropTypes의 종류

종류	설명
array	배열
bool	boolean값
func	함수
number	숫자
string	문자열
instanceOf(클래스)	특정 클래스의 인스턴스
oneOf(['dog','cat'])	주어진 배열 요소 중 하나
oneOfType([React.PropTypes.string, PropTypes.number])	주어진 배열 안의 type 중 하나
any	아무 종류

3.4 state

5

바뀌지 않는 값: 0


+1
Elements
Console
<!DOCTYPE html>
<html lang="en">
<head>…</head>
html
body
Console
top
Filter
Download the React DevTools for a
Warning: Failed prop type: The p
at MyComponent (http://localh
at App
방금 setState 가 호출되었습니다.
{number: 1, fixedNumber: 0}
방금 setState 가 호출되었습니다.
{number: 2, fixedNumber: 0}
방금 setState 가 호출되었습니다.
{number: 3, fixedNumber: 0}
방금 setState 가 호출되었습니다.
{number: 4, fixedNumber: 0}
방금 setState 가 호출되었습니다.
{number: 5, fixedNumber: 0}
App.js M
Counter.js U
MyComponent.js U

src > Counter.js > Counter > render

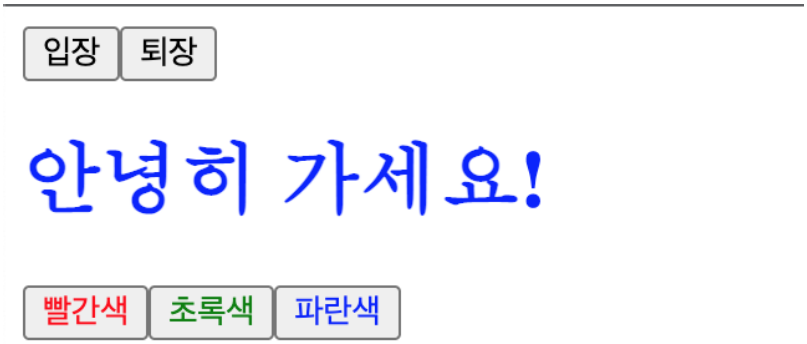
```jsx
import React, { Component } from 'react';

class Counter extends Component {
  state = {
    number: 0,
    fixedNumber: 0,
  };
  render() {
    const { number, fixedNumber } = this.state;
    return (
      <div>
        <h1>{number}</h1>
        <h2>바뀌지 않는 값: {fixedNumber}</h2>
        <button
          onClick={() => {
            this.setState(
              {
                number: number + 1,
              },
              () => {
                console.log('방금 setState 가 호출되었습니다
                console.log(this.state);
              }
            );
          }}
        >
          +1
        </button>
      </div>
    );
  }
}

export default Counter;
```

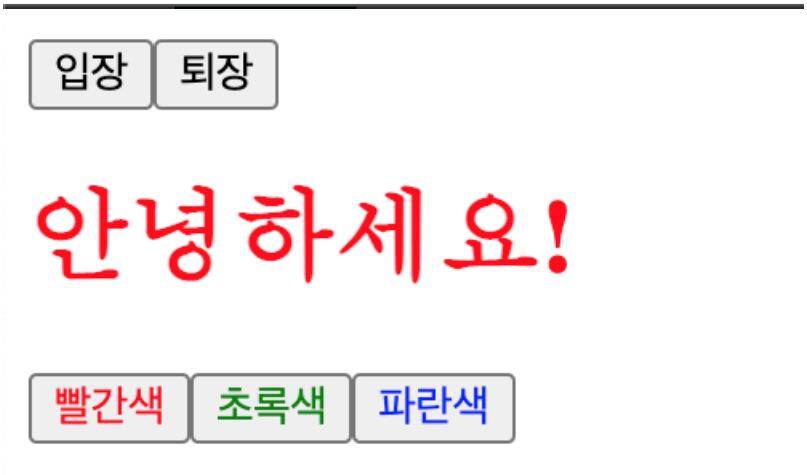
3.4.2.1 배열 비구조화 할당

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

3.4.2 함수 컴포넌트에서 useState 사용하기

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

const Say = () => {
  const [message, setMessage] = useState('');
  const onClickEnter = () => setMessage('안녕하세요!');
  const onClickLeave = () => setMessage('안녕히 가세요!');

  const [color, setColor] = useState('black');

