8253jang@daum.net
장희정

React Component

"Elements are the smallest building blocks of React apps."

React컴포넌트가 해주는 역할은 어떠한 속성들(props)을 입력으로 받아서 **그에 맞는 React엘리먼트를 생성하여 리턴** 해준다.

1) 컴포넌트는 리액트의 핵심 개념 중 하나이다

2) html에서 화면을 구성하는 요소(버튼, 링크, 이미지 등)로 볼 수 있다.

3) 프로젝트 규모가 커져도 이미 작성한 컴포넌트를 재사용 한다면 개발 속도를 빠르게 할 수 있으며, 유지보수에도 유리하다.

4) 컴포넌트는 javascript 파일 내의 하나의 함수로 작성하거나 javascript 파일 자체로도 작성할 수 있다

- **클래스 컴포넌트**
  - 클래스 컴포넌트는 ES6 클래스를 활용하여 정의
  - React.Component를 상속받아 작성됨
  - 컴포넌트의 상태(state)를 관리할 수 있음
  - 생명주기 메서드를 사용할 수 있음

- **함수형 컴포넌트**
  - 함수로 정의
  - 상태와 생명주기 메서드를 사용할 수 없었지만, React Hooks 도입으로 가능해짐
  - 간결하고 가독성이 좋음.
  - 클래스 컴포넌트보다 성능이 약간 우수

구분	클래스 컴포넌트	함수형 컴포넌트
정의 방식(문법)	ES6 클래스로 정의	함수로 정의
상태 관리	가능 (this.state)	React Hooks 사용 가능 (useState)
생명주기 메서드	사용 가능	React Hooks 사용 가능 (useEffect)
가독성	일반적이고 명확한 구조	간결하고 가독성이 좋음
성능	일반적인 성능	성능이 약간 우수
사용 시기	초기의 React	상태가 필요하지 않은 경우, 간단한 UI 표현
장점	복잡한 로직 처리에 적합	간결하고 가독성이 좋음
발전 과정	초기 React 컴포넌트 방식	React 16.8에서 Hooks 도입으로 활용 범위 확장

컴포넌트의 이름은 **반드시 대문자로** 시작

컴포넌트를 어느 정도 수준까지 추출하는 것이 좋은지에 대한 정해진 기준은 없지만

기능단위로 구분하는 것이 좋고 나중에 곧바로 재사용이 가능한 형태로 추출하는 것이 좋다.

함수 형식으로 컴포넌트를 작성하며, 함수이름은 파일명과 동일하게 하며, 함수가 두 개 이상인 경에는 다른 이름으로 작성

JSX(JavaScript XML) 파일

JSX는 리액트에서 사용하는 파일로 JavaScript 코드 안에 HTML과 유사한 코드를 작성할 수 있게 해준다.

리액트 컴포넌트를 정의하고 UI를 구성하는데 매우 유용하다.

브라우저에서 직접 실행되지 않으며, Babel과 같은 변환기를 통해 Javascript로 변환 후 실행된다.

리액트 자체 빌드 도구 덕분에 *.js, *.jsx로 작성해도 아무런 문제없이 실행은 가능하나 jsx파일로 컴포넌트 만들 것을 권장한다.

8253jang@daum.net
장희정

컴포넌트 생성 – Componet1.jsx

```
// 함수 정의
function Component1() {
    // 출력할 내용을 return() 에 작성
    return(
        // 출력할 내용을 감싸기 위한 태그
        <>
            <h2>Hello Component1</h2>
        </>
    )
}
// 내보내기를 위한 문법
export default Component1;
```

컴포넌트 적용

- App.jsx 로 돌아와서 Component1.jsx import

```
import Component1 from './Component1'
```

- return() 내부의 <></> 태그 내에 Component1 작성

컴포넌트에 함수가 2개 이상인 경우

Component2.jsx 를 만들고 Fun1, Fun2 함수 작성

함수를 각각 export 하기 위해 함수 이름 앞에 export 작성

```jsx
export function Func1() {
  return (
    <>
      <h2>hello Fun1</h2>
    </>
  );
}

export function Func2() {
  return (
    <>
      <h2>hello Fun2</h2>
    </>
  );
}
```

