

# Chapter 8. 이벤트 핸들링

교재: 처음만난 리액트 (저자: 이인제, 한빛출판사)



# Contents

- CHAPTER 8: 이벤트 핸들링

8.1 이벤트 처리하기

8.2 Arguments 전달하기

8.3 (실습) 클릭 이벤트 처리하기

## SECTION 8.1 이벤트 처리하기

### ◦ 이벤트(Events)

- 브라우저에서 사용자의 조작이나 환경의 변화로 벌어진 사건  
ex. 사용자가 버튼을 클릭한 사건 – 버튼 클릭 이벤트

### ◦ 이벤트 핸들링

- 다양한 이벤트를 원하는 대로 처리하는 것

### ◦ 이벤트 핸들러(Event Handler)

- 어떤 이벤트가 발생했을 때 해당 이벤트를 처리하는 함수
- 이벤트 리스너(Listener)라고도 함

## SECTION 8.1 이벤트 처리하기

### ◦ DOM의 Event

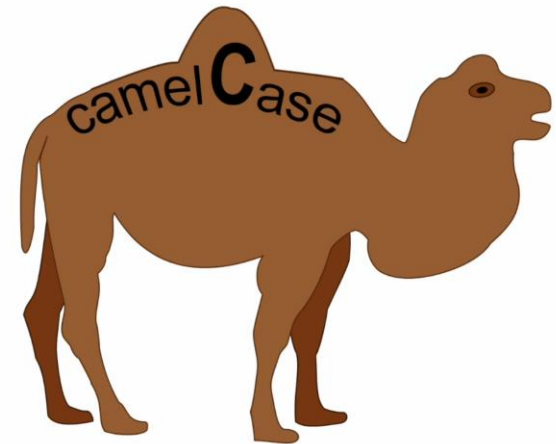
```
1 <button onclick="activate( )">  
2   Activate  
3 </button>
```

- 함수를 문자열로 전달

### ◦ 리액트의 Event

```
1 <button onClick={activate}>  
2   Activate  
3 </button>
```

- 카멜 표기법 사용
- 함수 그대로 전달



[https://en.wikipedia.org/wiki/Camel\\_case](https://en.wikipedia.org/wiki/Camel_case)

## SECTION 8.1 이벤트 처리하기

- 클래스 컴포넌트의 이벤트 처리
  - bind() 함수를 사용하는 방법

```
1 import React from "react";
2
3 class Toggle extends React.Component {
4   constructor(props) {
5     super(props);
6
7     this.state = { isToggleOn: true };
8
9     this.handleClickButton = this.handleClick.bind(this);
10  }
11
12  handleClick() {
13    this.setState(prevState => ({
14      isToggleOn: !prevState.isToggleOn
15    }));
16  }
```

```
17
18   render() {
19     return (
20       <button onClick={this.handleClickButton}>
21         {this.state.isToggleOn ? 'On' : 'Off'}
22       </button>
23     )
24   }
25 }
26
27 export default Toggle;
```

setState(updater 함수) – 이전 state 값에 접근할 수 있음  
prevState: 이전 state

## SECTION 8.1 이벤트 처리하기

- 클래스 컴포넌트의 이벤트 처리
  - 클래스 필드 문법 사용

```
1 class MyButton extends React.Component {  
2   handleClick = () => {  
3     console.log('this is:', this);  
4   }  
5  
6   render() {  
7     return (  
8       <button onClick={this.handleClick}>  
9         클릭  
10      </button>  
11    );  
12  }  
13 }
```

\* 클래스 필드 -

클래스 블록 안에서 할당 연산자(=)를 이용해  
인스턴스 속성을 지정할 수 있는 문법

화살표 함수 안에서의 this 키워드는 바로  
바깥쪽 scope에 존재하는 this와 같은 객체를  
가리킴

## SECTION 8.1 이벤트 처리하기

- 클래스 컴포넌트의 이벤트 처리
  - 이벤트 핸들러에 화살표 함수 사용

```
1 class MyButton extends React.Component {
2   handleClick() {
3     console.log('this is:', this);
4   }
5
6   render() {
7     // 이렇게 하면 `this`가 바운드됩니다.
8     return (
9       <button onClick={() => this.handleClick()}>
10         클릭
11       </button>
12     );
13   }
14 }
```

