Ask Company

리액트와 함께 장고 시작하기 / 리액트

React Element

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

리액트

UI 라이브러리 (웹 프론트엔드 및 앱 Native, VR 등에서 활용)

UI데이터를 관리하는 방법을 제공

- 부모 컴포넌트로부터 내려받는 속성값 → props
- 컴포넌트 내부에서 생성/관리되는 상탯값 → state

UI데이터(UI에 연결된 속성값/상탯값)가 변경되면, 해당 컴포넌트의 render() 함수가 호출이 되어 화면을 자동으로 갱신

- 클래스형 컴포넌트에서는 render() 함수가 호출
- 함수형 컴포넌트에서는 그 함수가 매번 호출. 컴포넌트에서 유지해야할 값들은 Hook을 통해 관리

리액트의 핵심 - 선언적 UI (Declarative UI)

UI에 변화를 가할 때마다 일일이 코드를 수행하는 것이 아니라,

데이터 (속성값/상탯값)에 맞춰 보여질 UI를 미리 선언해두면,

데이터 변경가 변경되면, 그 즉시 데이터에 맞춰 UI가 그려집니다.

리액트 개발의 핵심

UI에 보여질 값 (속성값/상탯값)들을 효율적으로 관리하고,
그 값들의 변경에 맞춰, UI가 불필요하게 업데이트되지 않도록 하기

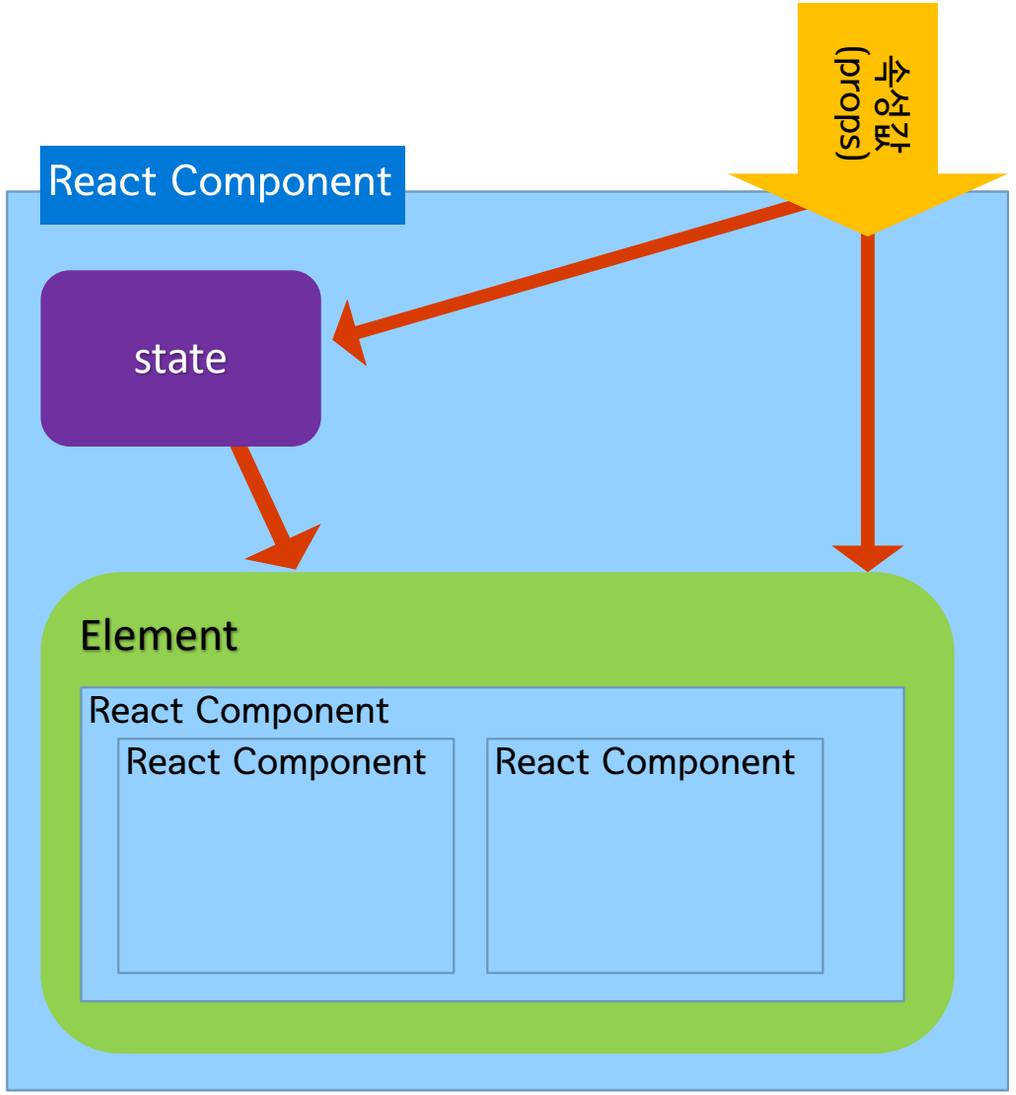
jQuery에서처럼 DOM에 직접 접근하여 추가/변경/삭제를 하는 방식을 지양합니다. 필요하다면 할 수는 있습니다. (Ref 사용)

