

컴포넌트 만들기

1. 비슷한 컴포넌트 여러 개 만들기

1 비슷한 컴포넌트 여러 개 만들기

- ❖ 우리는 앞에서 컴포넌트를 많이 만들 때 `<Food />` `<Food />` `<Food />` ...와 같이 컴포넌트를 직접 입력했다.
- ❖ 이게 최선의 방법일까?
- ❖ 어떻게 하면 컴포넌트를 효율적으로 출력할 수 있는지 알아보자.

1 비슷한 컴포넌트 여러 개 만들기

❖ 앞에서 만든 컴포넌트 형태 다시 살펴보기

- 우리가 마지막으로 작성한 **App.js** 파일을 다시 열어 코드가 효율적인지 살펴보자.

1	
2	<code>function Food({fav}){</code>
3	<code> return <h1>I like {fav}</h1>;</code>
4	<code>}</code>
5	
6	<code>function App() {</code>
7	<code> return (</code>
8	<code> <div></code>
9	<code> <h1>Hello</h1></code>
10	<code> <Food fav="kimchi" /></code>
11	<code> <Food fav="ramen" /></code>
12	<code> <Food fav="samgiopsal" /></code>
13	<code> <Food fav="chukumi" /></code>
14	<code> </div></code>
	<code>...</code>

1 비슷한 컴포넌트 여러 개 만들기

❖ 앞에서 만든 컴포넌트 형태 다시 살펴보기

- 이 코드는 효율적이지 않다.
- 왜냐하면 새 음식을 추가할 때마다 `<Food fav= "... " />`를 복사해야 하기 때문이다.
- 만약 음식이 1,000개라면 난리가 날 것이다.
- 1,000개를 반복해서 작성해야 하고, 그때마다 `fav props`에 다른 값을 입력해 줘야하기 때문이다.
- 또 서버에서 음식 데이터를 받아 출력하는 경우, 음식 데이터의 개수를 알 수 없다면 이 방법은 점점 더 문제가 될 것이다.
- 그때그때 서버에서 넘어오는 데이터 개수만큼 컴포넌트를 작성할 수도 없는 노릇이다.
 - 5개가 넘어오면 컴포넌트를 5개 작성하고, 3개가 넘어오면 컴포넌트를 3개 작성하고
- 지금부터 이 문제를 해결하는 방법을 알아볼 것이다.
- 다만! 우리는 아직 서버에서 데이터를 받아오는 방법을 모르니까 일단 서버에서 데이터를 받았다고 가정하고, 그 데이터를 출력하는 방법을 알아볼 것이다.

1 비슷한 컴포넌트 여러 개 만들기

❖ 음식 데이터 만들기

- 서버에서 넘어온 데이터를 저장할 수 있도록 `foodLike`라는 변수를 만든 다음 빈 배열을 할당하자.
- 그리고 아쉽지만 `App` 컴포넌트 안에 한 땀 한 땀 입력했던 `Food` 컴포넌트는 모두 삭제하자.

1	
2	<code>function Food({fav}){</code>
3	<code> return <h1>I like {fav}</h1>;</code>
4	<code>}</code>
5	
6	<code>const foodLike = [];</code>
7	
8	<code>function App() {</code>
9	<code> return (</code>
10	<code> <div></code>
11	<code> <h1>Hello</h1></code>
12	<code> </div></code>
13	<code>);</code>
	<code>...</code>

1 비슷한 컴포넌트 여러 개 만들기

❖ 음식 데이터 만들기

- 서버에서 데이터가 넘어온다고 상상하면서 다음과 같이 코드를 작성해 보자.
- **image** 키값의 경우 인터넷에서 찾은 이미지의 주소를 복사하여 붙여 넣은 거라 직접 입력하지 않아도 된다.

	...
6	const foodLike = [
7	{
8	name: 'Kimchi',
9	image: 'https://cbmpress.sfo2.digitaloceanspaces.com/tfood/2516102861_Dou6sN2f_83893dfb9a46cdd27c7d3d51eff246dc766b2dc8.jpg'
10	},
11	{
12	name: 'Samgyeopsal',
13	image: 'https://pds.joongang.co.kr/news/component/htmlphoto_mmdata/201702/27/117f5b49-1d09-4550-8ab7-87c0d82614de.jpg'
14	},
15	{
16	name: 'Bibimbap',

1 비슷한 컴포넌트 여러 개 만들기

❖ 음식 데이터 만들기

- 서버에서 데이터가 넘어온다고 상상하면서 다음과 같이 코드를 작성해 보자.
- **image** 키값의 경우 인터넷에서 찾은 이미지의 주소를 복사하여 붙여 넣은 거라 직접 입력하지 않아도 된다.

17	image: 'https://health.chosun.com/site/data/img_dir/2021/01/27/2021012702508_0.jpg'
18	},
19	{
20	name: 'Doncasu',
21	image: 'https://img.hankyung.com/photo/202105/01.26364315.1.png'
22	},
23	{
24	name: 'Kimbap',
25	image: 'https://cdn.mkhealth.co.kr/news/photo/202008/img_MKH200810001_0.jpg'
26	}
27];
28	
29	function App() {

1 비슷한 컴포넌트 여러 개 만들기

❖ 음식 데이터 만들기

- 서버에서 데이터가 넘어온다고 상상하면서 다음과 같이 코드를 작성해 보자.
- **image** 키값의 경우 인터넷에서 찾은 이미지의 주소를 복사하여 붙여 넣은 거라 직접 입력하지 않아도 된다.

30	return (
31	<div>
32	<h1>Hello</h1>
33	</div>
34);
35	}
36	
37	export default App;

- 조금 더 의미있는 코딩을 하기 위해서 **foodLike**에 음식의 이름(**name**)과 이미지(**image**)를 추가했다.
- 이제 **foodLike**에 있는 데이터를 이용하여 여러 개의 컴포넌트를 만들기만 하면 된다.
- 그럼 무엇을 알아야 할까?

2 map() 함수로 컴포넌트 많이 만들기

- ❖ foodLike에 있는 데이터로 컴포넌트를 여러 개 만들려면 자바스크립트 함수 `map()`의 사용 방법을 알아야 한다.
- ❖ `map()` 함수의 동작을 알아보기 위해 브라우저의 콘솔을 사용할 것이다.
- ❖ 브라우저를 켜 다음 F12를 누르면 콘솔이 실행된다.

