UI 표현하기

> Component? UI의 구성 요소

React Component? JavaScript로 작성된 함수

> 16에서 함수형이 도입되고, 16.8버전에서 Hook이 도입된 후 현재의 React는 함수형으로 컴포넌트를 작성하는 것을 원칙으로 하고 있다

React Component의 정의

```
// Pascal Case의 형태로 선언한다
function Component() {
  return (
    <Flex>
      <div>Test</div>
    </Flex>
  );
}
```

> Pascal Case가 아니라면?

JSX의 변환

JSX의 파싱이 끝나고 나면 `React.createElement`로 변환된다

```
function Component() {
  return React.createElement(
    Flex,
    null,
    React.createElement('div', null, 'Test')
  );
}
```

이때 소문자로 시작하면(`<component />`) `React.createElement('component', null)`로 전환된다 (예전)

import/export

```
// 그냥 자연스럽게 쓰고 있는 import 형태이며, default export된 형태
import ComponentA from '@/components/ComponentA';

// ESM 스타일 import
import ComponentB from '@/components/ComponentB.tsx';

// ESM 스타일, named export 형태
import { ComponentC } from '@/components/ComponentC.tsx';
```

JSX?

> JSX와 React는 서로 다른 별개의 개념입니다.

위에서 `React.createElement~`의 형태로 전환됨을 보았습니다
하지만 React와 JSX의 발전으로 Babel, TypeScript 등에 의해 React의존없이 변환됩니다

> 따라서, `import React from "react";` 또한 이제는 필요가 없어졌죠

변화된 JSX 컴파일

```
// 컴파일러에 의해 자동으로 추가됩니다!
import { jsx as _jsx } from 'react/jsx-runtime';

function App() {
  return _jsx('h1', { children: 'Hello world' });
}
```

하나의 태그로 감싸야 한다

return되는 부분은 여러 태그가 부모가 되어선 안됩니다.
하나의 객체로 완성되어야 하기에 형제로 두기를 원한다면
`Fragment` 또는 `<> </>`로 감싸줘야 하죠

HTML attributes?

```
<main>
    <div class="class1">
        Div with class name 'class1'
    </div>
    <label for="input">Text Input</label>
    <input id="input" type="text"></input>
</main>
```

일반적인 html은 이렇게 작성되죠

JSX로 작성할때는 for, class와 같이 JS의 예약어인 경우 `htmlFor`, `className` 같이 바꿔서 사용하며,
`-(dash)`로 연결되어야 하는 부분들은 JS의 변수명으로 지정될수 없어 `camelCase`로 작성해야 합니다.

```
<main>
  <div className="class1">Div with class name 'class1'</div>
  <label htmlFor="input">Text Input</label>
  <input id="input" type="text"></input>
</main>
```

> aria-속성과 data- dataset은 예외로 dash를 사용하여 작성합니다.

JSX 안에서 JS의 사용

다들 아시다시피 `{ }` 중괄호로 감싸 사용합니다

그럼 아래와 같은 변환 결과를 나타내게 되겠죠

```
// 조건부 렌더링
_jsx('div', {
  children: isTrue
    ? _jsx('span', { children: 'True' })
    : _jsx('span', { children: 'False' }),
});
// Array 렌더링
_jsx('ul', {
  children: items.map((item) =>
    _jsx('li', { key: item.id, children: item.name })
  ),
});
```

이중 중괄호?

style attribute를 설정할때 이중으로 중괄호를 사용하게 됩니다.

```
<div style={{ width: '100%' }} />
```

이는 JS의 객체를 style 속성에 전달하는 것으로,

```
const styleObj = { width: '100%' };
<div style={styleObj} />;
```

와 동일한 형태입니다.

> 인라인 css 작성시에도 key를 camelCase로 작성해야 합니다

Array rendring

```
<ul>
  {Array.from({ length: 10 }).map((item) => (
    <li key={item}>{item}</li>
  ))}
</ul>
```

Array rendering에서 중요한 것은 `key` 속성이죠

> key를 부여하지 않으면 eslint 등에 의한 에러를 보곤 합니다.
> 설정해주지 않는 경우 React는 array index를 key로 자동으로 넣습니다.

key의 설정

- Array의 index를 사용하기
- Math.random을 사용하기

위는 React에서 하지 말라고 하는 방식들입니다

`Reconciliation (재조정)`과정에서 효율적인 업데이트를 위해 사용하는 중요 속성이기 때문이죠

각 key는 Fiber Node의 고유 식별자로 작용하여 효율적인 렌더링 최적화에 중요 역할을 합니다.

> Reconciliation Motivation
> 트리 -> 트리 연산 수 **최신** 알고리즘 조차 시간복잡도가 무려 O(n^3)

순수함수로의 컴포넌트?

일반적인 React Project 생성 시 아래와 같이 StrintMode가 우리의 App을 감싸고 있습니다.
그리고 이로인한 두번의 렌더링으로 우리에게 에러를 보여주기도 하죠

```
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Strict Mode

> React Strict Mode
> React는 개발 중에 각 컴포넌트의 함수를 두 번 호출하는 "엄격 모드"를 제공합니다. 컴포넌트 함수를 두 번 호출함으로써, 엄격 모드는 이러한 규칙을 위반하는 컴포넌트를 찾는데 도움을 줍니다.

props와 state, context에 따라 두번 호출되어도, 동일한 결과를 내는지를 보여주는 게 목적이었습니다!