**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



컴포넌트의 이름은 **반드시 대문자로** 시작

컴포넌트를 어느정도 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.

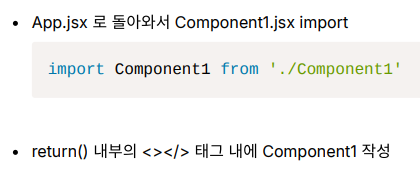
브라우저에서 직접 실행되지 않으며, Babel과 같은 변환기를 통해 Javascript로 변환 후 실행된다.

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

**컴포넌트 생성 – Componet1.jsx**

****

**컴포넌트 적용**

****

**컴포넌트에 함수가 2개 이상인 경우**

Component2.jsx 를 만들고 Fun1, Fun2 함수 작성

함수를 각각 export 하기 위해 함수 이름 앞에 export 작성

****

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

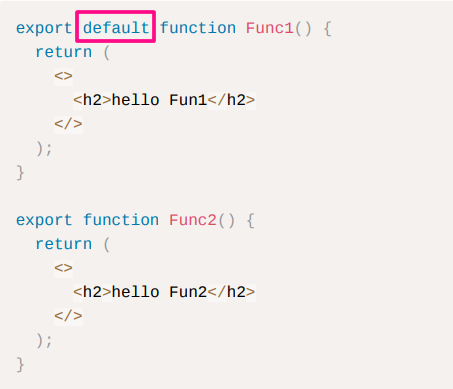
(ESLint를 사용하는 경우)

****

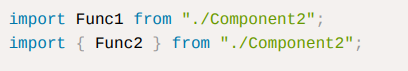
**import 한 컴포넌트 사용**

****

**하나의 함수에 default를 적용하는 경우**



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

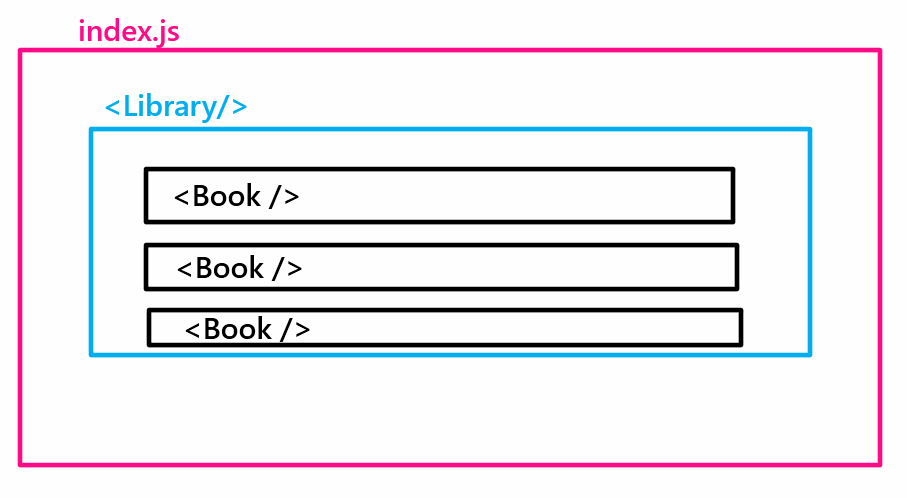
****

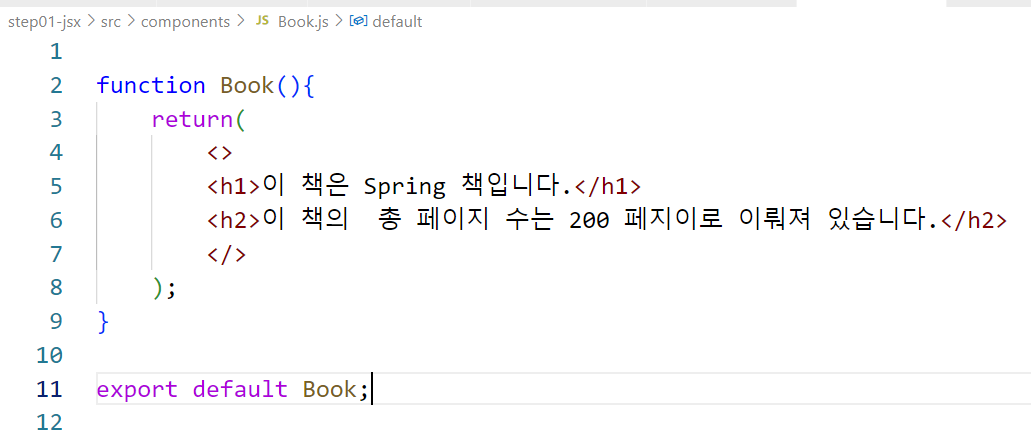
**실습하기**

- 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.







