App 함수와 JSX 문법

App 함수를 여러가지 방법으로 구현해 보자.

순수 자바스크립트 문법으로 App 함수 구현

src/App.css 수정

```
table { border-collapse: collapse; margin: 5px; }
td { border: 1px solid gray; padding: 4px; }
```

src/App.jsx 수정

```
import React from 'react';
import './App.css';

function App() {
  let persons = [{ name: '홍길동', age: 16 },
                 { name: '임꺽정', age: 19 },
                 { name: '전우치', age: 20 }];

    let td1 = React.createElement("td", null, "이름");
  let td2 = React.createElement("td", null, "나이");
  let tr = React.createElement("tr", null, td1, td2);
  let trlist = [tr];
  for (let person of persons) {
    td1 = React.createElement("td", null, person.name);
    td2 = React.createElement("td", null, person.age);
    tr = React.createElement("tr", null, td1, td2);
    trlist.push(tr);
  }
  let table = React.createElement("table", null, trlist);
  return table;
}
```

```
export default App;
```

App 함수를 순수 자바스크립트 문법으로 구현했다.
위 App 함수는 React.createElement(...) 메소드를 호출하여 Virtual DOM 태그 객체들을 생성해서 리턴한다.
다음 화면과 같은 Virtual DOM 태그 객체들이 생성되어 리턴된다.

이름	나이
홍길동	16
임꺽정	19
전우치	20

React.createElement(name, attribute, children) 메소드

Virtual DOM 객체를 생성하는 메소드

name: 태그 이름

attribute: 태그 애트리뷰트

children: 태그 자식 (예: <td>태그 자식</td>, <tr>태그 자식</tr>)

let td1 = React.createElement("td", null, "이름");

<td>이름</td> 태그 Virtual DOM 객체를 생성하여 td1 변수에 대입한다.

let td2 = React.createElement("td", null, "나이");

<td>나이</td> 태그 Virtual DOM 객체를 생성하여 td2 변수에 대입한다.

let tr = React.createElement("tr", null, td1, td2);

위에서 생성한 td1, td2 Virtual DOM 객체를 자식으로 갖는 tr 태그 Virtual DOM 객체, 즉 <
tr><td>이름</td><td>나이</td></tr> 태그 Virtual DOM 객체를 생성하여
tr 변수에 대입한다.

let trlist = [tr];

바로 윗줄에서 생성한 tr Vritual DOM 객체가 하나 들어있는 배열을 생성하여
trlist 변수에 대입한다.

for (let person of persons) {

persons 배열의 원소 각각을 person 변수에 대입하여 아래 코드를 반복 실행한다.

td1 = React.createElement("td", null, person.name);

<td>person.name값</td> 태그 virtual DOM 객체를 생성하여 td1 변수에 대입한다.

td2 = React.createElement("td", null, person.age);

<td>person.age값</td> 태그 virtual DOM 객체를 생성하여 td2 변수에 대입한다.

tr = React.createElement("tr", null, td1, td2);

위에서 생성한 td1, td2 Virtual DOM 객체를 자식으로 갖는 tr 태그 Virtual DOM 객체를 생성하여 tr 변수에 대입한다.

trlist.push(tr);

바로 윗줄에서 생성한 tr Vritual DOM 객체를 trlist 배열의 끝에 추가한다.

}

let table = React.createElement("table", null, trlist);
return table;

trlist 배열의 tr 태그 Virtual DOM 객체들을 자식으로 갖는
table 태그 virtual DOM 객체를 생성하여 리턴한다.

이렇게 App 함수가 리턴한 Virtual DOM 태그 객체들이
index.html 웹페이지의 태그 사이에 렌더링 된다.

이름	나이
홍길동	16
임꺽정	19
전우치	20

위 문장을 좀 더 쉽게 설명하면 다음과 같다.

> 이렇게 App 함수가 리턴한 Virtual DOM 태그 객체들과 동일한 구조의 DOM 객체들이 생성되어
> index.html 웹페이지의 태그 사이에 삽입된다.

JSX 문법으로 구현

Virtual DOM 개체 생성 코드를 쉽게 구현할 수 있도록 리액트는 자바스크립트 문법을 확장하였다.
그 문법이 JSX 이다.