2 map() 함수로 컴포넌트 많이 만들기

❖ map() 함수 사용법 알아보기

- 브라우저에서 콘솔을 연 다음 다음과 같이 코드를 입력해 보자.
- 만약 `node`를 사용할 줄 안다면 `node`를 사용해도 된다.

```
const friends = ["dal", "mark", "lynn", "japan guy"]
```

- `friends`에 친구 4명의 이름(문자열)을 배열에 담아 저장했다.
- 콘솔에 `friends`를 입력해 보면 친구 4명의 이름이 배열로 저장된 것을 확인할 수 있다.

```
> friends  
< ▶ (4) ['dal', 'mark', 'lynn', 'japan guy']  
> |
```

- 이 배열에 들어 있는 이름(문자열) 각각에 작은 하트를 붙여 보자.
- 바로 `map()` 함수를 사용할 순간이 왔다.
- `map()` 함수는 배열의 모든 원소마다 특정 작업을 하는 함수를 적용하고, 그 함수가 반환한 결과를 모아서 배열로 반환해 준다.
- 좀 더 실습을 진행하면서 알아보자.

2 map() 함수로 컴포넌트 많이 만들기

❖ map() 함수 사용법 알아보기

- 다음과 같이 콘솔에 코드를 입력해 보자.
- **map()** 함수의 첫 번째 인자로 특정 작업을 하는 함수를 전달한 것이다.


```
> friends.map(current => {  
  console.log(current);  
  return 0;  
})
```

```
> friends.map(current => {  
  console.log(current);  
  return 0;  
})  
dal VM472:2  
mark VM472:2  
lynn VM472:2  
japan guy VM472:2  
< ▶ (4) [0, 0, 0, 0]  
>
```

2 map() 함수로 컴포넌트 많이 만들기

❖ map() 함수 사용법 알아보기

- dal, mark, lynn, japan guy가 출력된 다음 배열 [0, 0, 0, 0]이 반환되었다.
- 여기서 map() 함수의 2가지 특징을 알 수 있다.



The diagram shows four names: 'japan guy', 'lynn', 'mark', and 'dal'. Four curved arrows originate from these names and point to the 'current' parameter in the function call 'friends.map((current) => {'.

```
friends.map((current) => {  
  console.log(current);  
  return 0;  
});
```

여기서 current가 출력되고

여기서 0을 반환해

[0, 0, 0, 0]

2 map() 함수로 컴포넌트 많이 만들기

❖ map() 함수 사용법 알아보기

- 첫 번째는 map() 함수의 인자로 전달한 함수는 배열 friends의 원소를 대상으로 실행된다는 것이다.
- friends에는 4개의 원소가 들어 있기 때문에 함수는 4번 실행된다.
- 두 번째는 그 함수가 반환한 값이 모여 배열이 되고, 그 배열이 map() 함수의 반환값이 된다.
- 이 2가지 특징을 잘 기억해야 한다.
- Current 인자에 하트를 추가하여 반환하면 친구들 이름에 하트가 추가된 배열을 만들 수 있다.

2 map() 함수로 컴포넌트 많이 만들기

❖ map() 함수로 이름에 하트 추가한 배열 만들기

- friends에 저장된 값을 콘솔에서 다시 한번 확인하자.

```
> friends  
< ▶ (4) ['dal', 'mark', 'Lynn', 'japan guy']  
> |
```

- friends.map()의 인자로 이름 뒤에 하트를 붙여 주는 함수를 전달하면 될까?
- 함수의 인자 이름을 current 대신 friend라고 하자.
- 중요한 건 인자의 이름이 아니라 인자의 배열에 들어 있는 원소가 1개씩 전달되면서 함수가 반복 실행된다는 것이다.

2 map() 함수로 컴포넌트 많이 만들기

❖ map() 함수로 이름에 하트 추가한 배열 만들기

- friend에 하트를 더하면 이름 뒤에 하트가 붙은 이름을 원소로 가지는 배열을 얻을 수 있다.

```
> friends.map(function(friend) {  
  return friend + "♥";  
})  
< ▶ (4) ['dal♥', 'mark♥', 'Lynn♥', 'japan guy♥']  
> |
```

- 여기서는 화살표 함수가 아니라 이름없는 함수를 전달했다.
- 이름없는 함수의 friend에는 friends 배열의 원소가 하나씩 넘어오고, 그 원소에 하트를 붙여 반환하기 때문에, ["dal ♥", "mark ♥ ", "lynn ♥", "japan guy ♥"] 를 얻을 수 있다.

2 map() 함수로 컴포넌트 많이 만들기

❖ map() 함수로 Food 컴포넌트 많이 만들어 보기

- 자! 이제 브라우저에서 벗어나 VSCode 화면으로 돌아가자.
- 그리고 **foodLike** 배열을 다시 한번 눈으로 확인하면서 **map()** 함수를 어떻게 적용할지 상상해 보자.

	...
6	const foodLike = [
7	{
8	name: 'Kimchi',
9	image: 'https://cbmpress.sfo2.digitaloceanspaces.com/tfood/2516102861_Dou6sN2f_83893dfb9a46cdd27c7d3d51eff246dc766b2dc8.jpg'
10	},
11	{
12	name: 'Samgyeopsal',
13	image: 'https://pds.joongang.co.kr/news/component/htmlphoto_mmdata/201702/27/117f5b49-1d09-4550-8ab7-87c0d82614de.jpg'
14	},
15	{
16	name: 'Bibimbap',

2 map() 함수로 컴포넌트 많이 만들기

❖ map() 함수로 Food 컴포넌트 많이 만들어 보기

- `foodLike.map(...)`과 같이 작성하고, `map()` 함수에 전달할 인자에는 `dish => <Food ... />`와 같이 컴포넌트를 반환하는 함수를 전달하면 될 것 같다.
- `dish`에는 배열에 있는 원소, 즉 객체 `{name : '...', image : '...'}`이 하나씩 넘어올 것이다.
- 이걸 `dish.name`, `dish.image`와 같은 방법으로 컴포넌트에 전달하면 된다.
- 이제 천천히 코딩해 보자.

2 map() 함수로 컴포넌트 많이 만들기

❖ map() 함수로 Food 컴포넌트 많이 만들어 보기

- map() 함수를 foodLike 배열에 적용하여 코드를 작성하자.
- `<h1>Hello</h1>`은 삭제하고, `{foodLike.map(...)}`을 추가하면 된다.
- 그리고 Food 컴포넌트에서 받는 인자를 `{ name }`으로 수정하자.

