4. 이벤트 핸들링

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

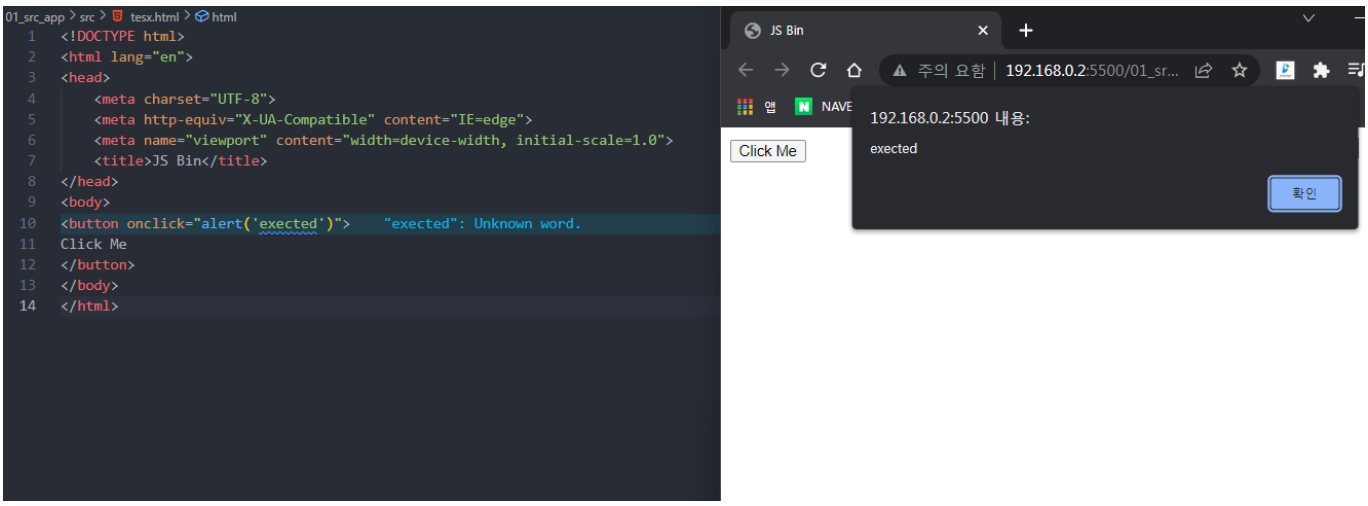
4.0 HTML에서 DOM 요소에 이벤트 설정

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">
    <meta name="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 리액트의 이벤트 시스템

리액트의 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>
(...)
```

4.1.1 이벤트 사용시 주의사항

1. 이벤트 이름은 카멜 표기법으로 작성되야함 ⇒ 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.

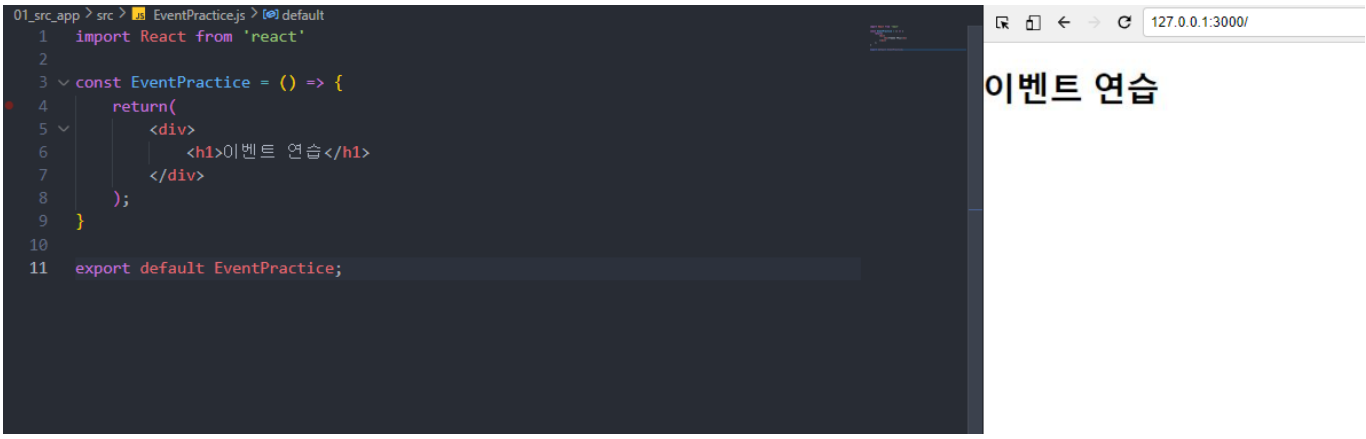
4.2 예제로 이벤트 핸들링 익히기

4.2.1 컴포넌트 생성 및 불러오기

4.2.1.1 컴포넌트 생성

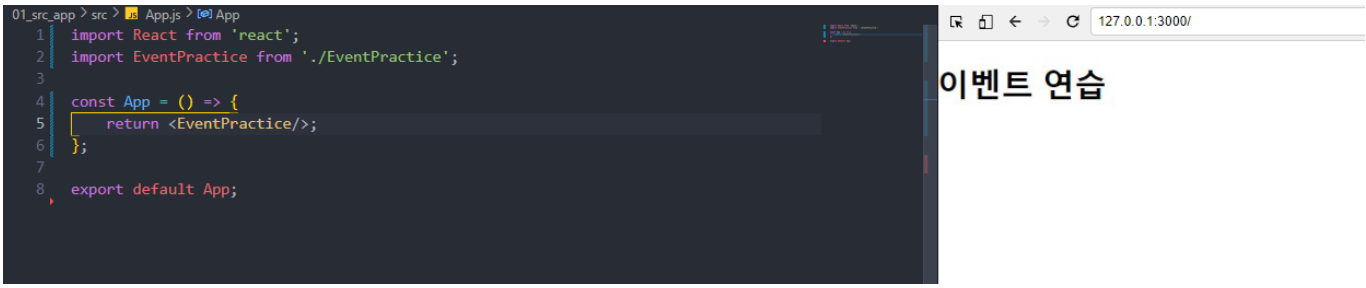
EventPractice.js