  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;
```





> message:변수 setMessage : 변수에 대한 setter useState : 배열을 return ('')변수의 초기값 {message} : setter에 의해 값이 변경되면 실시간으로 화면이 갱신됨

- useState 함수의 인자에는 상태의 초깃값
- 값의 형태는 자유
- useState를 호출하면 배열이 반환, 배열의 첫번째 원소는 현재상태 두번째는 상태를 바꿔주는 함수 (setter 함수)
- 배열 비구조화 할당을 통해 이름을 자유롭게 정해 줄 수 있습니다
- useState는 한 컴포넌트에서 여러번 사용 가능

3.5 state를 사용할 때의 주의사항

state값을 바꿔야 할 때는 useState를 통해 리턴받은 setter함수를 반드시 사용해야 함 배열이나 객체를 state값으로 사용했을 경우에도 마찬가지

1. 배열이나 객체의 사본을 만든다.
2. 그 사본의 값을 업데이트 한다.

- 객체의 모든 내용을 복제

```
const object = {a: 1, b: 2, c: 3}
const nextObject = { ...object, b: 5};
```

3.6 정리

- props는 부모 컨포넌트가 설정하고
- state는 컴포넌트 자체적으로 지닌값
- 부모컴포넌트의 state를 자식 컴포넌트의 props로 전달하고
- 자식컴포넌트에서 특정 이벤트가 발생할때 부모 컴포넌트의 메서드를 호출 -> props 유동적 사용

4장 - 이벤트 핸들링

이벤트는 사용자가 웹 브라우저에서 DOM요소들과 상호작용 하는 것

4.1 리액트의 이벤트 시스템

HTML의 이벤트와 웹브라우저 인터페이스(사용법)가 동일하다.

4.1.1 이벤트를 사용할 때 주의사항

1. 이벤트 이름은 카멜 표기법(camelCase)으로 작성
2. 이벤트에 실행할 자바스크립트 코드가 아니라 함수 형태의 객체를 전달 callback으로 전달
3. 바로만들어서 전달(콜백형식) / 렌더링 부분 외부에 만들어(별도 함수연결) 전달
4. DOM 요소에만 이벤트 설정이 가능 직접 만든 component에는 이벤트를 자체적 설정불가 하지만 props로 전달받아 이를 컴포넌트 내부의 DOM이벤트로 설정은 가능

4.2.2 onChange 이벤트 핸들링하기

4.2.2.1 onChange 이벤트 설정

- 콘솔에 기록되는 e객체는 SyntheticEvent 웹브라우저의 네이티브 이벤트를 감싸는 객체
- 네이티브 이벤트와 인터페이스가 같으므로 순수 자바스크립트에서 HTML을 다루듯이 사용

4.2.2.2 state에 input 값 담기

1. 처음 input태그의 value는 message의 초깃값인 ''빈 문자열
2. onChange가 동작하여, 입력받는 값이 message에 setter로 전달됨
3. 입력값을 전달받은 message가 input태그의 value에 적용됨

4.2.2.3 버튼을 누를 때 comment 값을 공백으로 설정

Violation => 이벤트에 걸린 시간이 오래걸린다는 경고(not 오류) -> 콘솔 필터에서 '상세'를 끄면 안보임 프레임 드랍이 생길 수 있는 부분에 관한 경고메시지

1. 버튼을 누를 경우 message값을 alert로 띄워줌
2. setter를 이용하여 message값을 ''빈 문자열으로 바꿈
3. input의 value값이 빈 문자열으로 바뀌면서 리셋

4.2.3 임의 메서드(함수) 만들기

위에서 작성한 코드의 이벤트에 callback으로 전달하던 함수를 외부에 임의로 정의하여 호출

4.3 함수 컴포넌트로 구현해보기