컴포넌트가 렌더링될 때마다  
다른 콜백 함수를 생성  
- 성능 이슈로 권장하지 않음

# SECTION 8.1 이벤트 처리하기

## 함수 컴포넌트의 이벤트 처리

- 이벤트 핸들러를 함수로 정의하거나, 화살표 함수를 사용하여 정의

```
1 import React, { useState } from "react";
2
3 function ToggleFunc(props) {
4   const [isToggleOn, setIsToggleOn] = useState(true);
5
6   // 방법 1. 함수로 정의
7   function handleClick() {
8     setIsToggleOn((isToggleOn) => !isToggleOn);
9   }
10
11   // 방법 2. arrow function을 사용하여 정의
12   const handleClick2 = () => {
13     setIsToggleOn((isToggleOn) => !isToggleOn);
14   }
15
16   return (
17     <button onClick={handleClick}>
18       {isToggleOn ? 'On' : 'Off'}
19     </button>
20   )
21 }
22
23 export default ToggleFunc;
```

this를 사용하지 않음



## SECTION 8.2 Arguments 전달하기

- Arguments (매개 변수)
  - 함수에 전달할 데이터 (event handler에 전달할 데이터)

- 클래스 컴포넌트의 이벤트 예시

```
<button onClick={(event) => this.deleteItem(id, event)}>삭제하기</button>  
<button onClick={this.deleteItem.bind(this, id)}>삭제하기</button>
```