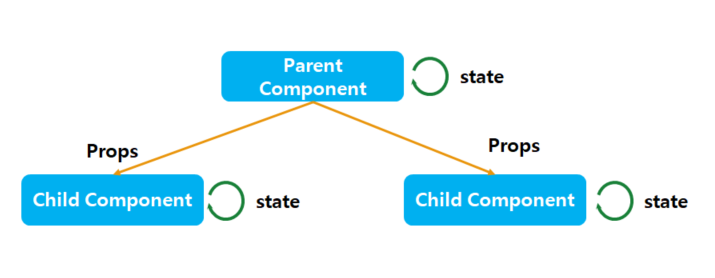


실행결과



**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**

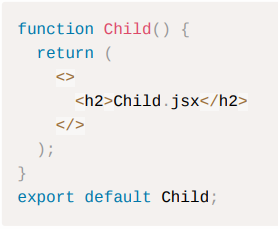
**부모 컴포넌트로부터 자식 컴포넌트에 데이터를 보낼 수 있게 해주는 방법**

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

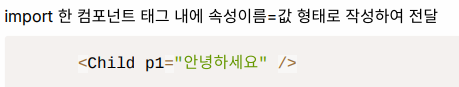
- Parent Component는 Child Component에게 props를 통해 데이터를 전달한다.

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

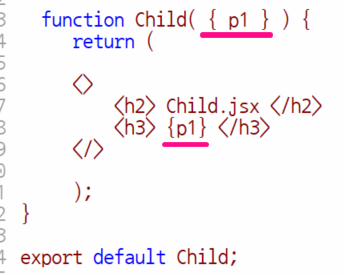
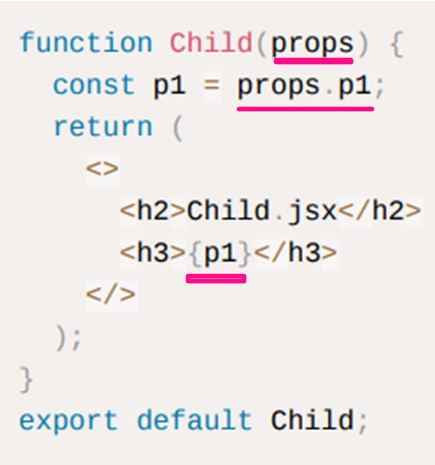
**Props 사용하기**

** **

**props 전달**



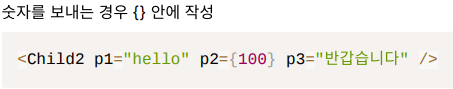
1. 2)

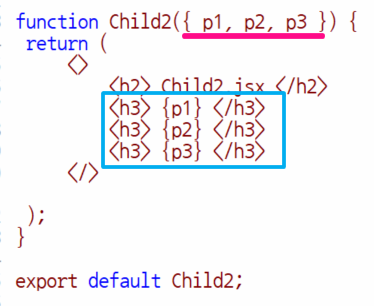
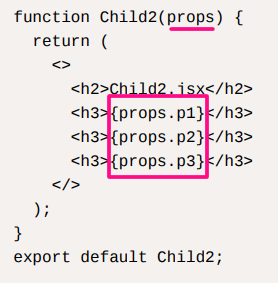
1)이지미 : 전달받은 props를 직접 받으려면 {} 안에 속성 이름 작성

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

**여러 개의 값 전달**

****

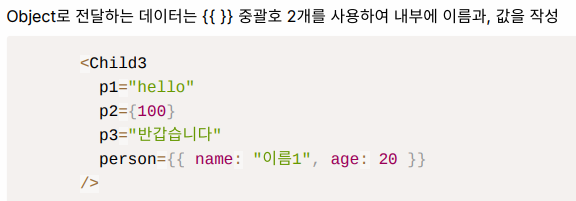
1. **2)**

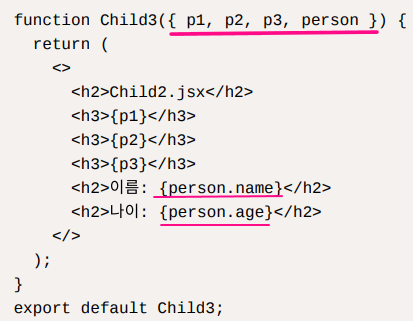
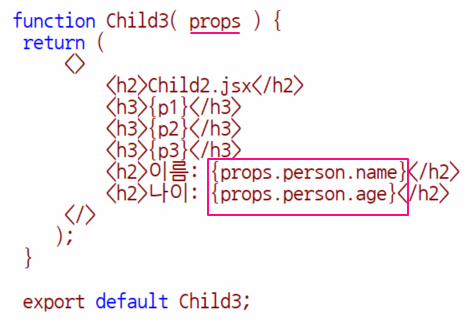
** **