비교) 일반적인 jQuery 코드

```
<script src="https://code.jquery.com/jquery-2.2.4.js"></script>

<div id="post_list"></div>

<button id="add_post">Add Post</button>

<script>
$(function() {
  const post_list = [];

  $("#add_post").click(function() {
    const post_title = "new post";
    const post = "<div class='post'>" + post_title + "</div>";
    $('#post_list').append(post);
  });
});
</script>
```

```
function App() {
  const [postList, setPostList] = useState([]);

  const addPost = () => {
    const post_title = "New Post";
    setPostList([
      ...postList,
      { title: post_title },
    ]);
  };

  return (
    <div>
      {postList.map(post =>
        <div className="post">{post.title}</div>
      )}
      <button onClick={addPost}>Add Post</button>
    </div>
  );
}
```

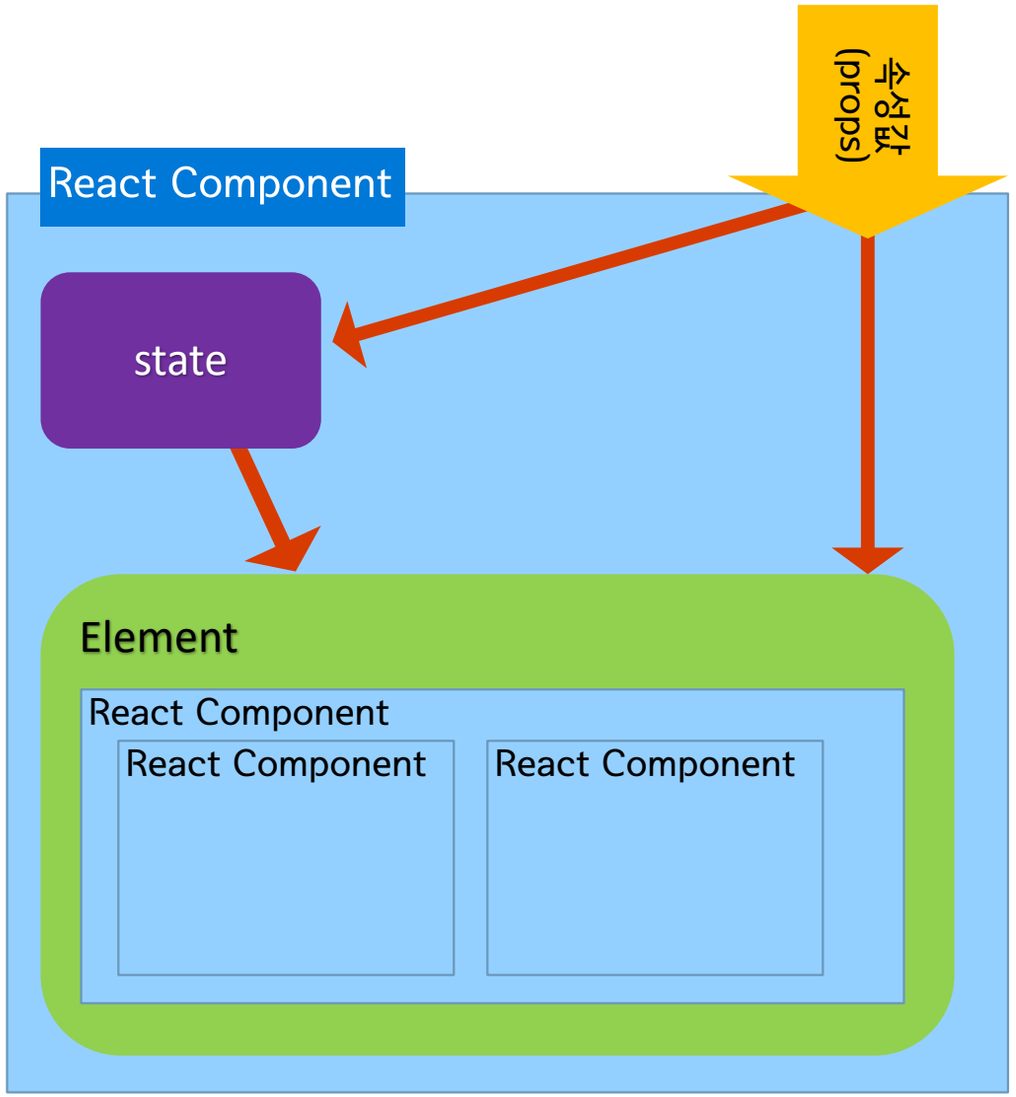
예제) 카운터 컴포넌트 (클래스)




```jsx
import React from "react";

class Counter1 extends React.Component {
  state = {
    value: this.props.initialValue,
  };

  render() {
    const { value } = this.state;
    return (
      <div>
        Counter1: {value}
        <button
          onClick={() => {
            this.setState({ value: value + 1})
          }}>
            value +1
        </button>
      </div>
    );
  }
}
```

```jsx
<Counter1 initialValue={10} />
```