App.jsx에서 사용하기 위해 import 문장 작성한다.
사용할 함수 이름을 { } 안에 콤마로 구분하여 작성 한다.
처음 작성시 함수 아래 빨간 줄이 생기는 이유는 사용하지 않았다는 오류 발생이 원인 (ESLint를 사용하는 경우)

```jsx
import { Func1, Func2 } from "./Component2";
```

import 한 컴포넌트 사용

```jsx
return (
  <>
    <Component1 />
    <Func1 />
    <Func2 />
    <div>
      <a href="https://vitejs.dev" target="_blank">
```

하나의 함수에 default를 적용하는 경우

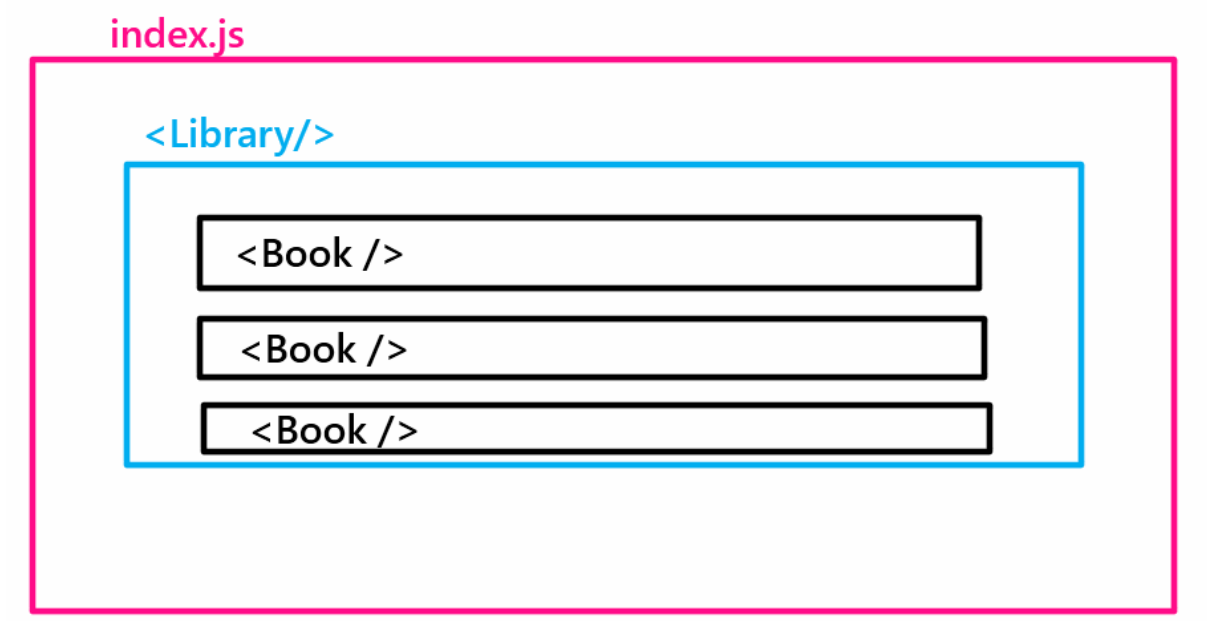
```
export default function Func1() {
  return (
    <>
      <h2>hello Fun1</h2>
    </>
  );
}

export function Func2() {
  return (
    <>
      <h2>hello Fun2</h2>
    </>
  );
}
```

default를 작성한 함수는 { } 로 import를 하지 않으며, 아래와 같이 import 문장 작성

```
import Func1 from "./Component2";
import { Func2 } from "./Component2";
```

실습하기

- 프로젝트에 componets 폴더를 하나 만든다.
- 만든 폴더 안에 Book.jsx 파일을 만들고 Book 함수 컴포넌트를 만든다.
- 같은 폴더에 Library.jsx 파일을 만들고 Library 함수 컴포넌트를 만든다.



step01-jsx > src > components > Book.js > default

```
1
2  function Book(){
3      return(
4          <>
5          <h1>이 책은 Spring 책입니다.</h1>
6          <h2>이 책의  총 페이지 수는 200 페이지이로 이뤄져 있습니다.</h2>
7          </>
8      );
9  }
10
11 export default Book;
12
```

step01-jsx > src > components > Library.jsx > ...

```
1
2   import Book from "./Book";
3
4   function Library(){
5       return(
6           <>
7           <Book/>
8           <Book/>
9           <Book/>
10          </>
11      )
12  }
13
14  export default Library;
```

