04. Event

- 웹브라우저에서 DOM과 상호작용하는 것을 이벤트라고 한다.
- 리액트의 이벤트는 웹브라우저의 HTML이벤트와 인터페이스, 사용법이 유사

4.1 리액트 이벤트 시스템

4.1.1 이벤트사용시 주의사항

1. 이벤트명은 카멜 표기법으로 작성
2. 이벤트에 실행할 자바스크립트 코드를 전달하는 것이 아니라 `함수형태의 값을 전달` 한다.
3. DOM 요소에만 이벤트를 설정할 수 있다.
   - `직접 만든 컴퍼넌트에는 이벤트를 설정할 수 없다.`

4.1.2 이벤트종류

이벤트 명	JSX DOM 이벤트 프로퍼티	이벤트 호출 시점
click	onClick	엘리먼트에 마우스나 키보드가 클릭 된 경우
change	onChange	엘리먼트의 내용이 변경된 경우
submit	onSubmit	폼의 데이터가 전송될 때
keydown	onKeyDown	키보드 버튼이 눌린 경우 (값 입력전에 발생하며, shift, alt, ctrl 등 특수키에 동작한다.) (한/영, 한자 등은 인식불가)
keyup	onKeyUp	키보드 버튼이 눌렸다 뗀 경우 (값 입력후에 발생하며, onKeyDown 과 동일하게 동작한다.)
keypress	onKeyPress	키보드 버튼이 눌려져 있는 경우 (실제 글자가 작성될때 이벤트이며, ASCII 값으로 사용되어 특수키를 인식 못한다.)
focus	onFocus	엘리먼트가 포커스 된 경우
blur	onBlur	엘리먼트가 포커스가 사라진 경우
mousemove	onMouseMove	엘리먼트 위에서 마우스 커서가 움직일 때
mousedown	onMouseDown	마우스 버튼이 클릭되기 시작할 때
mouseup	onMouseUp	마우스 버튼 클릭이 끝날때

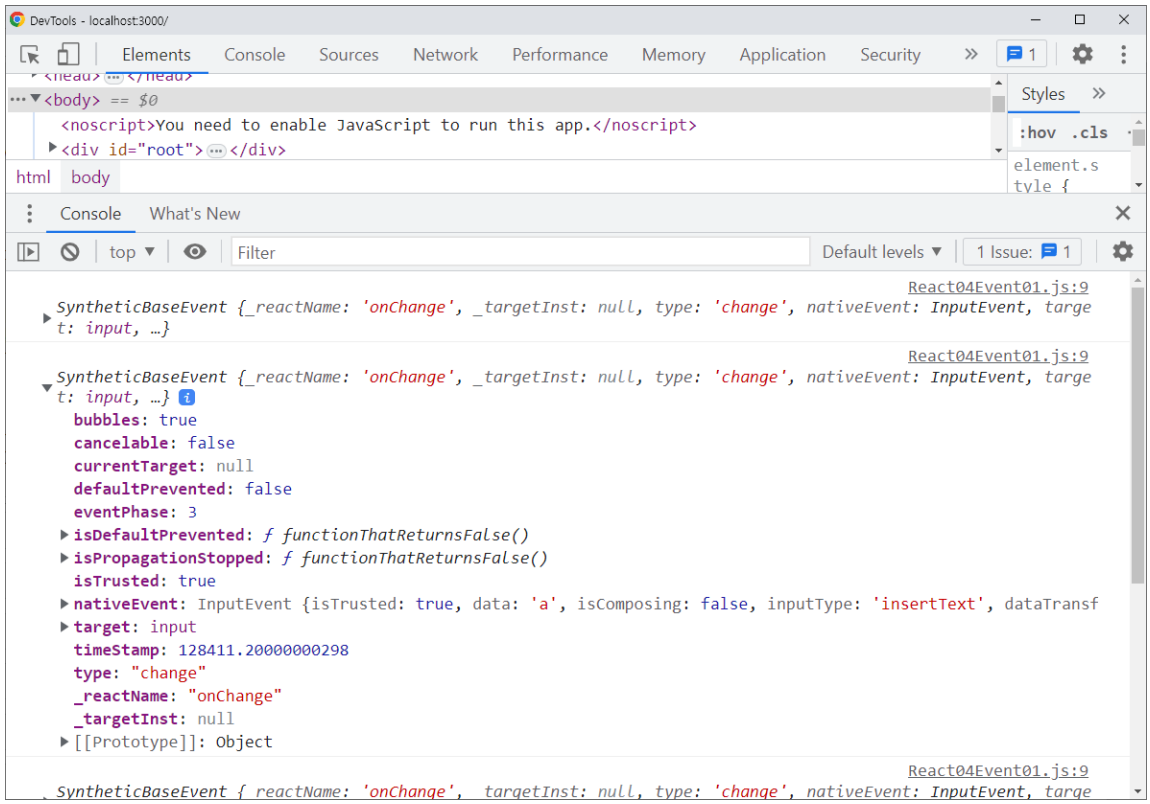
- 참고 : https://ko.reactjs.org/docs/events.html