예제) 카운터 컴포넌트 (함수)




```jsx
import React, { useState } from "react";

function Counter2(props) {
  const [state, setState] = useState({ value: props.initialValue });
  const { value } = state;
  return (
    <div>
      Counter2: {value}
      <button
        onClick={() => {
          setState({ ...state, value: value + 1 })
        }}>
          value +1
      </button>
    </div>
  );
}
```

```jsx
<Counter2 initialValue={10} />
```

React Element

화면을 담당하며, React 앱의 가장 작은 단위

```jsx
// jsx 문법
const reactElement1 = <h1>Hello, React!</h1>;

// js 문법
const reactElement2 = React.createElement('h1', null, 'Hello, React!');
```

일반 객체 (Plain Object)

React DOM은 React Element와 일치하도록 DOM을 업데이트

Element는 Component에서 화면을 담당

컴포넌트의 주요 구성요소 : 속성값 (props), 상탯값 (state), 엘리먼트 (element), 그리고 로직

https://ko.reactjs.org/docs/rendering-elements.html

React Component

Component를 통해 UI를 재사용 가능한 개별적인 여러 조각으로 나눕니다.

개념적으로 JavaScript 함수와 유사

속성값을 전달받아, Element를 반환

클래스로 구현하는 컴포넌트가 먼저 지원되었으며, 최근에 함수로 구현하는 컴포넌트를 지원

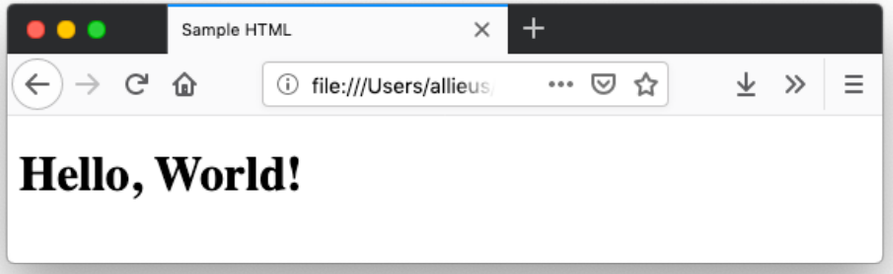
```
class Person1 extends React.Component {
  render() {
    return (
      <div>
        Person: {this.props.name}
      </div>
    );
  }
}
```

```
function Person2(props) {
  return (
    <div>
      Person: {props.name}
    </div>
  );
}
```