1		
2	<code>function Food({name}){</code>	
3	<code> return <h1>I like {name}</h1>;</code>	
4	<code>}</code>	
5		
6	<code>const foodLike = [(생략...)];</code>	
7		
8	<code>function App()</code>	
9	<code> return (</code>	
10	<code> <div></code>	
11	<code> {foodLike.map(dish => (<Food name={dish.name} />))}</code>	
12	<code> </div></code>	
13	<code>);</code>	
14	<code>}</code>	
	<code>...</code>	

React App

localhost:3000

I like Kimchi

I like Samgyeopsal

I like Bibimbap

I like Doncasu

I like Kimbap

2 map() 함수로 컴포넌트 많이 만들기

❖ map() 함수로 Food 컴포넌트 많이 만들어 보기

- 여기서 가장 중요한 부분은 `{foodLike.map(dish =>(<Food name={dish.name} />))}`이다.
- `dish`에 `foodLike` 배열에 있는 원소가 하나씩 넘어가고, 그 원소는 `{ name : image : }`와 같은 객체 형태이므로 Food 컴포넌트에 `dish.name`과 같이 음식 이름을 `name props`로 넘겨준 것이다.
- 결국 `map()` 함수는 `[<Food name={...} />, ...]`와 같이 Food 컴포넌트 원소 5개를 가진 배열을 반환할 것이다.
- 그 결과 음식 이름 5개가 화면에 표시된다.
- `map()` 함수의 첫 번째 인자로 넘어가는 함수의 첫 번째 인자인 `dish`에는 `foodLike`의 원소가 하나씩 넘어간다는 점을 꼭 기억해야 한다.
- 영화 앱을 만들 때도 구조가 같은 데이터를 사용할 것이다.
- 이제 음식 이미지까지 출력해 보자.
- Food 컴포넌트에 `dish.image`와 같은 방법으로 `picture props`를 전달하면 될까?.

2 map() 함수로 컴포넌트 많이 만들기

❖ Food 컴포넌트에 음식 이미지 출력하기

- Food 컴포넌트에 `picture props`를 추가하자.
- `picture props`에는 `dish.image`를 전달할 것이다.

1	
2	<code>function Food({name}){</code>
3	<code> return <h1>I like {name}</h1>;</code>
4	<code>}</code>
5	
6	<code>const foodILike = [(생략...)];</code>
7	
8	<code>function App()</code>
9	<code> return (</code>
10	<code> <div></code>
11	<code> {foodILike.map(dish => (</code>
12	<code> <Food name={dish.name} picture={dish.image} /></code>
13	<code>)))</code>
14	<code> </div></code>
15	<code>);</code>
	<code>...</code>

2 map() 함수로 컴포넌트 많이 만들기

❖ Food 컴포넌트에 음식 이미지 출력하기

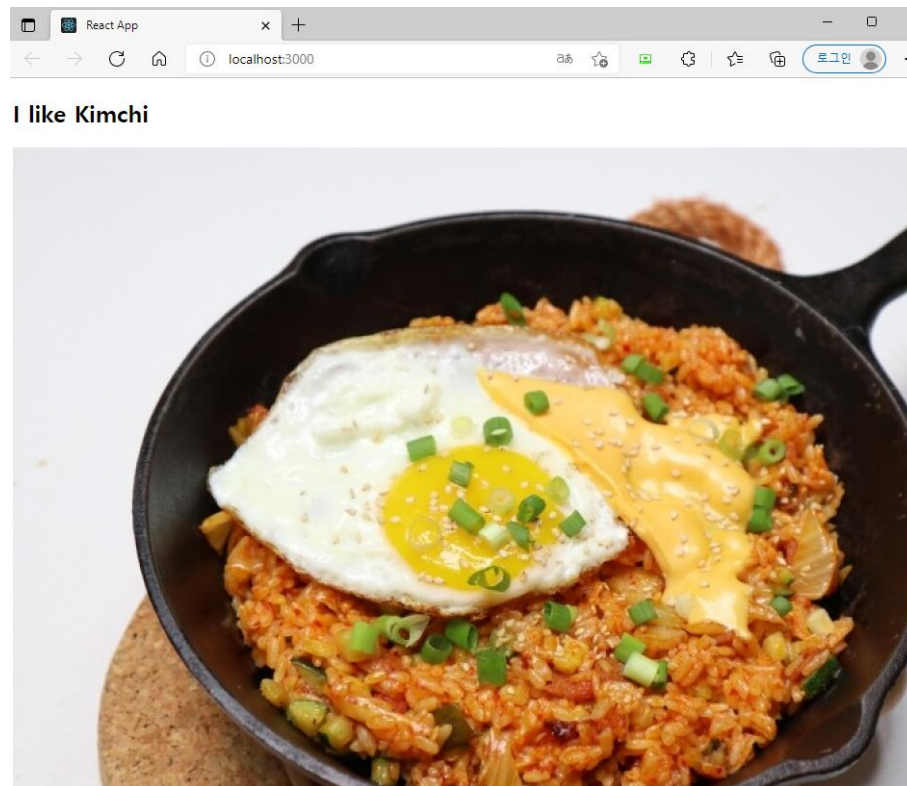
- 코드를 수정하고 저장하면 화면에 아무런 변화가 없을 것이다.
- 아직 전달한 `picture props`를 사용하지 않았기 때문이다.
- `Food` 컴포넌트(함수)에서 `picture props`를 받을 수 있도록 코드를 수정하자.
- 그런 다음 `h1` 엘리먼트를 `h2` 엘리먼트로 바꾸고 `img` 엘리먼트를 추가하자.
- 마지막으로 `div` 엘리먼트로 `h2`, `img` 엘리먼트를 감싸면 된다.

1	
2	<code>function Food({name, picture}){</code>
3	<code> return (</code>
4	<code> <div></code>
5	<code> <h2>I like {name} </h2></code>
6	<code> </code>
7	<code> </div></code>
8	<code>);</code>
9	<code>}</code>
	<code>...</code>

2 map() 함수로 컴포넌트 많이 만들기

❖ Food 컴포넌트에 음식 이미지 출력하기

- 이제 음식의 이름과 이미지가 모두 나타난다.
- 이렇게 `map()` 함수를 사용하면 배열에 데이터가 몇 개 있는지 컴포넌트를 여러 개 손쉽게 출력할 수 있다.



3 음식 앱 이리저리 만지고, 고쳐보기

- ❖ 앞 절에서 만든 음식 앱을 이리저리 만져 보면서 리액트와 `map()` 함수가 어떤 상호 작용을 하는지 조금만 더 자세히 알아보자.
- ❖ 우선 `map()` 함수의 인자로 함수를 전달하도록 만들어 보자.

3 음식 앱 이리저리 만지고, 고쳐보기

❖ map() 함수의 인자로 함수 전달하기

- {foodLike.map(dish =>(<Food name={!dish.name} picture={dish.image} />))}를 {foodLike.map(renderFood)}로 변경하자.

1	
2	function Food({name, picture}){
3	return (
4	<div>
5	<h2>I like {name} </h2>
6	
7	</div>
8);
9	}
10	
11	const foodLike = [(생략...)];
12	
13	function App() {
14	return (
15	<div>
16	{foodLike.map(dish => (

3 음식 앱 이리저리 만지고, 고쳐보기

❖ `map()` 함수의 인자로 함수 전달하기

- `{foodILike.map(dish =>(<Food name={!dish.name} picture={dish.image} />))}`를 `{foodILike.map(renderFood)}`로 변경하자.

