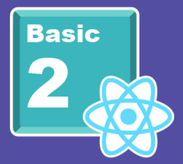
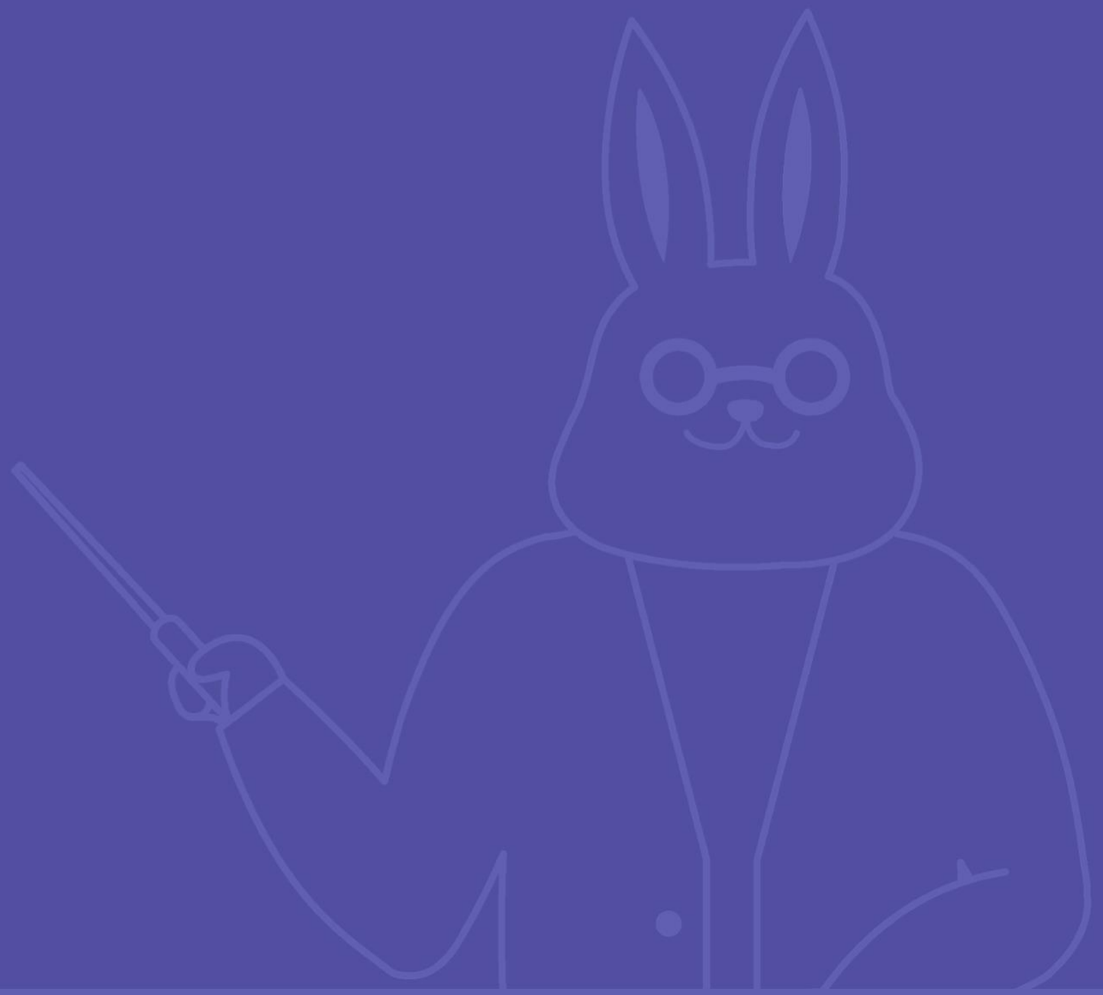


React 기초 II

02 이벤트 처리



/* elice */

목차

01

이벤트 소개

이벤트란

클릭!

```
function onClick(event) {
  ...
}
```

이벤트(event)란 웹 브라우저가 알려주는 HTML 요소에 대한 사건의 발생을 의미합니다.
유저의 행동에 의해 발생할 수도 있으며 개발자가 의도한 로직에 의해 발생할 수도 있습니다.
이렇게 발생된 이벤트를 자바스크립트를 이용해 대응할 수 있습니다.

Element가 로딩되었을 때, 사용자가 Element를 클릭했을 때, 마우스를 올렸을 때, 더블 클릭했을 때,
키보드 입력을 주었을 때 등 다양한 이벤트가 존재합니다.

이벤트 핸들러 함수에서는 다양한 로직을 처리하고 그 결과를 사용자에 출력하여 알릴 수도 있습니다.

이벤트 처리(핸들링) 방법

코드

```
const App = () => {
  const handleClick = () => {
    alert("클릭했습니다.");
  }
  return  (
    <div>
      <button onClick={handleClick}>클릭하세요</button>
    </div>
  );
};
```

핸들링 함수 선언

코드

```
const App = () => {
  return  (
    <div>
      <button onClick={() => { alert('클릭했습니다.') }
        }>클릭하세요</button>
    </div>
  )
}
```

익명 함수로 처리

React에서 이벤트를 처리하는 방법은 크게 두 가지 방법이 있습니다.
별도의 핸들링 함수를 선언하고 Element에 넘겨주는 방법과
이벤트를 할당하는 부분에서 익명 함수를 작성하는 방법으로 나뉩니다.