https://ko.reactjs.org/docs/components-and-props.html

React.createElement 샘플

포커스 : JSX를 쓰지 않아도 리액트를 사용할 수 있지만, JSX를 사용하지 않는다면, 이렇게 번거롭구나.



React.createElement

React.createElement(component, props, ...children) ➔ ReactElement

1. component
   - 문자열이나 리액트 컴포넌트. 문자열일 경우 DOM 요소를 생성
2. props
   - 컴포넌트가 사용하는 데이터
3. ...children
   - 해당 컴포넌트가 감싸고 있는 내부의 컴포넌트들을 다수 지정

HTML로만 작성된 페이지

```html
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Hello, HTML!</title>
</head>
<body>
    <div id="container">
        <h1>Hello, HTML!</h1>
    </div>
</body>
</html>
```

생성된 DOM 트리

```
<!doctype html>
<html>
▼<head>
    <meta charset="utf-8">
    <title>Hello, HTML!</title>
  </head>
▼<body>
  ▼<div id="container"> == $0
      <h1>Hello, HTML!</h1>
    </div>
  </body>
</html>
```

Vanilla JS로 작성된 페이지

```
<html>
<head>
  <meta charset="utf-8" />
  <title>Hello JavaScript</title>
</head>
<body>
</body>

<script>
  var container = document.createElement('div');
  container.setAttribute('id', 'container');

  var h1 = document.createElement('h1')
  var textNode = document.createTextNode('Hello, JavaScript!');
  h1.appendChild(textNode);

  container.appendChild(h1);

  document.body.appendChild(container);
</script>
</html>
```

생성된 DOM 트리

```
<html>
▼<head>
    <meta charset="utf-8">
    <title>Hello JavaScript</title>
  </head>
▼<body>
  ▼<script>
          var container = document.createElement('div');
          container.setAttribute('id', 'container');

          var h1 = document.createElement('h1')
          var textNode = document.createTextNode('Hello, JavaScript!');
          h1.appendChild(textNode);

          container.appendChild(h1);

          document.body.appendChild(container);
    </script>
  ▼<div id="container"> == $0
      <h1>Hello, JavaScript!</h1>
    </div>
  </body>
</html>
```

Vanilla JS에서 React 써보기

```html
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Hello, React!</title>
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
</head>
<body>
    <div id="container"></div>
    <script>
        const reactElement = React.createElement('h1', null, 'Hello, React!');
        const container = document.getElementById('container');

        ReactDOM.render(reactElement, container);
    </script>
</body>
</html>
```

JSX 문법으로 가독성을 높일 수 있습니다.

생성된 DOM 트리

```
<!doctype html>
<html>
▶ <head>…</head>
▼ <body>
  ▼ <div id="container"> == $0
      <h1>Hello, React!</h1>
    </div>
  ▼ <script>
            const reactElement = React.createElement('h1', null, 'Hello, React!');
            const container = document.getElementById('container');

            ReactDOM.render(reactElement, container);

    </script>
  </body>
</html>
```

JSX로 React 써보기 (1)

React.createElement가 너무 번거롭다.

JSX

JavaScript의 문법 확장 + HTML과 비슷한 문법 → JavaScript 코드로 변환

어떤 브라우저도 지원하지 않음 → babel을 통한 transpile이 필요

class가 예약어이므로, HTML Tag 속성의 class대신에 className을 사용

{} 안에는 자바스크립트 식을 지정 (함수 등)

jsx 코드

```
const ele1 = <h1>Hello, World!</h1>;
```

변환된 JavaScript 코드

```
React.createElement('h1', null, 'Hello, World!');
```

https://reactjs-kr.firebaseapp.com/docs/introducing-jsx.html

JSX로 React 써보기 (2)

webpack 세팅을 통해, 다양한 기능을 얻을 수 있습니다.