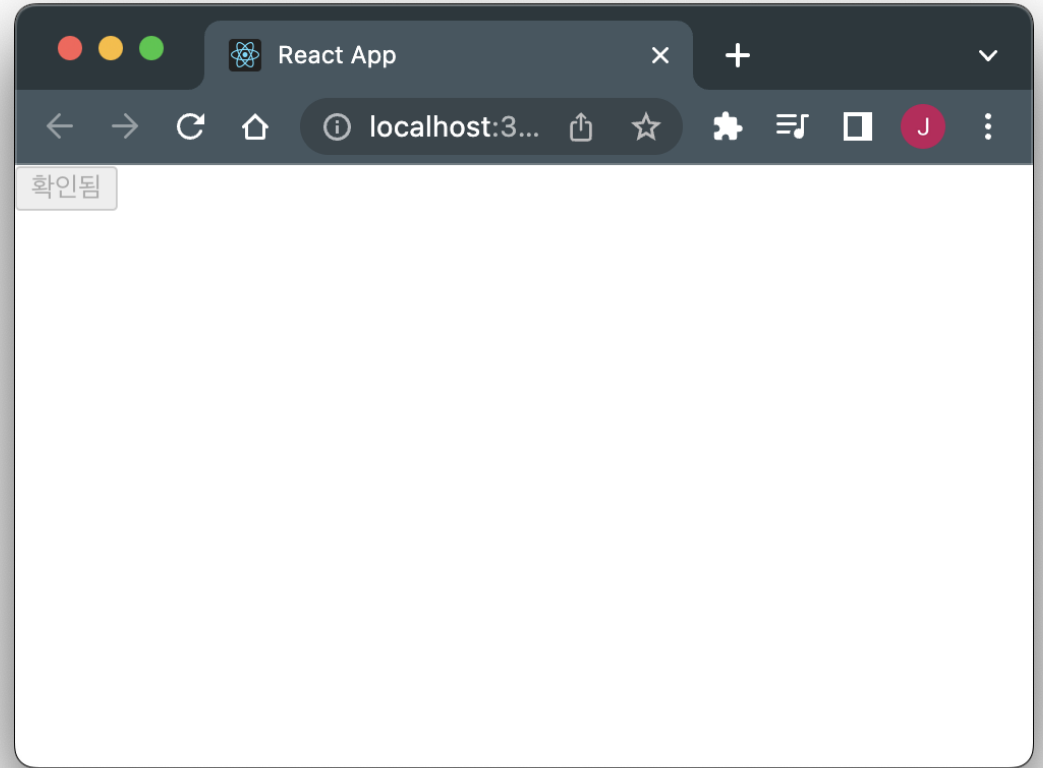
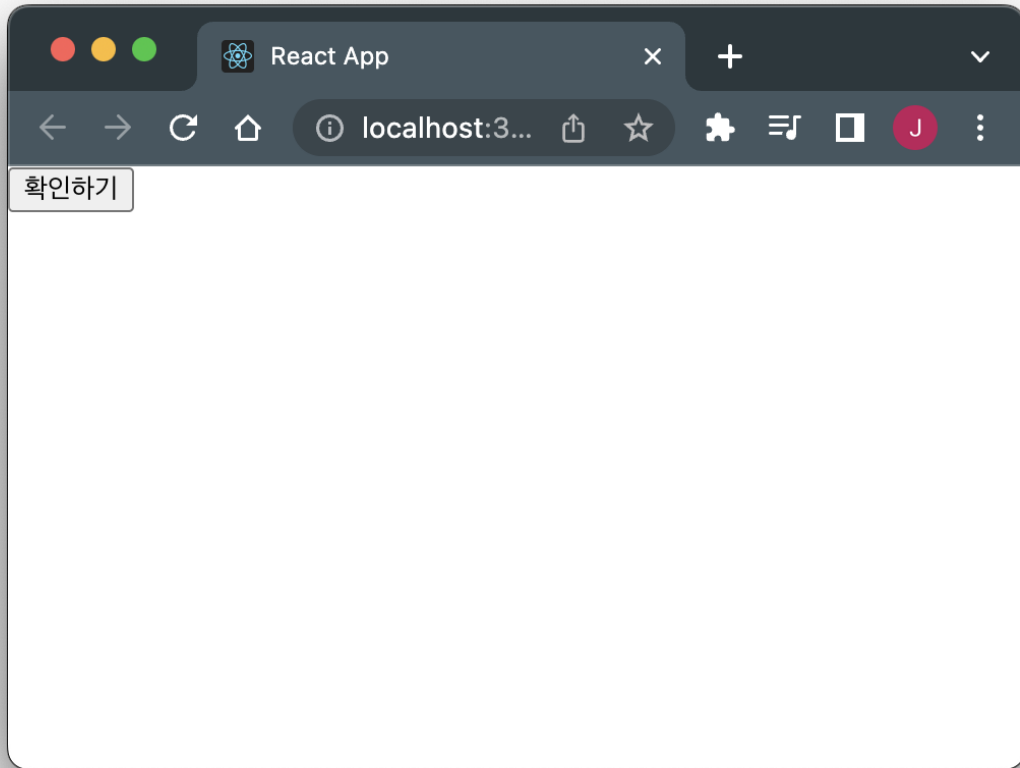
event 객체는 마지막 매개 변수로  
자동 전달됨

- 함수 컴포넌트의 이벤트 예시

```
1 function MyButton(props) {  
2   const handleDelete = (id, event) => {  
3     console.log(id, event.target);  
4   };  
5  
6   return (  
7     <button onClick={(event) => handleDelete(1, event)}>  
8       삭제하기  
9     </button>  
10  );  
11 }
```