17	<code>{foodILike.map(renderFood)}</code>
18	<code></div></code>
19	<code>);</code>
20	<code>}</code>
21	
22	<code>export default App;</code>

3 음식 앱 이리저리 만지고, 고쳐보기

❖ renderFood() 함수 정의하기

- 그런 다음 renderFood() 함수를 정의하자.

1	
2	function Food({name, picture}){
3	return (
4	<div>
5	<h2>I like {name} </h2>
6	
7	</div>
8);
9	}
10	
11	const foodLike = [(생략...)];
12	
13	function renderFood(dish){
14	return <Food name={dish.name} picture={dish.image} />;
15	}
16	

3 음식 앱 이리저리 만지고, 고쳐보기

❖ `renderFood()` 함수 정의하기

- 그런 다음 `renderFood()` 함수를 정의하자.

```
17 function App() {  
18   return (  
19     <div>  
20       {foodILike.map(renderFood)}  
21     </div>  
22   );  
23 }  
24  
25 export default App;
```

- `map()` 함수의 첫 번째 인자로 전달한 화살표 함수를 밖으로 빼서 일반 함수 `renderFood()`로 작성했을 뿐, 음식 앱의 기능이 달라진 건 아닙니다.
- 코드를 저장하고 앱을 실행해 보면 결과 화면은 똑같다.

3 음식 앱 이리저리 만지고, 고쳐보기

❖ renderFood() 함수 정의하기

- 이제 map() 함수의 1 번째 인자로 전달할 renderFood() 함수를 분리했다.
- 그러면 계속해서 map() 함수가 반환하는 값이 구체적으로 무엇인지 자세히 살펴보자.

❖ renderFood() 함수를 화살표 함수로 작성

- renderFood() 함수는 화살표 함수로 작성해도 똑같이 작동한다.
- 만약 화살표 함수로 작성하고 싶다면 다음과 같이 코드를 수정해보자.

```
const renderFood = dish => <Food name={dish.name} picture={dish.image} />;
```

3 음식 앱 이리저리 만지고, 고쳐보기

❖ `map()` 함수의 반환값 살펴보기

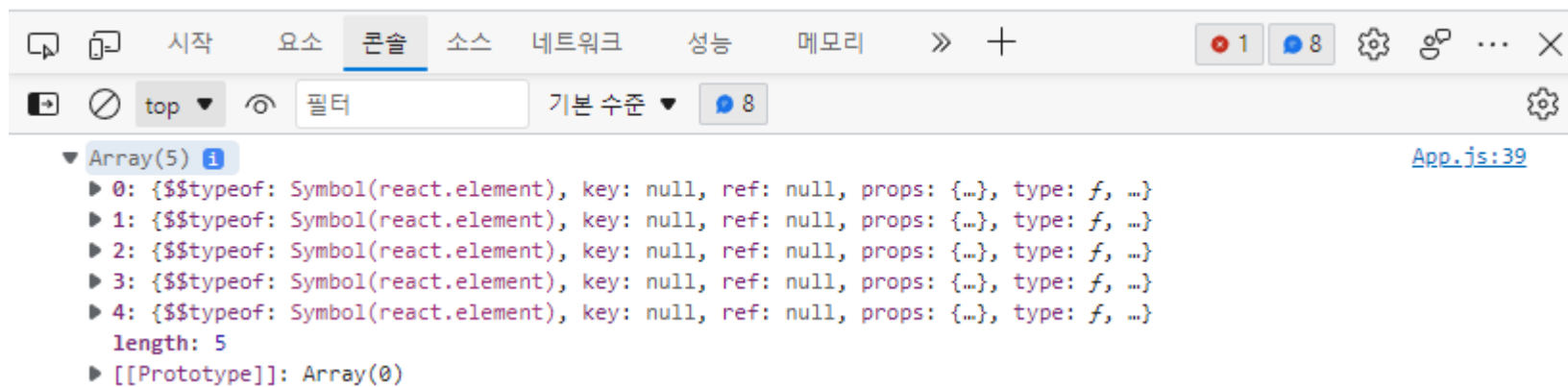
- 다음과 같이 코드를 수정 해서 `map()` 함수의 반환값을 그대로 출력해 보자.
- 그리고 출력한 값을 자세히 살펴보자.

17	<code>function App() {</code>
18	<code> console.log(foodLike.map(renderFood));</code>
19	<code> return (</code>
20	<code> <div></code>
21	<code> {foodLike.map(renderFood)}</code>
22	<code> </div></code>
23	<code>);</code>
24	<code>}</code>
25	
26	<code>export default App;</code>

3 음식 앱 이리저리 만지고, 고쳐보기

❖ map() 함수의 반환값 살펴보기

- [콘솔] 탭을 보면 `Array(5)`가 보인다(경고 메시지는 일단 무시).
- ▶를 눌러서 펼쳐보면, 뭔가 이상한 배열이 출력되고 있다.
- 이게 바로 `map()` 함수가 반환한 리액트 컴포넌트이다.
- 리액트 컴포넌트가 어떤 구조인지 보고 싶다면 또 다시 ▶를 눌러서 펼쳐서 구경하면 된다.



3 음식 앱 이리저리 만지고, 고쳐보기

❖ 음식 앱 다시 원래대로 돌려놓기

- `renderFood()` 함수는 `map()` 함수가 반환한 리액트 컴포넌트를 출력하려고 사용해 본 것이므로 다시 원래대로 코드를 돌려놓자.

1	
2	<code>function Food({name, picture}){</code>
3	<code> return (</code>
4	<code> <div></code>
5	<code> <h2>I like {name} </h2></code>
6	<code> </code>
7	<code> </div></code>
8	<code>);</code>
9	<code>}</code>
10	
11	<code>const foodILike = [(생략...)];</code>
12	
13	<code>function App() {</code>
14	<code> return (</code>
15	<code> <div></code>
16	<code> {foodILike.map(dish => (</code>

3 음식 앱 이리저리 만지고, 고쳐보기

❖ 음식 앱 다시 원래대로 돌려놓기

- `renderFood()` 함수는 `map()` 함수가 반환한 리액트 컴포넌트를 출력하려고 사용해 본 것이므로 다시 원래대로 코드를 돌려놓자.

17	<code><Food name={dish.name} picture={dish.image} /></code>
18	<code>)}}</code>
19	<code></div></code>
20	<code>);</code>
21	<code>}</code>
22	
23	<code>export default App;</code>

- 이렇게 하는 이유는 `App.js` 파일 안에 또 다른 함수를 만들지 않기 위해서이다.
- 함수가 많아지면 나중에 관리하기 어려워기 때문이다.
- 그나저나 콘솔에서 이상한 경고 메시지가 있었다.
- 그건 뭘까?