4.2 이벤트핸들링

4.2.1 onChange

```
src/App.js
import "./App.css";
import React04Event01 from "./mysrc/React04Event01";

function App() {
```

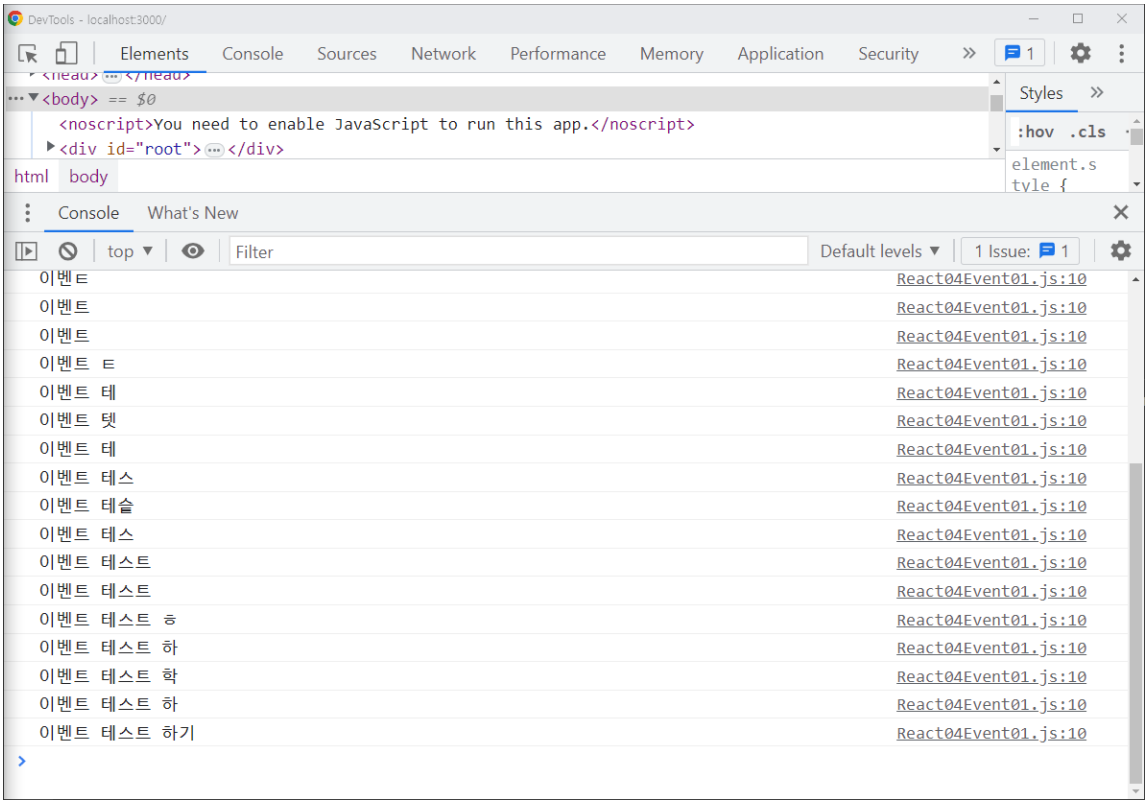
```
    return (
      <>
        <React04Event01 />
      </>
    );

  }
  export default App;

  src/mysrc/React04Event01.js
  import React, { Component } from 'react';

  class React04Event01 extends Component {
    render() {
      return (
        <div>
          <h1>onChange Event</h1>
          <input type="text" name="message" placeholder='onChange
  Test...'
            onChange={e => console.log(e)}
          />
        </div>
      );
    }
  }

  export default React04Event01;
```