## PRACTICE 8.3 클릭 이벤트 처리하기

- 클릭 이벤트를 처리하는 컴포넌트 만들기
  - 클릭 이벤트 발생 시 버튼 텍스트와 disabled 속성을 변경



## PRACTICE 8.3 클릭 이벤트 처리하기

- 클래스 컴포넌트 - ConfirmButton.jsx
  - bind() 함수 사용

```
1 import React from "react";
2
3 class ConfirmButton extends React.Component {
4   constructor(props) {
5     super(props);
6
7     this.state = {
8       isConfirmed: false,
9     };
10
11     this.handleClick = this.handleClick.bind(this);
12   }
13
14   handleClick() {
15     this.setState((prevState) => ({
16       isConfirmed: !prevState.isConfirmed,
17     }));
18   }
19 }
```

```
19
20 render() {
21   return (
22     <button onClick={this.handleClick}
23       disabled={this.state.isConfirmed}>
24       {this.state.isConfirmed ? '확인됨' : '확인하기'}
25     </button>
26   )
27 }
28 }
29
30 export default ConfirmButton;
```

## PRACTICE 8.3 클릭 이벤트 처리하기

- 클래스 컴포넌트 - ConfirmButton.jsx
  - 클래스 필드 문법 사용

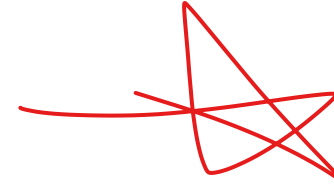
```
1 import React from "react";
2
3 class ConfirmButton extends React.Component {
4   constructor(props) {
5     super(props);
6
7     this.state = {
8       isConfirmed: false,
9     };
10  }
11
12  handleConfirm = () => {
13    this.setState((prevState) => ({
14      isConfirmed: !prevState.isConfirmed,
15    }));
16  }
```

bind 코드 제거  
Arrow function  
으로 변경

```
19
20   render() {
21     return (
22       <button onClick={this.handleConfirm}
23         disabled={this.state.isConfirmed}>
24         {this.state.isConfirmed ? '확인됨' : '확인하기'}
25       </button>
26     )
27   }
28 }
29
30 export default ConfirmButton;
```

## PRACTICE 8.3 클릭 이벤트 처리하기

함수 컴포넌트 - ConfirmButtonFunc.jsx



```
1 import React, { useState } from "react";
2
3 function ConfirmButton(props) {
4   const [isConfirmed, setIsConfirmed] = useState(false);
5
6   const handleConfirm = () => {
7     setIsConfirmed((prevIsConfirmed) => !prevIsConfirmed);
8   };
9
10  return (
11    <button onClick={handleConfirm} disabled={isConfirmed}>
12      {isConfirmed ? '확인됨' : '확인하기'}
13    </button>
14  )
15 }
16
17 export default ConfirmButton;
```

## [요약]

- 이벤트란?
  - 사용자가 버튼을 클릭하는 등의 사용자의 조작이나 환경의 변화로 벌어진 사건
- 이벤트 처리하기
  - DOM의 이벤트
    - 이벤트의 이름을 모두 소문자로 표기
    - 이벤트를 처리할 함수를 문자열로 전달
  - 리액트의 이벤트
    - 이벤트의 이름을 카멜표기법으로 표기
    - 이벤트를 처리할 함수를 그대로 전달
  - 이벤트 핸들러(이벤트 리스너)
    - 이벤트가 발생했을 때 해당 이벤트를 처리하는 함수

## [요약]

- 이벤트 핸들러
  - 클래스 컴포넌트
    - 클래스의 함수로 정의하고 생성자에서 바인딩해서 사용
    - 클래스 필드 문법도 사용 가능
  - 함수 컴포넌트
    - 함수 안에 함수로 정의하거나 arrow function을사용해서 정의
- Arguments 전달하기
  - Arguments란
    - 함수에 전달할 데이터
    - 매개변수 또는 파라미터로 부르기도 함
  - 클래스 컴포넌트
    - arrow function을사용하거나 Function.prototype.bind를 사용해서 전달
  - 함수 컴포넌트
    - 이벤트 핸들러 호출 시 원하는 순서대로 매개변수를 넣어서 사용