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

2)이미지 : props로 받는 경우

**JavaScript Object를 포함한 여러 개의 값 전달**

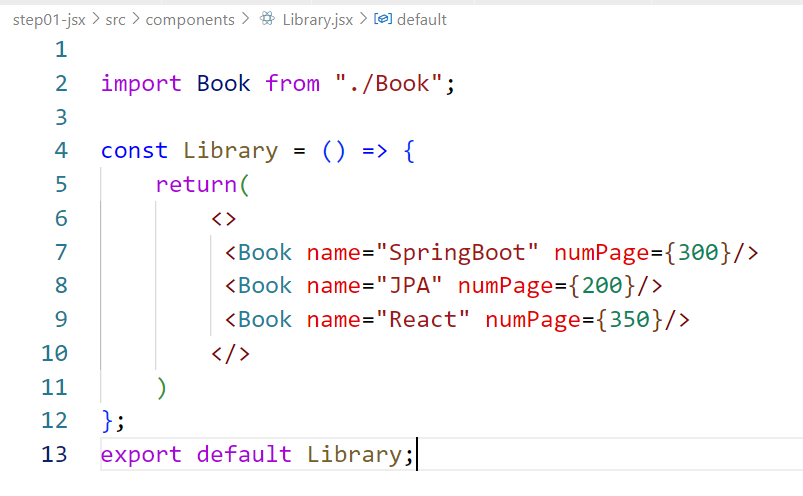
****

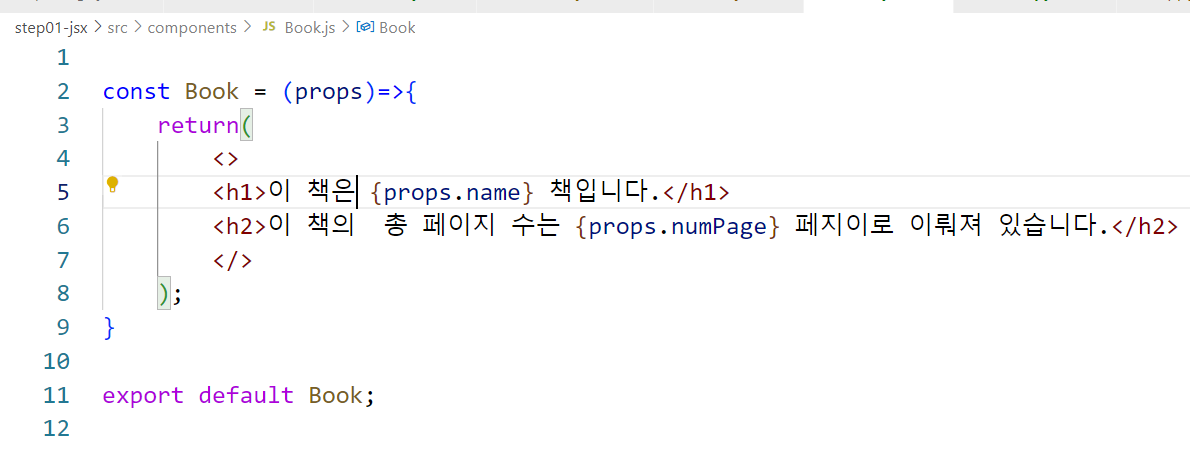
** **

**-----------------------------------------------------------------------------------**

**실습해보자**

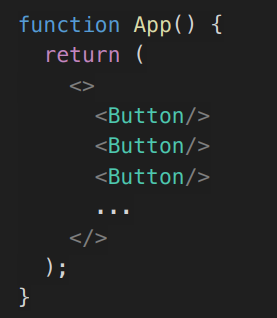
**☞ 위 Library 와 Book 예제의 Book 정보(책이름, 총페이지수)를 Props를 이용해 수정해보자.**