- 콘솔에 기록되는 `e객체는 합성 이벤트(SyntheticEvent)로 웹브라저의 네이티브 이벤트를 감싸는 객체` 이다.
- 네이티브 이벤트와 인터페이스가 같으므로 순수 자바스크립트에서 HTML이벤트와 동일하게 사용하면 된다.

- `SyntheticEvent`는 네이티브 이벤트와 달리 이벤트가 종료되면 이벤트가 초기화 되기 때문에 정보를 참조할 수 없다.
- 만약, 비동기적으로 이벤트 객체를 참조하려면 `e.persist()`함수를 호출해야 한다

```
//  onChange={e => console.log(e)}를 아래와 같이 수정해 보기
    onChange={e => console.log(e.target.value)}
```



4.2.2 state에 input값 저장하기

```
src/mysrc/React04Event01.js
import React, { Component } from 'react';

class React04Event01 extends Component {

  state = {
    message: ''
  }

  render() {
    return (
      <div>
        <h1>onChange Event</h1>
        <input type="text" name="message" placeholder='onChange
Test...'
          value={this.state.message}
          onChange={e => {
              this.setState({message: e.target.value})
            }
          }
        />
        <br/>
        <button onClick={e => {
```

```
          alert(this.state.message);
          this.setState({
            message: ''
          })
        }}>메시지확인하기</button>
      </div>
    );
  }
}

export default React04Event01;
```

4.3 임의의 메서드 만들기

- 이벤트에는 실행할 자바스크립트 코드가 아니라 `함수를 전달`하기 때문에 이벤트를 처리할 때 랜더링을 하는 동시에 함수를 전달
- 함수를 사전에 정의하고 전달하는 것이 가독성이 높고 사용에 편리하다.

4.3.1 기본방식

```
src/mysrc/React04Event02.js
import React, { Component } from 'react';

class React04Event02 extends Component {
  // 1. 함수를 사전에 정의 - 기본방식
  state = {
    message: ''
  }

  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
    this.handleClick = this.handleClick.bind(this);
  }

  handleChange(e) {
    this.setState({
      message: e.target.value
    })
  }

  handleClick(e) {
    this.setState({
      message: ''
    })
  }

  render() {
    return (
      <div>
        <h3>1. 기본방식</h3>
        <input type="text" name="message" placeholder='onChange
Test...'
          value={this.state.message}
```

```
          onChange={this.handleChange}
        />
        <br/>
        <button onClick={this.handleClick}>메시지지우기</button>
      </div>
    );
  }
}

export default React04Event02;
```

- 함수가 호출될 때 호출하는 곳에 따라 this가 결정되기 때문에
- 클래스의 임의의 메서드가 특정 HTML요소의 이벤트로 등록되는 과정에서 메서드와 this의 관계가 끊어 진다.
- 이 때문에 임의의 메서드가 이벤트로 등록되어도 this가 컴퍼넌트 자신으로 가리키기 윟서는 메서드를 this와 binding해야 한다

  > - this.handleChange = this.handleChange.bind(this);
  > - this.handleClick = this.handleClick.bind(this);

- binding하지 않는 경우 this가 undefined를 가리키게 된다.

