Ask Company

리액트와 함께 장고 시작하기 / 리액트

이벤트 처리하기

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

이벤트

컴포넌트에는 여러 이벤트가 발생 → 이벤트에 대한 처리를 커스텀

웹브라우저의 HTML이벤트를 기본적으로 지원

이벤트 핸들러 속성명은 camelCase로만 작성 (HTML에서는 onclick, 리액트는 onClick)

이벤트 핸들러에는 필히 함수를 지정 (HTML에서는 문자열로 코드를 지정)

DOM 요소에만 이벤트가 지원됩니다.

커스텀 리액트 컴포넌트에서는 HTML 이벤트를 지원하지 않습니다.

하지만 내부 Element에 DOM요소를 담아, 핸들러를 지정할 수 있습니다.

```
<div onClick={this.onClick} />

<Counter onClick={this.onClick} />
```

https://reactjs.org/docs/events.html

지원 이벤트

Clipboard Events : onCopy, onCut, onPaste

Composition Events : onCompositionEnd, onCompositionStart, onCompositionUpdate

Keyboard Events : onKeyDown, onKeyPress, onKeyUp

Focus Events : onFocus, onBlur

Form Events : onChange, onInput, onInvalid, onSubmit

Mouse Events : onClick, onContextMenu, onDoubleClick, onDrag, onDragEnd, onDragEnter, onDragExit, onDragLeave, onDragOver, onDragStart, onDrop, onMouseDown, onMouseEnter, onMouseLeave, onMouseMove, onMouseOut, onMouseOver, onMouseUp

Pointer Events : onPointerDown, onPointerMove, onPointerUp, onPointerCancel, onGotPointerCapture, onLosPointerCapture, onPointerEnter, onPointerLeave, onPointerOver, onPointerOut

Selection Events : onSelect

Touch Events : onTouchCancel, onTouchEnd, onTouchMove, onTouchStart

UI Events : onScroll

Wheel Events : onWheel

Media Events : onAbort, onCanPlay, onCanPlayThrough, onDurationChange, onEmptied, onEncrypted, onEnded, onLoadedData, onLoadedMetadata, onLoadStart, onPause, onPlay, onPlaying, onProgress, onRateChange, onSeeked, onSeeking, onStalled, onSuspend, onTimeUpdate, onVolumeChange, onWaiting

Image Events : onLoad, onError

Animation Evnets : onAnimationStart, onAnimationEnd, onAnimationIteration

Transition Events : onTransitionEnd

Other Events : onToggle

https://reactjs.org/docs/events.html#supported-events

이벤트 핸들러와 bind

```
class App extends React.Component {
  constructor(props) {
    super(props);
    this.onChangeInput2 = this.onChangeInput2.bind(this);
  }

  handleChangeInput1 = function(e) {
    const { name, value } = e.target;
    console.log(`[handleChangeInput1] name=${name}, value=${value} <= this=`, this);  // ➔ undefined
  }

  handleChangeInput2 = function(e) {
    const { name, value } = e.target;
    console.log(`[handleChangeInput2] name=${name}, value=${value} <= this=`, this);  // ➔ OK
  }

  handleChangeInput3 = (e) => {
    const { name, value } = e.target;
    console.log(`[handleChangeInput3] name=${name}, value=${value} <= this==`, this); // ➔ OK
  }

  render() {
    return (
      <input name={"myquery"} onChange={this.handleChangeInput2} />
    );
  }
}
```

상탯값 접근/변경 등을 위해 this 접근이 필요합니다.

아직 ECMA 표준 X. babel-plugin-transform-class-properties이 필요.

input 입력 문자열을 상탯값으로 저장하기

```
import React from 'react';


class App extends React.Component {
  state = { myquery: '' }

  onChangeInput3 = (e) => {
    const { name, value } = e.target;
    this.setState({
      [name]: value,
    });
  }

  render() {
    return (
      <div>
        myquery: {this.state.myquery} ({this.state.myquery.length}) <br />
        <input name={"myquery"} onChange={this.onChangeInput3} />
      </div>
    );
  }
}

export default App;
```

Life is short.
You need Python and Django.

I will be your pacemaker.