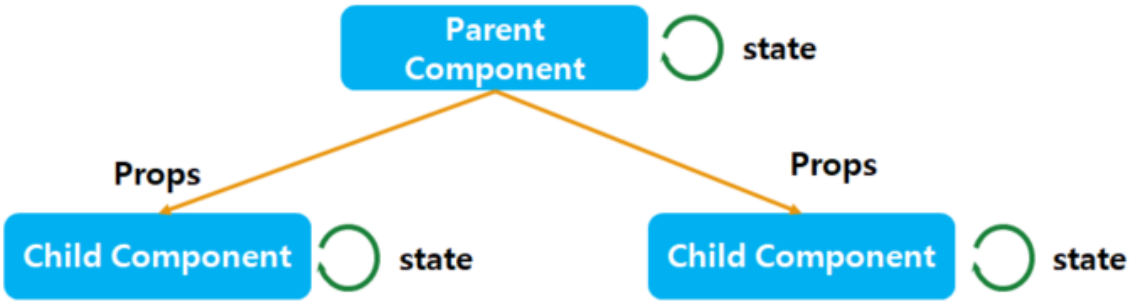
8253jang@daum.net
장희정

```
step01-jsx > src > JS index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import Library from './components/Library';
4
5  const root = ReactDOM.createRoot(document.getElementById('root'));
6  root.render(
7    <Library/>
8  );
9
10
```

실행결과

React 상태관리

리액트 컴포넌트에서 다루는 데이터는 props 와 state로 나누어 진다.



☞ **State란**

- State는 상태를 의미한다.

클래스컴포넌트 : this.state 객체를 통해 관리되며, this.setState() 메서드를 통해 업데이트 한다.

함수형컴포넌트 : React Hook인 useState를 사용하여 상태를 관리한다.

- 컴포넌트 내부에서 변경 가능한 데이터를 뜻 한다.

- 상태는 사용자 인터랙션, API 호출 등을 통해 변경된다.

- 상태가 변경될 때마다 컴포넌트는 자동으로 재 렌더링되어 UI 업데이트 된다.

- state는 컴포넌트 내부에서 선언하며 내부에서 값을 변경 할 수 있다.

☞**Props란**

부모 컴포넌트로부터 자식 컴포넌트에 데이터를 보낼 수 있게 해주는 방법

- properties의 약자로 부모 컴포넌트에서 자식 컴포넌트로 전달되는 데이터(객체, 배열 등)나 함수 컴포넌트 간의 데이터를 전달하기 위해 사용되는 리액트 객체이다.

- Parent Component는 Child Component에게 props를 통해 데이터를 전달한다.

- 자식 컴포넌트에서는 props를 받아 오기 만 하고 props를 수정 할 수 없다.

Props 사용하기

```jsx
import Child from "./Child";

function App() {
  return (
    <>
      <h2>App.jsx</h2>
      <Child />
    </>
  );
}

export default App;
```

```jsx
function Child() {
  return (
    <>
      <h2>Child.jsx</h2>
    </>
  );
}
export default Child;
```

props 전달

import 한 컴포넌트 태그 내에 속성이름=값 형태로 작성하여 전달

```jsx
<Child p1="안녕하세요" />
```

1)

```jsx
function Child( { p1 } ) {
  return (

  <>
    <h2> Child.jsx </h2>
    <h3> {p1} </h3>
  </>

  );
}

export default Child;
```

2)

```jsx
function Child(props) {
  const p1 = props.p1;
  return (
    <>
      <h2>Child.jsx</h2>
      <h3>{p1}</h3>
    </>
  );
}
export default Child;
```

1)이지미 : 전달받은 props를 직접 받으려면 {} 안에 속성 이름 작성

2)이미지 : 함수의 매개변수를 props로 작성한 다음 props.속성이름 형태로 값을 사용할 수 있음

8253jang@daum.net
장희정

여러 개의 값 전달

숫자를 보내는 경우 {} 안에 작성

```
<Child2 p1="hello" p2={100} p3="반갑습니다" />
```

1)

```
function Child2({ p1, p2, p3 }) {
 return (
     <>
         <h2> Child2 jsx </h2>
         <h3> {p1} </h3>
         <h3> {p2} </h3>
         <h3> {p3} </h3>
     </>

 );
}

export default Child2;
```

2)

```
function Child2(props) {
  return (
    <>
      <h2>Child2.jsx</h2>
      <h3>{props.p1}</h3>
      <h3>{props.p2}</h3>
      <h3>{props.p3}</h3>
    </>
  );
}
export default Child2;
```