3 음식 앱 이리저리 만지고, 고쳐보기

❖ `map()` 함수로 만든 컴포넌트에 `key props` 추가하기

- [콘솔] 탭의 경고 메시지를 한 번 읽어보자.

```
⚠ Warning: Each child in a list should have a unique "key" prop.    react-jsx-dev-runtime.development.js:117
Check the render method of `App`. See https://reactjs.org/link/warning-keys for more information.
  at Food (http://localhost:3000/static/js/bundle.js:24:5)
  at App
```

- 해석하면 ‘리스트의 각 원소는 유일한 `"key"` prop을 가져야 한다’는 뜻이다.
- `foodILike` 배열 속성을 보면 `key`의 값이 실제로 없어서(`null`) 이런 메시지가 나온 것이다.
- 리액트의 원소들은 유일해야 하는데 리액트 원소가 리스트에 포함되면서 유일성이 없어진 것이다.

3 음식 앱 이리저리 만지고, 고쳐보기

❖ `map()` 함수로 만든 컴포넌트에 `key props` 추가하기

- 이 문제를 해결하기 위해 `foodLike` 배열 원소에 `id`라는 값을 추가하자.
- 이 값으로 `key`값이 없다는 경고 메시지를 해결할 수 있다.

	...
6	<code>const foodLike = [</code>
7	<code>{</code>
8	<code> id: 1,</code>
9	<code> name: 'Kimchi',</code>
10	<code> image:</code> <code>'https://cbmpress.sfo2.digitaloceanspaces.com/tfood/2516102861_Dou6sN2f_83893dfb9a46cdd27c7d3d51eff246dc766b2dc8.jpg'</code>
11	<code>},</code>
12	<code>{</code>
13	<code> id: 2,</code>
14	<code> name: 'Samgyeopsal',</code>
15	<code> image:</code> <code>'https://pds.joongang.co.kr/news/component/htmlphoto_mmdata/201702/27/117f5b49-1d09-4550-8ab7-87c0d82614de.jpg'</code>
16	<code>},</code>

3 음식 앱 이리저리 만지고, 고쳐보기

❖ map() 함수로 만든 컴포넌트에 key props 추가하기

- 이 문제를 해결하기 위해 foodLike 배열 원소에 id라는 값을 추가함

```
17 {  
18   id: 3,  
19   name: 'Bibimbap',  
20   image:  
    'https://health.chosun.com/site/data/img_dir/2021/01/27/2021012702508_0.jpg'  
21 },  
22 {  
23   id: 4,  
24   name: 'Doncasu',  
25   image: 'https://img.hankyung.com/photo/202105/01.26364315.1.png'  
26 },  
27 {  
28   id: 5,  
29   name: 'Kimbap',  
30   image:  
    'https://cdn.mkhealth.co.kr/news/photo/202008/img_MKH200810001_0.jpg'  
31 }  
32 ];  
...
```

3 음식 앱 이리저리 만지고, 고쳐보기

❖ `map()` 함수로 만든 컴포넌트에 `key props` 추가하기

- 데이터에 `id`를 추가했다.
- 그나저나 왜 이걸 해야 할까?
- 리엑트는 `Food` 컴포넌트가 서로 다르다는 걸 알 방법이 없기 때문이다.
- 그리고 리엑트에게 컴포넌트가 서로 다르다는 것을 알려 주는 방법이 컴포넌트에 `key props`를 추가하는 것이다.

3 음식 앱 이리저리 만지고, 고쳐보기

❖ map() 함수로 만든 컴포넌트에 key props 추가하기

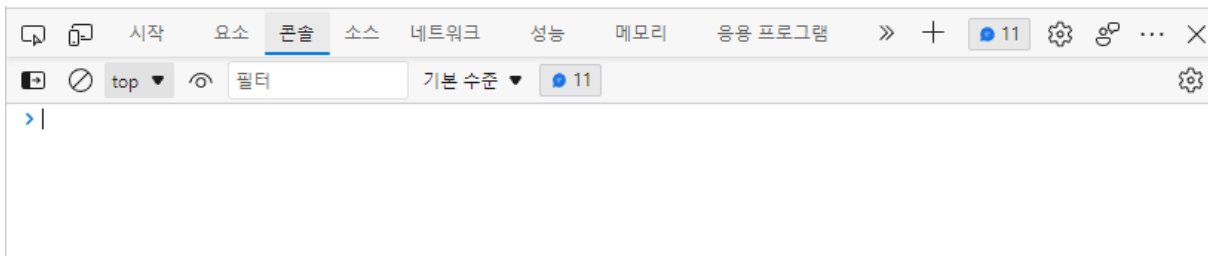
- Food 컴포넌트에 key props를 추가하자.
- key props의 값으로 {dish.id}를 전달하면 된다.

```
39 function App() {  
40   return (  
41     <div>  
42       {foodILike.map(dish => (  
43         <Food key={dish.id} name={dish.name} picture={dish.image} />  
44       ))}  
45     </div>  
46   );  
47 }  
48  
49 export default App;
```

3 음식 앱 이리저리 만지고, 고쳐보기

❖ `map()` 함수로 만든 컴포넌트에 `key props` 추가하기

- 앱을 다시 실행하고 [콘솔] 탭을 확인해 보면 경고 메시지가 없어졌을 것이다.
- 다만! `key props`는 리액트 내부에서 사용되는 특수한 `props`라서 `Food` 컴포넌트에 직접 전달되지 않는다.
- 이 특징을 꼭 기억해야 한다.
- 지금까지 `props`와 `map()` 함수를 사용해서 컴포넌트를 여러 개 만드는 방법을 알아봤다.
- 이것이 영화 앱을 만드는 기본기가 될 것이다.



- 마지막으로! 아까 `Food` 컴포넌트를 수정할 때 `img` 엘리먼트 관련 메시지를 설명하지 않았다.
- 이 메시지는 `img` 엘리먼트에 `alt` 속성을 추가하지 않아서 나타난 것입니다.

3 음식 앱 이리저리 만지고, 고쳐보기

❖ img 엘리먼트에 alt 속성 추가하기

- 다음과 같이 Food 컴포넌트를 수정해 보자.
- alt 속성을 추가하고 거기에 {name}을 대입한 것이다.

1	
2	function Food({name, picture}){
3	return (
4	<div>
5	<h2>I like {name} </h2>
6	
7	</div>
8);
9	}
	...