```
import React from 'react'

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

export default EventPractice;
```



App.js

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

export default App;
```

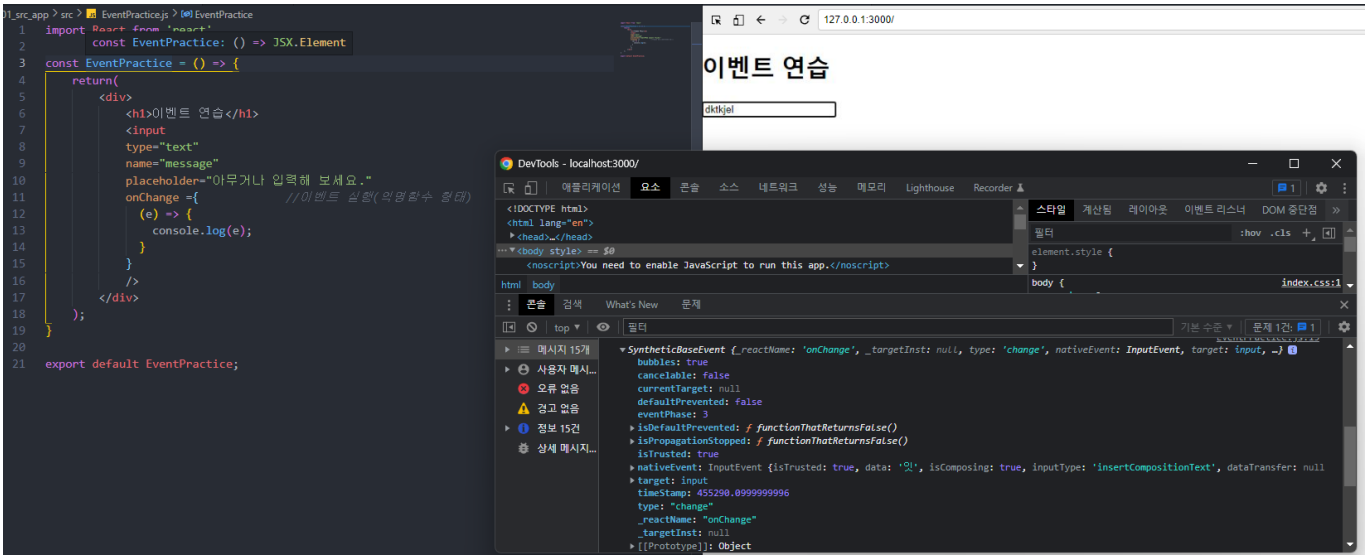


4.2.2 onChange 이벤트 핸들링하기

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>ㅇ
        );
    }

    export default EventPractice;
```



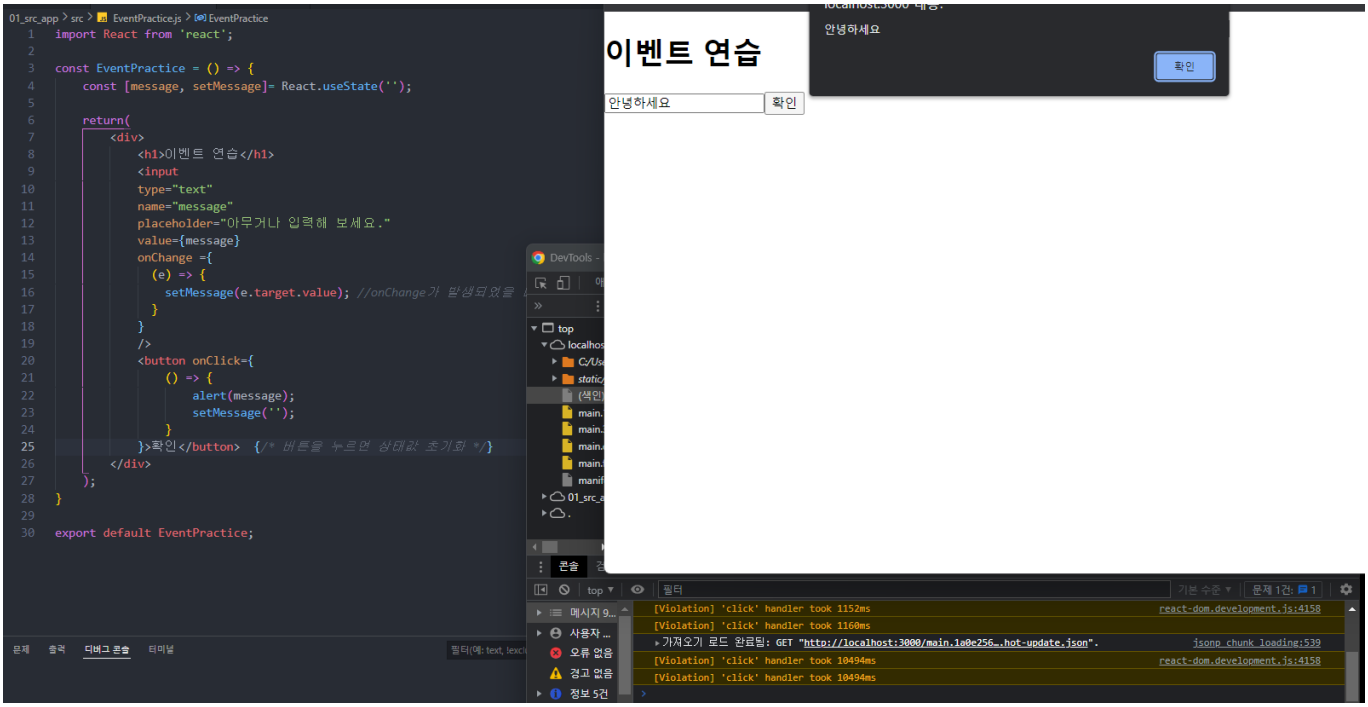
EventPractice.js

```
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
됨.