1)이지미 : 중괄호 안에 전달받는 속성 이름을 콤마(,)로 구분하여 나열

2)이미지 : props로 받는 경우

JavaScript Object를 포함한 여러 개의 값 전달

Object로 전달하는 데이터는 {{ }} 중괄호 2개를 사용하여 내부에 이름과, 값을 작성

```
<Child3
  p1="hello"
  p2={100}
  p3="반갑습니다"
  person={{ name: "이름1", age: 20 }}
/>
```

```
function Child3({ p1, p2, p3, person }) {
  return (
    <>
      <h2>Child2.jsx</h2>
      <h3>{p1}</h3>
      <h3>{p2}</h3>
      <h3>{p3}</h3>
      <h2>이름: {person.name}</h2>
      <h2>나이: {person.age}</h2>
    </>
  );
}
export default Child3;
```

```
function Child3( props ) {
 return (
     <>
         <h2>Child2.jsx</h2>
         <h3>{p1}</h3>
         <h3>{p2}</h3>
         <h3>{p3}</h3>
         <h2>이름: {props.person.name}</h2>
         <h2>나이: {props.person.age}</h2>
     </>
    );
 }

 export default Child3;
```

실습해보자

☞ 위 Library 와 Book 예제의 Book 정보(책이름, 총페이지수)를 Props를 이용해 수정해보자.

step01-jsx > src > components > Library.jsx > default

```
1
2   import Book from "./Book";
3
4   const Library = () => {
5       return(
6           <>
7            <Book name="SpringBoot" numPage={300}/>
8            <Book name="JPA" numPage={200}/>
9            <Book name="React" numPage={350}/>
10          </>
11      )
12  };
13  export default Library;
```

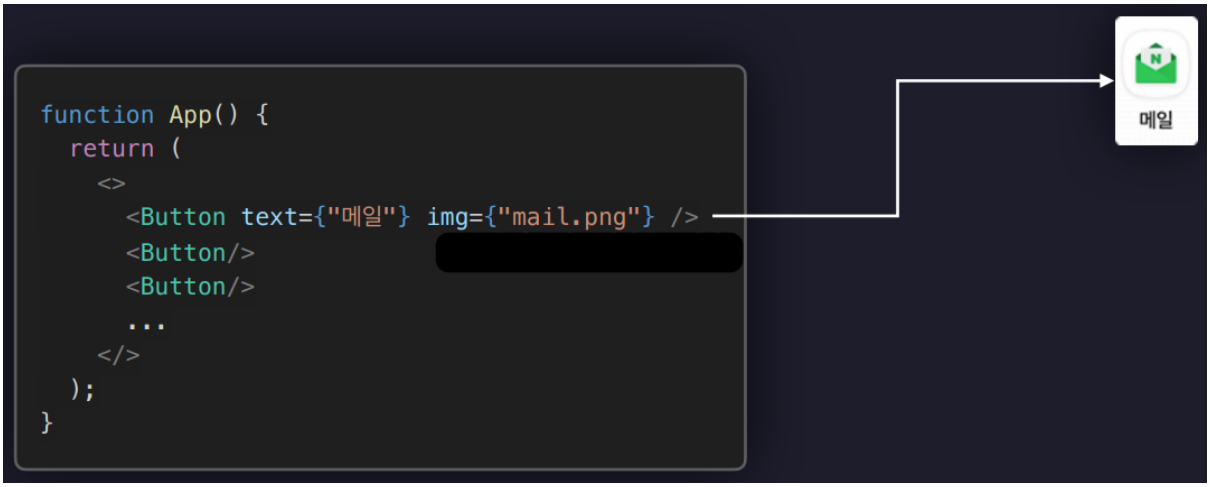
8253jang@daum.net
장희정

step01-jsx > src > components > Book.js > Book

```jsx

const Book = (props)=>{
    return(
        <>
        <h1>이 책은 {props.name} 책입니다.</h1>
        <h2>이 책의  총 페이지 수는 {props.numPage} 페이지이로 이뤄져 있습니다.</h2>
        </>
    );
}

export default Book;

```

실습 – Button 컴포넌트

```
function App() {
  return (
    <>
      <Button/>
      <Button/>
      <Button/>
      ...
    </>
  );
}
```



step02_vite > src > components > Button.jsx > Button

```jsx
import './ButtonStyle.css';

const Button =({text, img})=>{
        return(
        <div className='divBtn'>
         <img src={img}/>
         <button>{text}</button>
        </div>
    )
}

export default Button;
```

ButtonStyle.css

step02_vite > src > components > # ButtonStyle.css > img

```css
.divBtn{
    display: flex;
    flex-direction: column;
}

img{
    width: 50px;
}
```

```
6
7   import mail from "./assets/mail.png";
8   import location from "./assets/location.png";
9   import search from "./assets/search.png";
10
11  createRoot(document.getElementById('root')).render(
12    <StrictMode>
13      {/* <App /> */}
14      <div className="divButton">
15        <Button text={"메일"} img={mail}/>
16        <Button text={"위치"} img={location}/>
17        <Button text={"검색"} img={search}/>
18      </div>
19    </StrictMode>,
20  )
```

React에서 img 넣는 폴더는 어디에?