4 음식 앱에 prop-types 도입하기

- ❖ 음식 앱을 만들면서 `props`와 `map()` 함수의 사용 방법은 이제 어느 정도 익숙해졌을 것이다.
- ❖ 그런데 우리가 정의한 `props`의 값이 컴포넌트에 제대로 전달되지 않으면 어떻게 해야할까?
- ❖ 예를 들어 `picture` `props`에 `{dish.image}`가 아닌 `{true}`를 전달하면 어떻게 될까?
- ❖ 그러면 우리가 원하는 대로 음식 앱이 작동하지 않을 것이다?
- ❖ 이미지가 제대로 나오지 않을 것이다.
- ❖ 바로 이런 경우에 `props`를 검사하는 방법이 필요하다.
- ❖ 이번에는 `props`를 검사하는 방법을 알아보자.
- ❖ `foodLike`에 데이터를 좀 더 추가해 보자.
- ❖ 그래야 `props`를 검사하는 과정이 의미 있어진다.
- ❖ 예를 들어 음식 앱에 ‘평점’이라는 항목이 있다고 해보자.
- ❖ 이 상황을 가정하려면 `foodLike`에 평점이 있어야 한다.

4 음식 앱에 prop-types 도입하기

❖ 음식 데이터에 rating 추가하기

- foodLike 배열의 각 요소에 rating(평점)을 추가하자.
- 값의 자료형은 당연히 Number일 것이다.

```
1
2 function Food({name, picture, rating}){
3   return (
4     <div>
5       <h2>I like {name} </h2>
6       <img src={picture} alt={name} />
7     </div>
8   );
9 }
10
11 const foodLike = [
12   {
13     id: 1,
14     name: 'Kimchi',
15     image:
16       'https://cbmpress.sfo2.digitaloceanspaces.com/tfood/2516102861_Dou6sN2f_83893
17       dfb9a46cdd27c7d3d51eff246dc766b2dc8.jpg',
```

4 음식 앱에 prop-types 도입하기

❖ 음식 데이터에 rating 추가하기

- foodLike 배열의 각 요소에 rating(평점)을 추가하자.
- 값의 자료형은 당연히 Number일 것이다.

16	rating: 5
17	},
18	{
19	id: 2,
20	name: 'Samgyeopsal',
21	image: 'https://pds.joongang.co.kr/news/component/htmlphoto_mmdata/201702/27/117f5b49-1d09-4550-8ab7-87c0d82614de.jpg',
22	rating: 4.9
23	},
24	{
25	id: 3,
26	name: 'Bibimbap',
27	image: 'https://health.chosun.com/site/data/img_dir/2021/01/27/2021012702508_0.jpg',
28	rating: 3.8
29	},

4 음식 앱에 prop-types 도입하기

❖ 음식 데이터에 rating 추가하기

- foodLike 배열의 각 요소에 rating(평점)을 추가하자.
- 값의 자료형은 당연히 Number일 것이다.

```
30  {
31    id: 4,
32    name: 'Doncasu',
33    image: 'https://img.hankyung.com/photo/202105/01.26364315.1.png',
34    rating: 4.3
35  },
36  {
37    id: 5,
38    name: 'Kimbap',
39    image:
40    'https://cdn.mkhealth.co.kr/news/photo/202008/img_MKH200810001_0.jpg',
41    rating: 3.2
42  }
43  ];
```

4 음식 앱에 prop-types 도입하기

❖ 음식 데이터에 rating 추가하기

- foodLike 배열의 각 요소에 rating(평점)을 추가하자.
- 값의 자료형은 당연히 Number일 것이다.


```
44 function App() {  
45   return (  
46     <div>  
47       {foodLike.map(dish => (  
48         <Food key={dish.id} name={dish.name} picture={dish.image} />  
49       ))}  
50     </div>  
51   );  
52 }  
53  
54 export default App;
```

- rating이 포함된 음식 데이터가 준비되었다.
- 이제 rating props를 Food 컴포넌트에 전달하면서 이 값을 검사해 볼 것이다.
- 그러려면 props의 자료형을 검사할 수 있도록 만들어 주는 prop-types라는 도구를 설치해야 한다.

4 음식 앱에 prop-types 도입하기

❖ prop-types 설치하기

- 터미널에 명령어를 입력해서 prop-types를 설치하자.
>npm install prop-types
- package.json 파일을 열어 dependencies 키에 있는 값을 살펴보자.
- 그 중에 prop-types가 있을 것이다.
- 이게 있으면 설치가 잘 된 것이다.

```
{ } package.json > { } scripts >  start
1   {
2     "name": "movie_app_2021",
3     "version": "0.1.0",
4     "private": true,
5     "dependencies": {
6       "@testing-library/jest-dom": "^5.16.1",
7       "@testing-library/react": "^12.1.2",
8       "@testing-library/user-event": "^13.5.0",
9       "prop-types": "^15.8.0",
10      "react": "^17.0.2",
11      "react-dom": "^17.0.2",
12      "react-scripts": "5.0.0",
13      "web-vitals": "^2.1.2"
14    },
```

4 음식 앱에 prop-types 도입하기

❖ prop-types 설치하기

- **prop-types**는 무슨 일을 해줄까?
- 컴포넌트가 전달받은 **props**가 정말 내가 원하는 값인지 확인한다.
- 왜냐하면 개발하다 보면 실수하는 일이 생기기 때문이다.
- 예를 들어 **picture props**를 보내야 하는데 실수로 **image props**를 보낼 수도 있다.
- 이런 경우 **prop-types**를 통해 미리 'Food 컴포넌트는 반드시 **picture prop**가 전달돼야 한다'고 정의할 수 있다.
- 그러면 **picture props**가 아닌 **images props**가 전달되는 경우 오류 메시지가 나타날 것이다.
- 이제 **prop-types**를 사용해 보자.

4 음식 앱에 prop-types 도입하기

❖ prop-types 적용하기

- `import PropTypes from 'prop-types';`를 `App.js` 파일 맨 위에 추가해 주자.
- 그리고 `rating props`를 `Food` 컴포넌트에 전달하자.

1	<code>import propTypes from "prop-types";</code>
2	
3	<code>function Food({name, picture, rating}){</code>
4	<code> return (</code>
5	<code> <div></code>
6	<code> <h2>I like {name} </h2></code>
7	<code> <h4>{rating}/5.0</h4></code>
8	<code> </code>
9	<code> </div></code>
10	<code>);</code>
11	<code>}</code>
12	
13	<code>const foodLike = [</code>
14	<code> {</code>
15	<code> id: 1,</code>

4 음식 앱에 prop-types 도입하기

❖ prop-types 적용하기

- `import PropTypes from 'prop-types';`를 `App.js` 파일 맨 위에 추가해 주자.
- 그리고 `rating props`를 `Food` 컴포넌트에 전달하자.