              (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);
              }
            }
            />
            <button onClick={
                () => {
                    alert(message);
                    setMessage(''); {/*변경된 setter 값을 변수에 대한 setter에 대입
*/}
                }
            }>확인</button>  {/* 버튼을 누르면 상태값 초기화 */}
        </div>
    );
}
```

```
export default EventPractice;
```

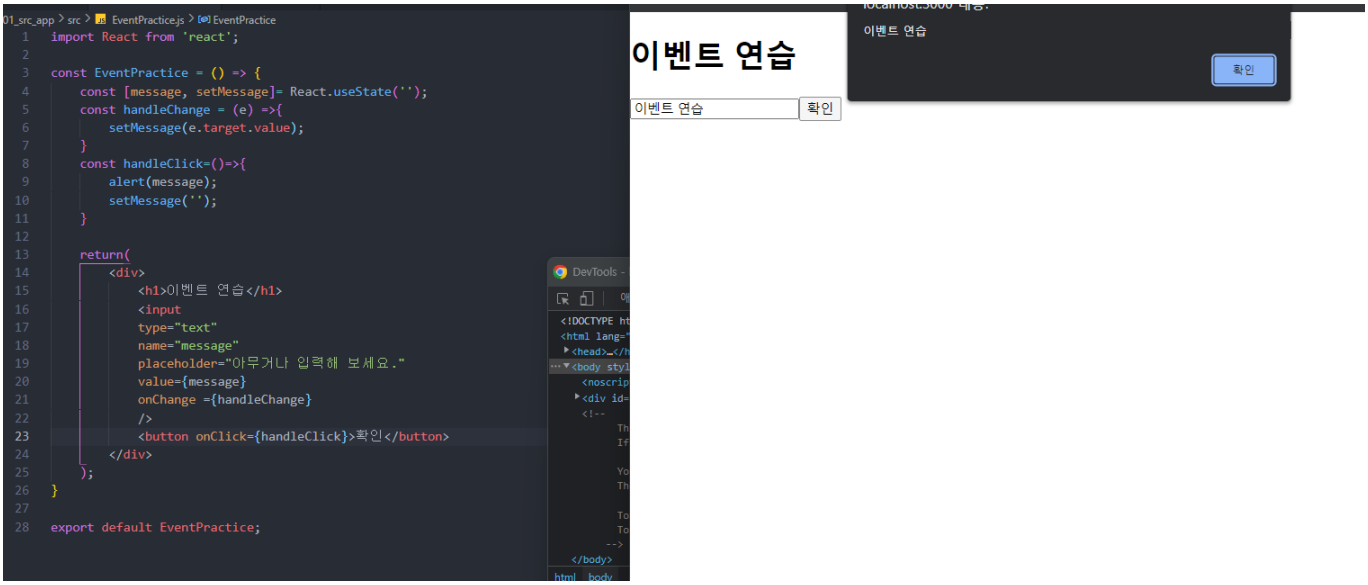


4.2.3 임의 메서드 만들기

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;
```