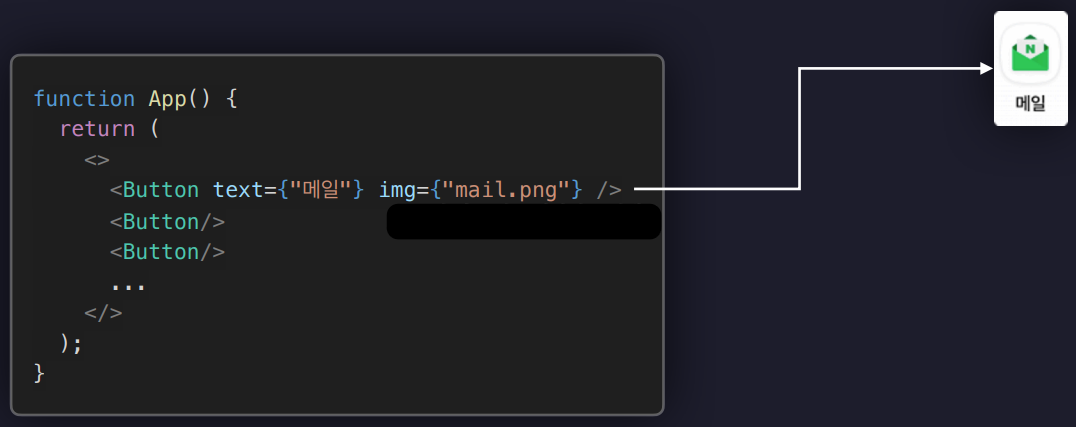
****

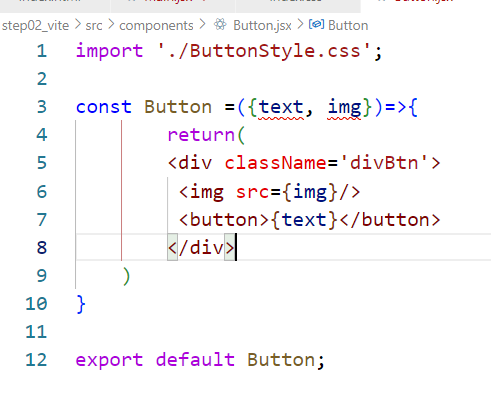
****

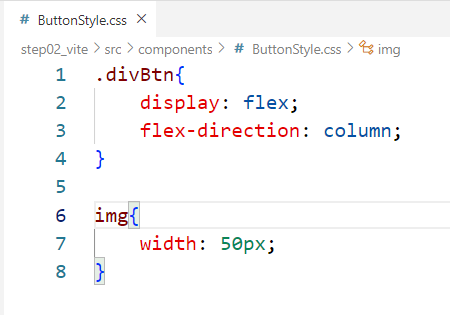
**실습 – Button 컴포넌트**

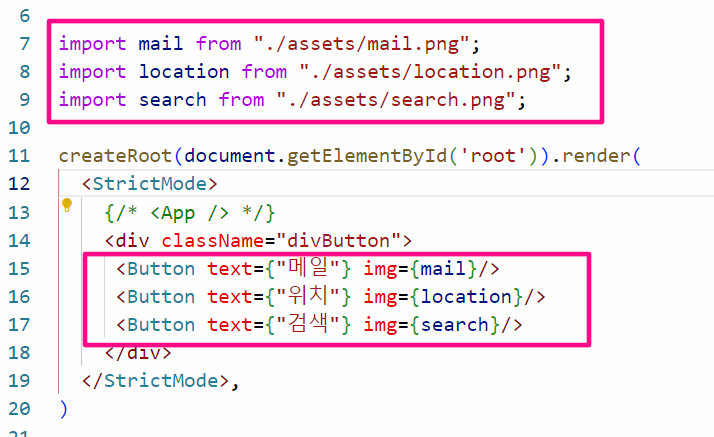
****

****

****

****

****

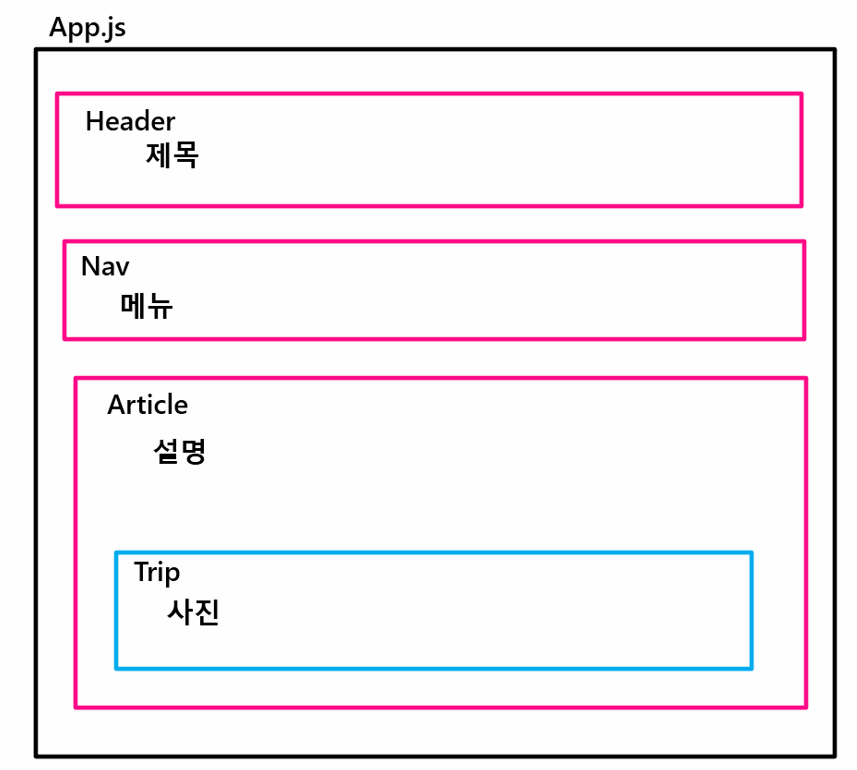
****

****

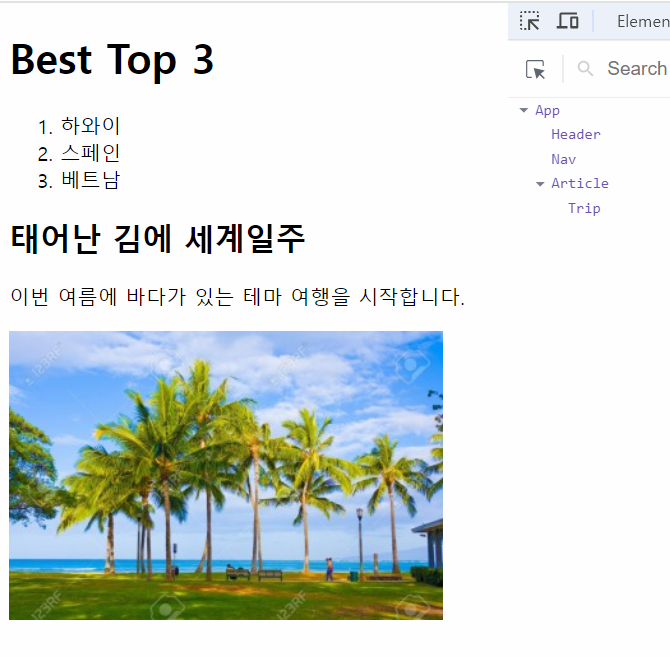