16	<code>name: 'Kimchi',</code>
17	<code>image:</code> <code>'https://cbmpress.sfo2.digitaloceanspaces.com/tfood/2516102861_Dou6sN2f_83893dfb9a46cdd27c7d3d51eff246dc766b2dc8.jpg',</code>
18	<code>rating: 5</code>
19	<code>},</code>
20	<code>{</code>
21	<code>id: 2,</code>
22	<code>name: 'Samgyeopsal',</code>
23	<code>image:</code> <code>'https://pds.joongang.co.kr/news/component/htmlphoto_mmdata/201702/27/117f5b49-1d09-4550-8ab7-87c0d82614de.jpg',</code>
24	<code>rating: 4.9</code>
25	<code>},</code>
26	<code>{</code>
27	<code>id: 3,</code>
28	<code>name: 'Bibimbap',</code>

4 음식 앱에 prop-types 도입하기

❖ prop-types 적용하기

- `import PropTypes from 'prop-types';`를 `App.js` 파일 맨 위에 추가해 주자.
- 그리고 `rating props`를 `Food` 컴포넌트에 전달하자.

29	<code>image:</code>
	<code>'https://health.chosun.com/site/data/img_dir/2021/01/27/2021012702508_0.jpg',</code>
30	<code>rating: 3.8</code>
31	<code>},</code>
32	<code>{</code>
33	<code>id: 4,</code>
34	<code>name: 'Doncasu',</code>
35	<code>image: 'https://img.hankyung.com/photo/202105/01.26364315.1.png',</code>
36	<code>rating: 4.3</code>
37	<code>},</code>
38	<code>{</code>
39	<code>id: 5,</code>
40	<code>name: 'Kimbap',</code>
41	<code>image:</code>
	<code>'https://cdn.mkhealth.co.kr/news/photo/202008/img_MKH200810001_0.jpg',</code>
42	<code>rating: 3.2</code>

4 음식 앱에 prop-types 도입하기

❖ prop-types 적용하기

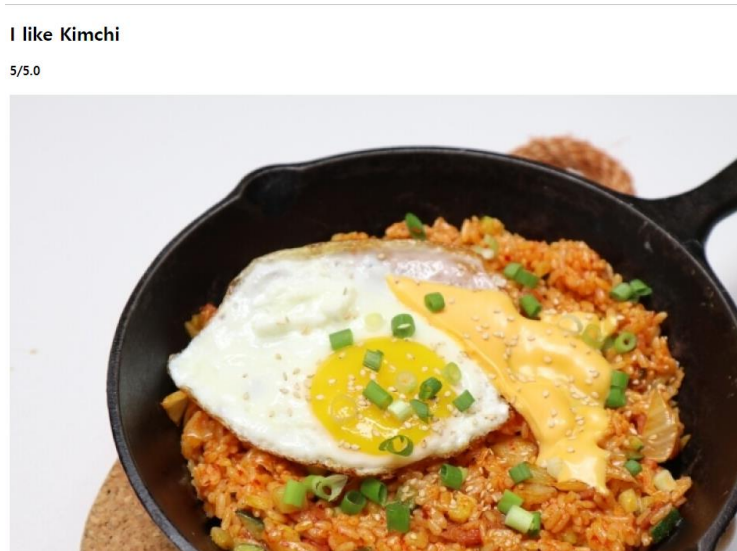
- `import PropTypes from 'prop-types';`를 `App.js` 파일 맨 위에 추가해 주자.
- 그리고 `rating props`를 `Food` 컴포넌트에 전달하자.

```
43   }
44 ];
45
46 function App() {
47   return (
48     <div>
49       {foodILike.map(dish => (
50         <Food key={dish.id} name={dish.name} picture={dish.image}
51         rating={dish.rating} />
52       ))}
53     </div>
54   );
55 }
56 export default App;
```

4 음식 앱에 prop-types 도입하기

❖ prop-types 적용하기

- 아직 prop-types를 적용하진 않았다.
- 그전에 음식 앱을 실행해 보면 rating props로 전달한 값이 잘 출력될 것이다.
- 일단 실행하는 데는 문제가 없다.



- 이제 prop-types# 적용할 차례다.
- 우리가 할 일은 Food.propTypes에 객체를 적어 주기만 하면 된다.
- 이게 어떤 효과를 낼지는 코드를 다 작성한 다음 알아보자.

4 음식 앱에 prop-types 도입하기

❖ prop-types 적용하기

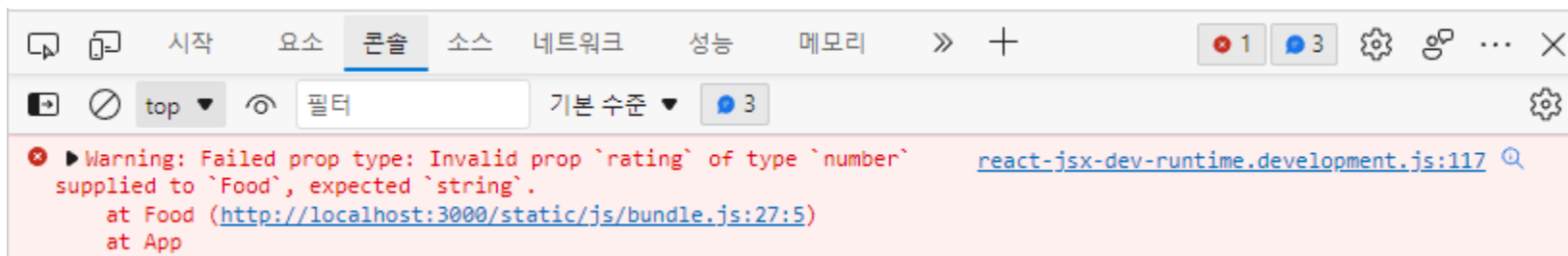
- 다음과 같이 `Food.propTypes`를 작성해 보자.
- 모든 props는 문자열이고 반드시 있어야 한다는 조건을 추가했다.
- 이게 무엇인지는 프로그램을 실행한 다음 알아보자.

1	<code>import propTypes from "prop-types";</code>
2	
3	<code>function Food({name, picture, rating}){ (생략...) }</code>
13	<code>const foodLike = [(생략...)];</code>
46	<code>function App() { (생략...) }</code>
56	<code>Food.propTypes = {</code>
57	<code> name: propTypes.string.isRequired,</code>
58	<code> picture: propTypes.string.isRequired,</code>
59	<code> rating: propTypes.string.isRequired</code>
60	<code>};</code>
61	
62	<code>export default App;</code>

4 음식 앱에 prop-types 도입하기

❖ prop-types 적용하기

- 코드를 작성한 다음 저장하고 음식 앱을 실행해 보면 별 문제가 없어 보일 것이다.
- 하지만 [콘솔] 탭을 확인해 보면 경고 메시지가 있을 것이다.
- 음식 앱을 실행하는 데는 문제가 없지만, 뭔가 검사가 진행되었고 그에 따라 경고메시지를 출력해준 것이다.