이벤트 객체

코드

```
const App = () => {
  const handleChange = (event) => {
    console.log(event.target.value)
                + "라고 입력하셨네요.");
  }
  return (
    <div>
      <input onChange={handleChange} />
    </div>
  );
};
```

- DOM Element의 경우 핸들링 함수에 이벤트 object를 매개변수로 전달합니다.

- 이벤트 object를 이용하여 이벤트 발생 원인, 이벤트가 일어난 Element에 대한 정보를 얻을 수 있습니다.

- 이벤트 형태(클릭, 키 입력 등)와 DOM 종류(button, form, input 등)에 따라 전달되는 이벤트 object의 내용도 다르니 유의하세요.

- 참고: https://developer.mozilla.org/ko/docs/Web/API/Event

많이 쓰이는 DOM 이벤트

onClick: Element를 클릭했을 때

onChange: Element의 내용이 변경되었을 때(input의 텍스트를 변경, 파일 선택 등)

onKeyDown, onKeyUp, onKeyPress: 키보드 입력이 일어났을 때

onDoubleClick: Element를 더블 클릭했을 때

onFocus: Element에 Focus되었을 때

onBlur: Element가 Focus를 잃었을 때

onSubmit: Form Element에서 Submit 했을 때

02

컴포넌트 내 이벤트 처리

DOM 버튼 클릭

코드

```
const App = () => {
  const handleClick = () => {
    alert("클릭했습니다.");
  }
  return (
    <div>
      <button onClick={handleClick}>클릭하세요</button>
    </div>
  );
};
```

DOM element의 클릭 이벤트를
전달받아 처리하는 간단한 예제입니다.

DOM Input 값을 State에 저장하기

코드

```
const App = () => {
  const [inputValue, setInputValue] =
useState("defaultValue");

  const handleChange = (event) => {
    setInputValue(event.target.value);
  }

  return (
    <div>
      <input onChange={hadleChange}
defaultValue={inputValue} />
      <br />
      입력한 값은: {inputValue}
    </div>
  );
};
```

event object의 **target**은 이벤트의 원인이 되는 Element를 가리킵니다.

현재 event의 target은 input element이므로 입력된 value를 가져와 setState를 하는 모습입니다.

여러 Input 동시에 처리하기

코드

```
const App = () => {
  const [user, setUser] = useState({ name; "민수",
school: "엘리스대학교" });

  const handleChange = (event) => {
    const { name, value } = event.target;
    const newUser = { ...user };
    newUser[name] = value;
    setUser(newUser);
  };

  return (
    <div>
      <input name="name" onChange={handleChange}
value={user.name} />
      <br />
      <input name="school" onChange={handleChange}
value={user.school} />
      <p>
        {user.name}님은 {user.school}에
        재학중입니다.
      </p>
    </div>
  );
};
```

State를 여러 개 선언할 수도 있지만 **object**를 활용하여 여러 개의 input을 state로 관리하는 방법이 있습니다.

target으로부터 **name**을 받아와 해당 **name**의 **key**에 해당하는 **value**를 변경하여 state에 반영합니다.

03

다른 컴포넌트로 이벤트 전달

컴포넌트간 이벤트 전달하기

코드

```
const MyForm = ({ onChange }) => {
  return (
    <div>
      <span>이름: </span>
      <input onChange={onChange} />
    </div>
  )
}
```

사용자가 입력한 정보를 현재 컴포넌트가 아닌 부모 컴포넌트에서 활용해야 하는 경우
예시와 같이 이벤트를 Props로 전달하여 처리할 수 있습니다.

```
const App = () => {
  const [username, setUsername] = useState('')
  return (
    <div>
      <h1>{username}님 환영합니다.</h1>
      <MyForm>
        onChange={(event) => {
          setUsername(event.target.value)
        }}
      />
    </div>
  )
}
```

커스텀 이벤트

코드

```
const SOS = ({onSOS}) => {
  const [count, setCount] = useState(0);
  const handleClick = () => {
    if(count >= 2) {
      onSOS();
    }
    setCount(count + 1);
  }
  return <button onClick={handleClick}>
    세 번 누르면 긴급호출({count})
  </button>
}
```

코드

```
const App = () => {
  return (
    <div>
      <SOS
        onSOS={() => {
          alert("긴급사태!");
        }}
      />
    </div>
  );
};
```

단순히 DOM 이벤트를 활용하는 것을 넘어서 나만의 이벤트를 만들 수도 있습니다.

이벤트 명명법

코드

```
const App = () => {
  const handleClick = () => {
    alert("클릭했습니다.");
  }
  return (
    <div>
      <button onClick={handleClick}>클릭하세요</button>
    </div>
  );
};
```

직접 이벤트를 만들 때에는
이름을 자유롭게 설정할 수 있습니다.

그러나 보통은 코드를 읽을 때 쉽고 빠르게 이해할 수 있도록 **“on” + 동사** 또는 **“on” + 명사 + 동사** 형태로 작성합니다.

예시: onClick, onButtonClick, onInputChange

핸들링 함수의 경우 마찬가지로
“handle” + 동사 또는 **“handle” + 명사 + 동사**
의 형태로 작성하며, “handle” 대신 이벤트명과 동일한 “on”을 앞에 붙여도 무방합니다.

크레딧

/* elice */

코스 매니저
강윤수

콘텐츠 제작자
마로

강사
마로

감수자
장석준

디자이너
강혜정

연락처

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io