표준 자바스크립트/타입스크립트 문법으로 구현된 소스코드 파일은 *.js*/.ts 이고,
확장된 JSX 문법으로 구현된 자바스크립트/타입스크립트 소스코드 파일은
.jsx/.tsx 이다.

예를 들어

```
let td1 = React.createElement("td", null, "이름"); //(가)
```

> <td>이름</td> 태그 Virtual DOM 객체를 생성하여 td1 변수에 대입한다.

위 코드를 JSX 문법으로 다음과 같이 구현할 수 있다.

```
let td1 = <td>이름</td>;  //(나)
```

JSX 문법으로 구현된 코드는, 프로젝트가 빌드될 때 순수 자바스크립트 코드로 변환된다.
예를 들어 (나) 소스 코드는 (가) 소스 코드로 변환된다.

---

src/App.jsx 수정

노란색으로 칠한 부분이 확장된 JSX 문법으로 구현된 Virtual DOM 객체 생성코드이다.

```
import React from 'react';
import './App.css';

function App() {
  let persons = [{ name: '홍길동', age: 16 },
                 { name: '임꺽정', age: 19 },
                 { name: '전우치', age: 20 }];

  let td1 = <td>이름</td>;
  let td2 = <td>나이</td>;
  let tr = React.createElement("tr", null, td1, td2);
  let trlist = [tr];
  for (let person of persons) {
    td1 = <td>{ person.name }</td>
    td2 = <td>{ person.age }</td>
    tr = React.createElement("tr", null, td1, td2);
    trlist.push(tr);
  }
  let table = React.createElement("table", null, trlist);
  return table;
}

export default App;
```

<td>이름</td>

'이름' 부분은 그냥 그대로 출력할 텍스트이므로, 위와 같이 구현한다.

<td>{ person.name }</td>

person.name 부분은 자바스크립트 표현식이라서, 위와 같이 { } 괄호가 필요하다.
{ } 괄호가 없으면 'person.name' 텍스트가 그냥 그대로 출력된다.

예를 들어 person.name 값이 '홍길동' 이라면

<td>{person.name}</td> 이 코드는 <td>홍길동</td> 태그 객체를 생성하지만
<td>person.name</td> 이 코드는 <td>person.name</td> 태그 객체를 생성한다.

참고 - 표현식(expression)

어떤 값이 있는 소스 코드 부분은 표현식이다.
숫자 값, 문자열 값, true false 값, 객체, 배열 등

if 문, for 문 그 자체에는 값이 없으므로 표현식이 아니다.

함수를 호출하는 코드는 그 함수의 리턴값이 값이기 때문에 표현식이다.

src/App.jsx 수정

tr 태그 생성 부분과 table 태그 생성 부분에도 JSX 문법을 적용하면 다음과 같다.

```
import React from 'react';
import './App.css';

function App() {
  let persons = [{ name: '홍길동', age: 16 },
                 { name: '임꺽정', age: 19 },
                 { name: '전우치', age: 20 }];

  let tr = <tr><td>이름</td><td>나이</td></tr>;
  let trlist = [tr];
  for (let person of persons) {
    tr = <tr><td>{ person.name }</td><td>{ person.age }</td><
    trlist.push(tr);
  }
  let table = <table>{ trlist }</table>;
  return table;
}

export default App;
```

<table>{ trlist }</table>

trlist는 자바스크립트 변수이므로 { } 괄호로 묶어야 한다.

src/App.jsx 수정

좀 더 간결하게 구현할 수 있다

```
import React from 'react';
import './App.css';

function App() {
  let persons = [{ name: '홍길동', age: 16 },
                 { name: '임꺽정', age: 19 },
                 { name: '전우치', age: 20 }];

  let trlist = [];
  for (let person of persons) {
    let tr = <tr><td>{ person.name }</td><td>{ person.age }
</td></tr>;
    trlist.push(tr);
  }
  let table = <table>
                <tr><td>이름</td><td>나이</td></tr>
                { trlist }
              </table>;
  return table;
}

export default App;
```

---

src/App.jsx 수정

리액트 개발자들 대부분은 아래와 같은 형태로 구현한다.

```
import React from 'react';
import './App.css';

function App() {
  let persons = [{ name: '홍길동', age: 16 },
                 { name: '임꺽정', age: 19 },
```

```
                  { name: '전우치', age: 20 }];

  let trlist = persons.map(person =>
                    <tr><td>{ person.name }</td><td>{ person.
  return <table>
            <tr><td>이름</td><td>나이</td></tr>
            { trlist }
         </table>;
}

export default App;
```

리액트 개발자는 자바스크립트 배열의 map, filter, reduce 메소드를 능숙하게 활용할 수 있어야 한다.