1) **public 폴더**

정적 자원 : public 폴더에 넣은 파일들은 정적(static) 파일로, 빌드 후 그대로 프로젝트에 포함되어진다.

URL 경로로 접근 : public 폴더에 있는 파일은 웹사이트의 루트 경로에서 직접 접근할 수 있다. 예를 들어, public/images/logo.png 파일은 http://domain:port/images/logo.png 로 접근 가능하다.

변경 불가: public 폴더에 있는 파일들은 React 애플리케이션의 JavaScript 번들에

포함되지 않기 때문에, 동적으로 변경되거나 처리되는 파일이 아니다.

용도 : public 폴더는 index.html 파일, favicon.ico, 로고, 배경 이미지, PDF 파일 등과 같은 앱과 관련된 정적 자원을 넣는 데 사용된다.

```
<img src="/images/logo.png" alt="logo" />
```

2) src/assets 폴더

JavaScript 번들에 포함 : src 폴더 내의 assets 폴더는 React 애플리케이션의 JavaScript 코드에 포함되는 파일들을 관리된다. 이 폴더에 있는 이미지, 글꼴, 스타일시트 파일 등은 JavaScript 번들에 포함되어 빌드 된다.

모듈 시스템 사용 : src 폴더 내의 파일들은 JavaScript 모듈로써 import 를 통해 코드에서 직접 참조하고 사용할 수 있다.

용도 : 애플리케이션에서 사용하는 이미지, 폰트 파일, 스타일시트 등 동적으로 참조되는 자원들은 src/assets 폴더에 넣는다. 이런 자원들은 React 의 빌드 프로세스에서 처리되어, 웹팩(Webpack) 등을 통해 최적화되고 번들에 포함된다.

```
import logo from './assets/images/logo.png';

function App() {
  return <img src={logo} alt="logo" />;
}
```

React 애플리케이션에서 자원을 어떻게 다룰지에 따라 public과 src/assets 폴더를 적절히 사용하면 된다.

용어정리

모듈 : 애플리케이션을 구성하는 개별적인 파일들로, JavaScript, CSS, 이미지 등 다양한 형태의 파일을 말한다.

Webpack : 애플리케이션의 여러 파일을 모듈로 처리하고, 이들을 최적화하여 하나 또는 여러 개의 번들로 묶어주는 도구이다.

번들 : 여러 개의 모듈을 Webpack 이 처리하여 최종적으로 하나 또는 여러 개의 파일로 묶은 결과물이다.

React 애플리케이션에서 Webpack 을 활용하면 애플리케이션의 성능을 최적화하고, 개발 환경에서 효율적으로 작업할 수 있게 된다.

실습 – TripTest.jsx

src < assts리액트 이미지 저장 경로

src /components < Header.jsx | Nav.jsx | Article.jsx | Trip.jsx 추가

실행결과 화면

위 결과에 css와 props 적용해보자.

src> TripTest.css

```
h1{
  color:white; background-color:aqua;
  text-align: center;
  text-shadow: 5px3px3px;
  padding: 10px;
  border-radius: 20px;
}

p{
  color: blue;font-style: italic;font-weight: bold;
}

.imgStyle{
  opacity: 0.7;
```

```
  border-radius: 20px;
  width: 400px;
}
```

Props 전달

8253jang@daum.net
장희정

step02-component > src > JS App.js > App

```
1  import Article from "./Article";
2  import Header from "./Header";
3  import Nav from "./Nav";
4
5
6  function App() {
7    return (
8     <>
9      <Header title="Trip"/>
10     <Nav/>
11     <Article  title="태어난김에 세계일주" body="올 여름 최고의 찬스"/>
12    </>
13    );
14 }
15
16 export default App;
17
```

8253jang@daum.net
장희정

실행결과 화면

Tip

VS_Code에서 component 자동완성 기는 rsc + enter