4.3.2 Property Initializer Syntax를 사용한 메서드

- 메서드 바인딩은 생성자에서 하는 거이 정석이지만 이 작업은 메서드를 만들 때 마다 생성자를 수정해야 하는 불편한 점도 있다.
- 이 작업을 간단하게 하는 방법은 Babel의 transform-class-properties 문법을 사용하여 화살표 함수 형태로 정의하는 것이다.

```
src/mysrc/React04Event03.js
import React, { Component } from 'react';

class React04Event03 extends Component {
  // 2. 화살표 문법
  state = {
    message: ''
  }

  handleChange = (e) => {
    this.setState({
      message: e.target.value
    })
  }

  handleClick = e => {
    alert()
    this.setState({
      message: ''
    })
  }

  render() {
    return (
      <div>
```

```
        <h3>2. 화살표 함수</h3>
        <input type="text" name="message" placeholder='onChange
Test...'
          value={this.state.message}
          onChange={this.handleChange}
        />
        <br/>
        <button onClick={this.handleClick}>메시지확인</button>
      </div>
    );
  }
}

export default React04Event03;
```

4.3.3 input여러개 다루기

- input 갯수에 맞춰 메서드를 여러개 만드는 것이 아니라 `evetn`객체를 활용해 `e.target.name 값을 사용` 하면 된다.
- `e.target.name`은 `input`태그의 `name`속성을 의미한다.

```
src/mysrc/React04Event04.js
import React, { Component } from 'react';

class React04Event04 extends Component {
  // 2. 화살표 문법
  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: ''
    })
  }



  render() {
    return (
      <div>
        <h3>3. Multi input box</h3>
        <input type="text" name="userName" placeholder='user name
...'
          value={this.state.userName}
          onChange={this.handleChange}
        />
```

```
        <br/>
        <input type="text" name="message" placeholder='message
...'
          value={this.state.message}
          onChange={this.handleChange}
        />
        <br/>
        <button onClick={this.handleClick}>메시지확인</button>
      </div>
    );
  }
}

export default React04Event04;
```

4.3.4 onKeyPress

- message input box에서 `enter키로 onClick메서드 호출` 하기

```
src/mysrc/React04Event05.js
import React, { Component } from 'react';

class React04Event05 extends Component {
  // 2. 화살표 문법
  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: ''
    })
  }

  handleKeyDown = e => {
    if(e.key === 'Enter') {
      this.handleClick();
    }
  }

  render() {
    return (
      <div>
        <h3>4. onKeyDown Event</h3>
        <input type="text" name="userName" placeholder='user name
...'
          value={this.state.userName}
```

```
          onChange={this.handleChange}
        />
        <br/>
        <input type="text" name="message" placeholder='message
...'
          value={this.state.message}
          onChange={this.handleChange}
          onKeyDown={this.handleKeyDown}
        />
        <br/>
        <button onClick={this.handleClick}>메시지확인</button>
      </div>
    );
  }
}

export default React04Event05;
```

4.4 함수 컴퍼넌트 구현

4.4.1 문자열로 처리

src/mysrc/React04Event06.js

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

const React04Event06 = () => {

  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 onKeyDown = e => {
    if(e.key === 'Enter') {
      onClick();
    }
  }
  return (
    <div>
    <h3>5. functional Component</h3>
    <input type="text" name="userName" placeholder='user name ...'
      value={userName}
      onChange={onChangeUserName}
    />
    <br/>
    <input type="text" name="message" placeholder='message ...'
      value={message}
      onChange={onChangeMessage}
```

```
        onKeyDown={onKeyDown}
      />
      <br/>
      <button onClick={onClick}>메시지확인</button>
    </div>
    );
};

export default React04Event06;
```

4.4.2 객체로 처리

```
src/mysrc/React04Event07.js
import { useState }  from 'react';

const React04Event07 = () => {

  const [form, setForm] = useState({
    userName: '',
    message: ''
  });

  const {userName, message} = form;

  const onChange = e => {
    const nextForm = {
      ...form,
      [e.target.name]: e.target.value
    }
    setForm(nextForm);
  };

  const onClick = () => {
    alert(userName + " : " + message);
    setForm({
      userName: '',
      message: ''
    })
  }

  const onKeyDown = e => {
    if(e.key === 'Enter') {
      onClick();
    }
  }
  return (
    <div>
    <h3>6. functional Component - Object</h3>
    <input type="text" name="userName" placeholder='user name ...'
      value={userName}
      onChange={onChange}
    />
    <br/>
    <input type="text" name="message" placeholder='message ...'
      value={message}
      onChange={onChange}
```

```
        onKeyDown={onKeyDown}
      />
      <br/>
      <button onClick={onClick}>메시지확인</button>
    </div>
    );
};

export default React04Event07;
```