JSX 문법 오류 #1

src/App.js 수정

```
import React from 'react';
import './App.css';

function App() {
  let persons = [{ name: '홍길동', age: 16 },
                 { name: '임꺽정', age: 19 },
                 { name: '전우치', age: 20 }];

  let trlist = persons.map(person =>
                    <tr><td>{ person.name }</td><td>{ person.
  return
    <table>
      <tr><td>이름</td><td>나이</td></tr>
      { trlist }
    </table>;
}
```

```
export default App;
```

위와 같이 구현하면, JSX 문법 오류이다.
직전 예제와 차이는 단지 return 문 다음에 줄 바꿈만 했을 뿐인데 오류라니...

위와 같은 JSX 문법 오류를 피하려면,
JSX 문법으로 구현된 태그 부분을 ( ) 괄호로 묶어주면 된다.

src/App.jsx 수정

```
import React from 'react';
import './App.css';

function App() {
  let persons = [{ name: '홍길동', age: 16 },
                 { name: '임꺽정', age: 19 },
                 { name: '전우치', age: 20 }];

  let trlist = persons.map(person =>
                    <tr><td>{ person.name }</td><td>{ person.
  return (
    <table>
      <tr><td>이름</td><td>나이</td></tr>
      { trlist }
    </table>
  );
}

export default App;
```

- 여러 줄로 구현된 JSX 태그 부분은 ( ) 괄호로 묶어주자.
- JSX 태그 부분에서 자바스크립트 표현식은 { } 괄호로 묶어주자.

JSX 문법 오류 #2

리액트 컴포넌트가 리턴하는 Virtual DOM 객체는 루트가 있는 트리 구조이어야 한다.

예시_1 : 루트가 h1 태그이므로 OK

```
import React from 'react';
import './App.css';

function App() {
  return (
    <h1>hello world</h1>
  );
}

export default App;
```

예시_2 : 루트가 없으므로 에러 (루트가 여러 개 이므로 에러)

```
import React from 'react';
import './App.css';

function App() {
  return (
    <h1>hello</h1>
    <h1>world</h1>
  );
}

export default App;
```

예시_3 : 루트가 div 태그이므로 OK

```
import React from 'react';
import './App.css';

function App() {
  return (
```

```
      <div>
        <h1>hello</h1>
        <h1>world</h1>
      </div>
  );
}

export default App;
```

CSS Style

src/App.css 수정

```
div { margin: 10px; font-size: 20pt; }
p.blue { color: blue; }
p.green { color: green; }
```

src/App.jsx 수정

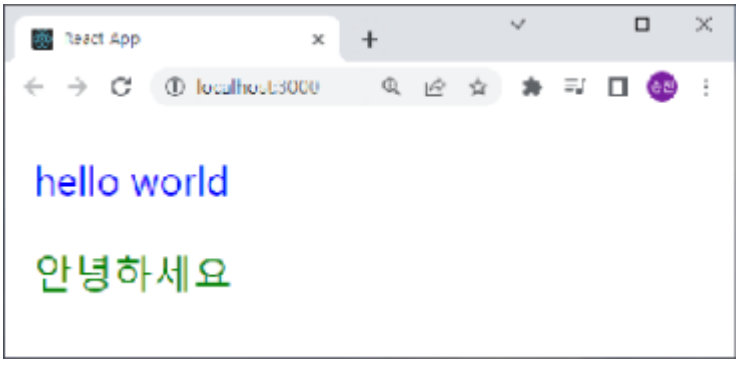
```
import React from 'react';
import './App.css';

function App() {
  return (
    <div>
      <p className="blue">hello world</p>
      <p className="green">안녕하세요</p>
    </div>
  );
}

export default App;
```

표준 HTML 태그라면 아래와 같이 구현해야 하지만

<p

class="blue">hello world</p>
    <p
class="green">안녕하세요</p>

JSX에서 class 단어는 자바스크립트 키워드라서 사용할 수 없다.
JSX에서는
class 대신 className을 사용해야 한다.



인라인 서식 (inline style)

표준 HTML 태그에서 인라인 서식을 아래와 같이 구현한다.