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Sample HTML</title>
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
    <script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
</head>
<body>
    <div id="container"></div>
    <script type="text/babel">
        const reactElement = <h1>Hello, World!</h1>;
        const container = document.getElementById('container');
        ReactDOM.render(reactElement, container);
    </script>
</body>
</html>
```

실제 Production에서는 서비스 배포시에 transpile 전처리를 수행합니다.

React.Component 샘플

HTML을 직접 쓰기

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Hello, HTML!</title>
</head>
<body>
    <div id="container">
        <div>
            <h1>좋아하는 과일</h1>
            <ul>
                <li>바나나</li>
                <li>사과</li>
                <li>딸기</li>
            </ul>
        </div>
    </div>
</body>
</html>
```

React로 동일한 DOM 구조 만들기

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Sample HTML</title>
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
</head>
<body>
    <div id="container"></div>
    <script>
        var container = document.getElementById('container');
        var reactElement = React.createElement(
            'div',
            null,
            React.createElement('h1', null, '좋아하는 과일'),
            React.createElement(
                'ul',
                null,
                ['바나나', '사과', '딸기'].map((name, idx) => (
                    React.createElement('li', {key: idx}, name)
                ))
            )
        );
        ReactDOM.render(reactElement, container);
</script>
</body>
</html>
```

React - 함수형 컴포넌트

```
const HomeComponent = (props) =>
  React.createElement(
    'div',
    null,
    React.createElement('h1', null, '좋아하는 과일'),
    React.createElement(
      'ul',
      null,
      ['바나나', '사과', '딸기'].map((name, idx) => (
        React.createElement('li', {key: idx}, name)
    ))
  )
);

let reactElement = React.createElement(HomeComponent);
let container = document.getElementById('container');

ReactDOM.render(reactElement, container);
```

React - 클래스형 컴포넌트

```
class HomeComponent extends React.Component {
  render() {
    return React.createElement(
      'div',
      null,
      React.createElement('h1', null, '좋아하는 과일'),
      React.createElement(
        'ul',
        null,
        ['바나나', '사과', '딸기'].map((name, idx) => (
          React.createElement('li', {key: idx}, name)
        ))
      )
    );
  }
}

let reactElement = React.createElement(HomeComponent);
let container = document.getElementById('container');

ReactDOM.render(reactElement, container);
```

React - JSX 활용

```
    <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
    <script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
</head>
<body>
    <div id="container"></div>
    <script type="text/babel">
        class HomeComponent extends React.Component {
            render() {
                return (
                    <div>
                        <h1>좋아하는 과일</h1>
                        <ul>
                            {['바나나', '사과', '딸기'].map((name, idx) => (
                                <li key={idx}>{name}</li>
                            ))}
                        </ul>
                    </div>
                );
            }
        }

        const reactElement = <HomeComponent />;
        const container = document.getElementById('container');

        ReactDOM.render(reactElement, container);
    </script>
```

실제로는 명령행에서 babel 명령을 통해,
코드 수정시마다 매번 자동 transpile 수행토록 설정합니다.

React 컴포넌트에 props 지정하기

클래스형 컴포넌트

```
class HomeComponent extends React.Component {
  render() {
    return (
      <div>
        <h1>좋아하는 과일</h1>
        <ul>
          {this.props.fruits.map((name, idx) => (
            <li key={idx}>{name}</li>
          ))}
        </ul>
      </div>
    );
  }
}
```

함수형 컴포넌트

```
const HomeComponent = (props) =>
  <div>
    <h1>좋아하는 과일</h1>
    <ul>
      {props.fruits.map((name, idx) => (
        <li key={idx}>{name}</li>
      ))}
    </ul>
  </div>;
```

```
const fruits = ['바나나', '사과', '딸기'];
const reactElement = <HomeComponent fruits={fruits} />;
const container = document.getElementById('container');

ReactDOM.render(reactElement, container);
```

Life is short.
You need Python and Django.

I will be your pacemaker.