**실습 – Trip.js 생성(이미지만 보여주는 컴포넌트)**

src < assts리액트 이미지 저장 경로

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



**실행결과 화면**



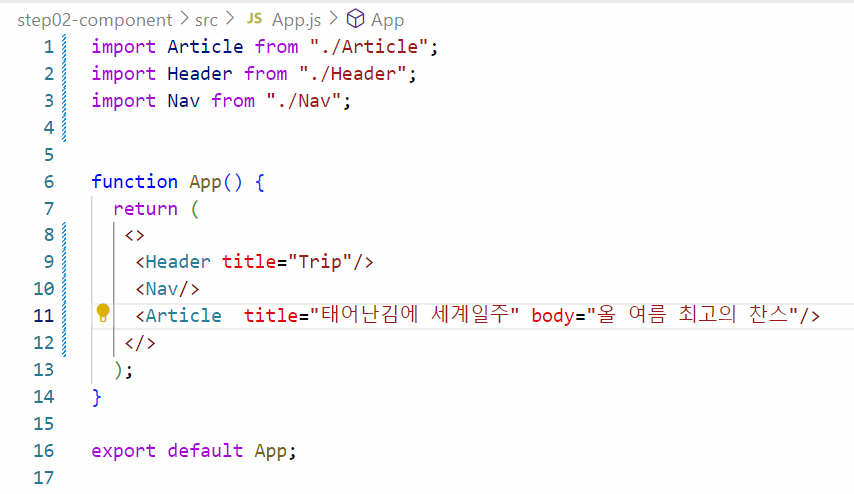
**위 결과에 css와 props 적용해보자.**

**src> index.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;  } |



**Props 전달**

****

**실행결과 화면**