    <p
style="color: blue; font-size: 20pt;">hello world</p>
    <p
style="color: green; font-size: 25pt;">안녕하세요</p>

HTML에서 p는 태그이고 style은 애트리뷰트이다.

JSX 태그에서 인라인 서식은 아래와 같이 구현해야 한다.

    <p style=
{{ color: "blue", fontSize: "20pt" }}>hello world</p>
    <p style=
{{ color: "green", fontSize: "25pt" }}>안녕하세요</p>

JSX에서 p는 자바스크립트 객체이고 style은 속성(property)이다.
속성(property)을 짧게 프롭(prop)이라고 부른다. (properties == props)

style 프롭스의 값은 문자열이 아니고, 자바스크립트 객체이어야 한다.

<p style={ 자바스크립트_객체 }>

JSX 태그 부분에 자바스크립트 표현식은 { } 괄호로 묶어야 한다.

{ color: "blue", fontSize: "20pt" }

이것은 인라인 서식을 표현한 자바스크립트 객체이다.
자바스크립트 변수 이름이나 속성 이름에 - 문자를 사용할 수 없다.
따라서 인라인 서식을 자바스크립트 객체로 구현할 때,
font-size 형태의 서식명을 fontSize 형태로 구현해야 한다.

HTML에서 CSS 서식 이름은 케밥 케이스 방식이지만
JSX에서 CSS 서식 이름은
카멜 케이스 방식을 사용해야 한다.

src/App.jsx 수정

```
import React from 'react';
import './App.css';

function App() {
  return (
    <div>
      <p style={{ color: "blue", fontSize: "20pt" }}>hello wo
      <p style={{ color: "green", fontSize: "25pt" }}>안녕하세요
    </div>
  );
}

export default App;
```

src/App.jsx 수정

```
import React from 'react';
import './App.css';
```

```
function App() {
  const s1 = { color: "blue", fontSize: "20pt" };
  const s2 = { color: "green", fontSize: "25pt" };
  return (
    <div>
      <p style={ s1 }>hello world</p>
      <p style={ s2 }>안녕하세요</p>
    </div>
  );
}

export default App;
```

위와 같이 구현할 수도 있다.

s1, s2 변수의 값이 자바스크립트 객체이다.
인라인 서식의 값이 자바스크립트 객체이어야 한다.

함수형 컴포넌트

App 컴포넌트

src/App.jsx

```
import React from 'react';
import './App.css';

function App() {
  let s1 = { color: "blue", fontSize: "20pt" };
  let s2 = { color: "green", fontSize: "25pt" };
  return (
    <div>
      <p style={ s1 }>hello world</p>
      <p style={ s2 }>안녕하세요</p>
    </div>
  );
```

```
}

export default App;
```

src/App.tsx 파일에 구현된 App 컴포넌트를 호출하는 곳은 src/index.tsx 이다.

src/index.jsx

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pa
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.
reportWebVitals();
```

App 함수형 컴포넌트를 호출하는 부분이 <App /> 이다.
App 함수가 생성해서 리턴하는 Virtual DOM 객체들이 여기에서 렌더링 된다.

MyTable 컴포넌트 구현

src/MyTable.css 생성

```
table.MyTable { border-collapse: collapse; margin: 5px; }
```

```
table.MyTable td { border: 1px solid gray; padding: 4px; }
```

src/MyTable.jsx 생성

```
import React from 'react';
import './MyTable.css';

function MyTable() {
    let persons = [{ name: '홍길동', age: 16 },
                   { name: '임꺽정', age: 19 },
                   { name: '전우치', age: 20 }];

    let trlist = persons.map(person =>
        <tr><td>{ person.name }</td><td>{ person.age }</td></
    return (
        <table className="MyTable">
            <tr><td>이름</td><td>나이</td></tr>
            { trlist }
        </table>
    );
}

export default MyTable;
```

MyTable 함수 컴포넌트는 자바스크립트 함수이다.
이 함수가 호출될 때 마다, 다음과 같은 구조의 Virtual DOM 객체들을 생성해서 리턴한다.