- Failed prop type이라는 문장을 보니 prop type에 실패한 거 같다.
- 그리고 'Food' 컴포넌트의 rating props 자료형이 string이어야 하는데, number라서 문제다'라고 이야기하고 있다.

4 음식 앱에 prop-types 도입하기

❖ prop-types 적용하기

- 우리가 작성한 Food.propTypes의 rating 키값을 다시 보자.

	...
56	Food.propTypes = {
57	name: propTypes.string.isRequired,
58	picture: propTypes.string.isRequired,
59	rating: propTypes.string.isRequired
60	};
	...

- rating 키값이 PropTypes.string.isRequired라고 되어 있다.
- String은 '문자열'이라는 뜻이고, isRequired는 '필요하다'라는 뜻이다.
- 이를 합치면 'rating에는 string이라는 자료형이 필요하다'는 뜻일 것이다.
- 그런데 우리가 넘겨준 값의 자료형은 뭐였을까?
- Number였다.
- 그래서 경고 메시지가 나타난 것이다.

4 음식 앱에 prop-types 도입하기

❖ prop-types 경고 해결하기

- 이제 `rating : propTypes.string.isRequired` 대신 `rating : propTypes.number.isRequired`라고 수정해 주자.

	...
56	<code>Food.propTypes = {</code>
57	<code> name: propTypes.string.isRequired,</code>
58	<code> picture: propTypes.string.isRequired,</code>
59	<code> rating: propTypes.number.isRequired</code>
60	<code>};</code>
	...

- 그런 다음 개발자 도구에서 [콘솔] 탭을 확인해 보면 `prop type` 경고 메시지가 사라져 있을 것이다.
- `prop-types`는 또한 `props`의 이름도 검사한다.
- 예를 들어 `Food` 컴포넌트에 전달하는 `picture props`의 이름을 `Food.propTypes`에서 정의한 이름인 `picture`가 아니라 `image`로 바꾸면 어떻게 될까?
- 말 그대로 정의한 이름이 아닌 다른 이름의 `props`를 전달하는 것이다.

4 음식 앱에 prop-types 도입하기

❖ 다른 종류의 prop-types 경고 해결하기

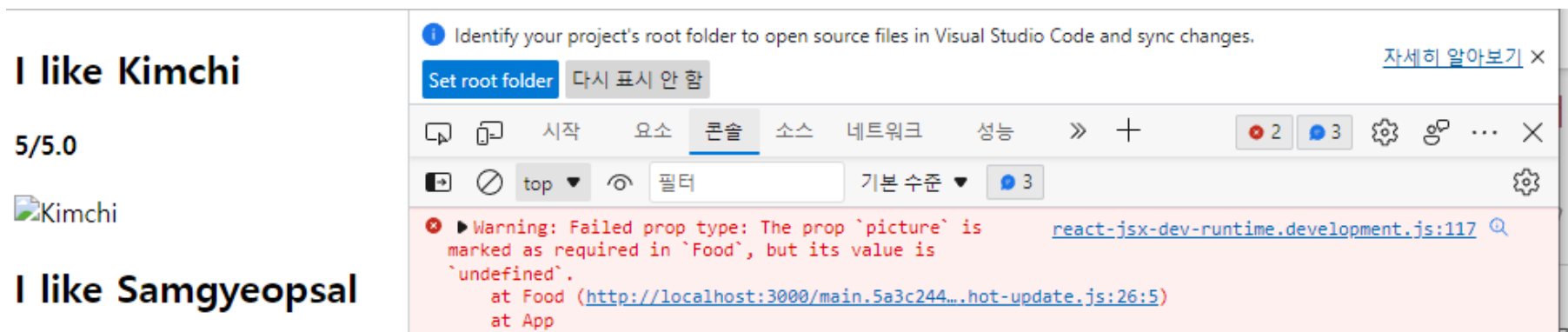
- Food 컴포넌트에 전달하는 picture props의 이름을 image로 바꿔보자.

	...
46	function App() {
47	return (
48	<div>
49	{foodLike.map(dish => (
50	<Food key={dish.id} name={dish.name} image ={dish.image}
	rating={dish.rating} />
51)}}
52	</div>
53);
54	}
55	
56	export default App;

4 음식 앱에 prop-types 도입하기

❖ 다른 종류의 prop-types 경고 해결하기

- 음식 앱을 실행해 보면 화면에 사진이 나오지 않을 것이다.
- [콘솔] 탭의 경고 메시지를 보면 그 이유를 알 수 있다.



- 경고 메시지를 읽어 보면 'Food' 컴포넌트에 **picture**라는 이름의 props가 필요한데, 그 값이 **undefined**라고 말하고 있다.
- Food 컴포넌트에 **picture** props가 아니라 **image** props를 전달했기 때문이다.
 - **picture**라는 이름의 props가 없으니까 **undefined**이다.

4 음식 앱에 prop-types 도입하기

❖ 다른 종류의 prop-types 경고 해결하기

- 이제 코드를 원래대로 돌려놓자.

	...
46	function App() {
47	return (
48	<div>
49	{foodLike.map(dish => (
50	<Food key={dish.id} name={dish.name} picture ={dish.image}
	rating={dish.rating} />
51)}}
52	</div>
53);
54	}
55	
56	export default App;

4 음식 앱에 prop-types 도입하기

❖ 다른 종류의 prop-types 경고 해결하기

- prop-types는 이런 식으로 props를 검사한다.
- 자료형과 그 이름의 값이 전달되었는지도 검사한다.
- 개발자가 실수하지 않도록 예방해 주는 유용한 도구라 할 수 있다.
- 영화 앱을 만들면서 이 도구를 적극적으로 사용할 거니까 prop-types의 사용 방법을 잘 익혀 두자.
- 그나저나 `rating : PropTypes.number.isRequired`에서 `.isRequired`는 필요하다는 뜻이었다.
- 그런데 이건 때에 따라 없어도 된다.
- 아직 평점이 등록되지 않은 영화일 수도 있기 때문이다.

4 음식 앱에 prop-types 도입하기

❖ `isRequired`의 뜻 살펴보기

- `rating`의 `.isRequired`를 제거해 보자.

	...
56	<code>Food.propTypes = {</code>
57	<code> name: propTypes.string.isRequired,</code>
58	<code> picture: propTypes.string.isRequired,</code>
59	<code> rating: propTypes.number.isRequired</code>
60	<code>};</code>
61	
62	<code>export default App;</code>

- `rating : PropTypes.number`라고 작성하면, 이제 `rating` props는 필수가 아니어도 되는 항목이 된다.
- 다만, 값이 전달되는 경우 자료형이 `number`이긴 해야 한다는 뜻이다.
- 영화 앱을 만들면서 `isRequired`는 필요에 따라 추가할 것이다.



Thank You !