Ask Company

리액트와 함께 장고 시작하기 / 리액트

첫 리액트 컴포넌트 만들기 (클릭 카운터)

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

React.Component

HTML은 거의 직접 만들지않고, JavaScript만으로 UI를 구성할 수 있습니다.

- jsx는 HTML이 아닙니다.

리액트가 변경된 속성값/상탯값을 기반으로 UI를 자동 갱신

가상 돔 (Virtual DOM)을 통한, 빠른 UI 갱신

- DOM 계산 비용이 비쌉니다.
- 이전 UI상태를 메모리에 유지 + 변경될 UI의 최소 집합을 계산

React.Component 주요 구성요소

1. props : 읽기 전용 → 불변 객체

컴포넌트 생성 시에, 부모 Component로부터 전달받습니다.

부모가 props를 변경하면 (부모 입장에서는 state), 변경된 props값을 참조하는 UI가 자동으로 업데이트됩니다.

2. state : 상탯값

각 컴포넌트가 개별로 생성/유지하는 상태값들 → 주로 컴포넌트 단위로 UI에 반영할 값들을 저장할 목적

상태가 변경되면, 변경된 state값을 참조하는 UI가 자동으로 업데이트됩니다.

불변객체로 처리해야만 합니다. → immer 팩키지를 통해 보다 편리한 처리도 가능.

제공되는 상탯값 setter 함수를 통해 변경해야만 합니다. 직접 변경하면 UI 업데이트가 되지 않습니다. (강제 업데이트가 있긴 하지만 ...)

4. Element Tree

4. Component Tree

리액트 컴포넌트의 시작

src/index.js

최상위 컴포넌트를 특정 DOM에 렌더링

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(
    <App />,
    document.getElementById('root'));
```

확장자는 .js이지만, .jsx 문법이 사용
→ 소스코드편집기에 jsx 문법으로 처리하기를 요청할 수 있습니다.

src/App.js

```
class Counter extends React.Component {
  render() {
    return (
      <div>
        0
      </div>
    );
  }
}
```

아직 속성값을 사용하진 않았습니다.

```
class App extends React.Component {
  render() {
    return (
      <Counter initialValue={10} />
    );
  }
}
```

속성값(props)으로 전달

속성값을 받아서, 상탯값으로 저장하기

src/App.js

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      value: this.props.initialValue,
    };
  }

  render() {
    return (
      <div>
        {this.state.value}
      </div>
    );
  }
}
```

상탯값을 직접 할당하는 것은 본 생성자에서만 허용

요즘은 클래스 필드 문법으로 이렇게 씁니다.
아직 ECMA 표준 X : @babel/plugin-proposal-class-properteis가 필요

```
class Counter extends React.Component {
  state = {
    value: this.props.initialValue,
  };

  render() {
    return (
      <div>
        {this.state.value}
      </div>
    );
  }
}
```

https://github.com/tc39/proposal-class-fields

클릭 카운터 마저 구현하기

```
class Counter extends React.Component {
  state = {
    value: this.props.initialValue,
  };

  onClick = () => {
    this.setState({ value: this.state.value + 1 })
  }

  render() {
    return (
      <div onClick={this.onClick}>
        {this.state.value}
      </div>
    );
  }
}

export default App;
```

Arrow Function이기에 this가 바로 가능

모든 상탯값은 관련 setter를 통해 갱신

click 리스너를 등록

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

https://www.npmjs.com/package/babel-plugin-transform-class-properties

Event Listener로 사용되는 함수는 Arrow Function를 사용하지 않으면, 생성자에서 별도의 bind 과정이 필요.

클래스형 컴포넌트에서 Event Listener로 사용되는 함수는 Arrow Function으로 정의하시면 편리

색상 있는 원형 카운터 컴포넌트

```jsx
import React from "react";

class Counter extends React.Component {
  state = {
    color: this.props.color,
    value: 0,
  }

  onClick = () => {
    this.setState({ value: this.state.value + 1 });
  }

  onContextMenu = (e) => {
    e.preventDefault();
    this.setState({ value: this.state.value - 1 });
  };

  render() {
    return (
      <div onClick={this.onClick} onContextMenu={this.onContextMenu}
           style={{ ...style, backgroundColor: this.state.color }}>
           {this.state.value}
      </div>
    );
  }
}
```

```jsx
const style = {
  width: '100px',
  height: '100px',
  borderRadius: '50px',
  lineHeight: '100px',
  textAlign: 'center',
  display: 'inline-block',
  fontSize: '3rem',
  margin: '1rem',
  userSelect: 'none',
};


export default Counter;
```

Life is short.
You need Python and Django.

I will be your pacemaker.