```
<table>
    <td><td>이름</td><td>나이</td></tr>
    <td><td>홍길동</td><td>16</td></tr>
    <td><td>임꺽정</td><td>19</td></tr>
    <td><td>전우치</td><td>20</td></tr>
</table>
```

src/App.jsx 수정

```
import React from 'react';
import MyTable from './MyTable';

function App() {
  return (
    <div>
      <MyTable />
      <hr />
      <MyTable />
    </div>
  );
}

export default App;
```

<MyTable />

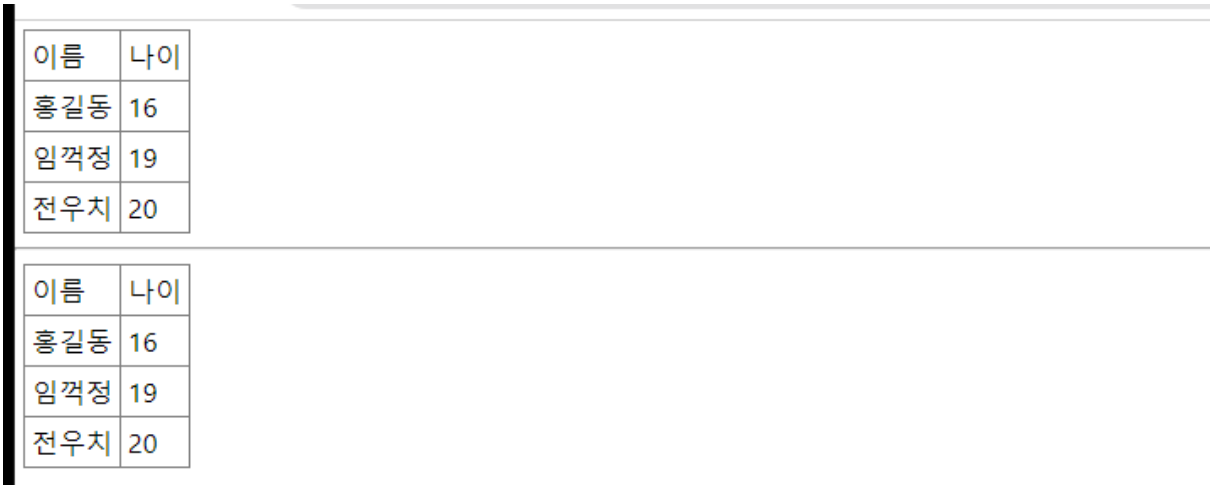
위 코드는 MyTable 컴포넌트를 호출한다.
MyTable 함수가 생성하여 리턴하는 Virtual DOM 객체들이 <MyTable /> 부분에 삽입된다.

App 함수는 MyTable 함수를 두 번 호출했다.
따라서 App 함수가 리턴하는 Virtual DOM 객체들은 다음과 같은 구조이다.

```
<div>
    <table>
        <td><td>이름</td><td>나이</td></tr>
        <td><td>홍길동</td><td>16</td></tr>
        <td><td>임꺽정</td><td>19</td></tr>
        <td><td>전우치</td><td>20</td></tr>
    </table>
    <hr />
    <table>
        <td><td>이름</td><td>나이</td></tr>
        <td><td>홍길동</td><td>16</td></tr>
        <td><td>임꺽정</td><td>19</td></tr>
        <td><td>전우치</td><td>20</td></tr>
```

```
    </table>
</div>
```

이 Virtual DOM 객체들은 다음과 같이 렌더링 된다.



이름	나이
홍길동	16
임꺽정	19
전우치	20

이름	나이
홍길동	16
임꺽정	19
전우치	20

요약

- 함수형 리액트 컴포넌트는 Virtual DOM 객체를 생성해서 리턴하는 자바스크립트 함수이다.
- JSX 문법으로 구현한 태그는 Virtual DOM 자바스크립트 객체이다. HTML 태그처럼 보이지만 HTML 태그가 아니다. 따라서 JSX 작성 규칙은 HTML 문서 작성 규칙과 다르다.
- 여러 줄로 구현된 JSX 태그 부분은 ( ) 괄호로 묶어주자.
- JSX 태그 부분에서 자바스크립트 표현식은 { } 괄호로 묶어주자.
- JSX 태그 부분에서는 class 대신 className을 사용해야 한다.
- *.css 파일은 표준 CSS 문법으로 구현한다.

JSX 태그 부분에서 style 값은 문자열이 아니고, 자바스크립트 객체이어야 한다.
JSX에서 CSS 서식 이름은 카멜 케이스 방식을 사용해야 한다.