```
const [form, setForm] = useState({
  username:'',
  message:''
})
```

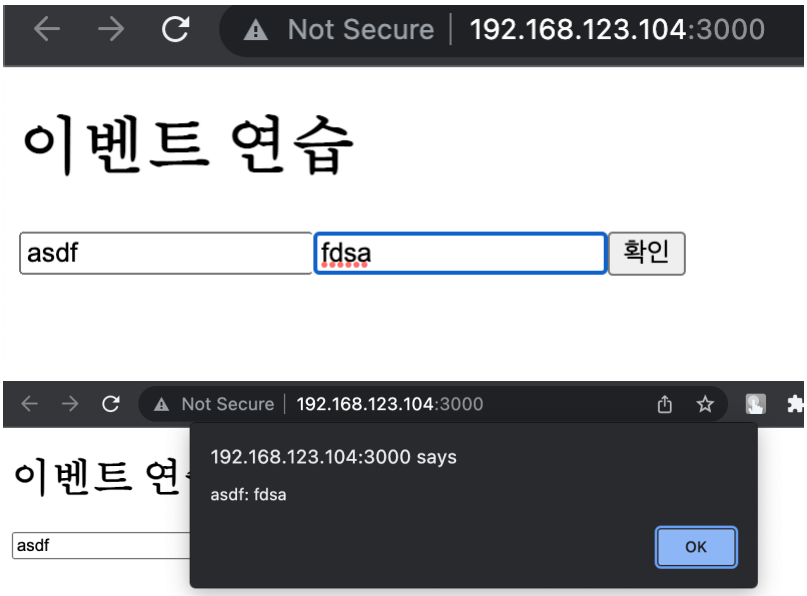
상태값이 JSON형태로 구성되어 있기 때문에 하나의 form이라는 값에 여러 하위값을 포함시킬수 있다.

```
const {username, message} = form;
const onChange = e => {
  ...form, //기존의 form 내용을 이자리에 복사한뒤
  [e.target.name]: e.target.value // 원하는 값 덮어 씌우기
};
setForm(nextForm);
```

원하는 값을 덮어 씌울때 써먹을수 있는 좋은 코드 !!!

4.4

자바스크립트를 안다면! 쉽게 다룰수 있다


Not Secure | 192.168.123.104:3000
이벤트 연습
asdf
fdsa
확인
Not Secure | 192.168.123.104:3000
192.168.123.104:3000 says
asdf: fdsa
OK
asdf

```

import React, { Component } from 'react';
class EventPractice extends Component {
  state = {
    username: '',
    message: '',
  };
  handleChange = (e) => {
    this.setState({
      [e.target.name]: e.target.value,
    });
  };
  handleClick = () => {
    alert(this.state.username + ': ' + this.state.message);
    this.setState({
      username: '',
      message: '',
    });
  };
  handleKeyPress = (e) => {
    if (e.key === 'Enter') {
      this.handleClick();
    }
  };
  render() {
    return (
      <div>
        <h1>이벤트 연습</h1>
        <input
          type="text"
          name="username"
          placeholder="유저명"
          value={this.state.username}
          onChange={this.handleChange}
        />
        <input
          type="text"
          name="message"
          placeholder="아무거나 입력해보세요"
          value={this.state.message}
          onChange={this.handleChange}
          onKeyPress={this.handleKeyPress}
        />
        <button onClick={this.handleClick}>확인</button>
      </div>
    );
  }
}
export default EventPractice;

```

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

const EventPractice = () => {
  const [form, setForm] = useState({
    username: '',
    message: '',
  });
  const { username, message } = form;
  const onChange = (e) => {
    setTimeout(() => console.log(e), 500);
```

```
    const nextForm = {
      ...form,
      [e.target.name]: e.target.value,
    };
    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="유저명"
        value={username}
        onChange={onChange}
      />
      <input
        type="text"
        name="message"
        placeholder="아무거나 입력해보세요"
        value={message}
        onChange={onChange}
        onKeyPress={onKeyPress}
      />
      <button onClick={onClick}>확인</button>
    </div>
  );
};
export default EventPractice;

```