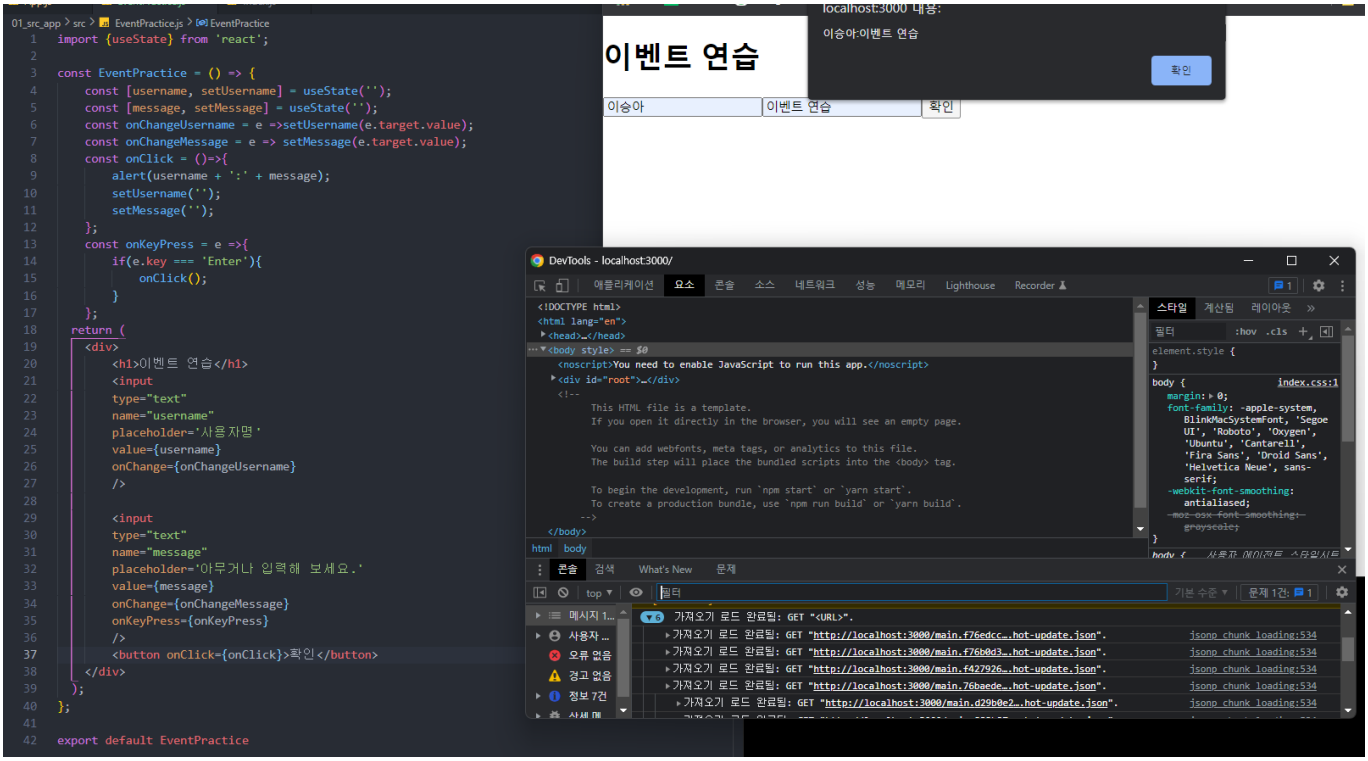


4.3 함수 컴포넌트로 구현해 보기

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}
            onKeyPress={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;
    /**
        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;
```