iOS개발실무

4주차

담당: 김희숙
(jasmin11@hanmail.net)

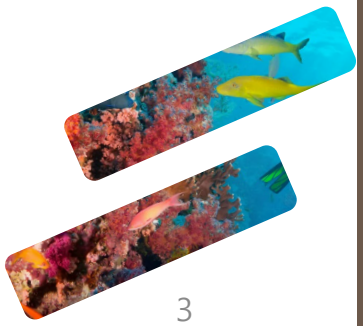
리액트(React)

담당: 김희숙

(jasmin11@hanmail.net)

key

담당: 김희숙
(jasmin11@hanmail.net)



[실습] 반복(IterationSample.js)

map()

❖ IterationSample.js (수정 전)

```
// ./src/Components/IterationSample.js
import React from 'react';

const IterationSample = () => {
  return (
    <ul>
      <li>눈사람</li>
      <li>얼음</li>
      <li>눈</li>
      <li>바람</li>
    </ul>
  );
};

export default IterationSample;
```

- 눈사람
- 얼음
- 눈
- 바람

❖ IterationSample.js (수정 후)

```
// ./src/Components/IterationSample.js
import React from 'react';

const IterationSample = () => {
  const names = ['눈사람', '얼음', '눈', '바람'];
  const nameList = names.map(name => <li>{name}</li>);
  return <ul>{nameList}</ul>;
};

export default IterationSample;
```

```
// 현재는 <li> 태그 간단한 구조이므로 문제가 안될 수 있지만
코드 복잡한 경우, 코드양이 늘어난다
유동적인 데이터인 경우, 다른 방법을 생각해야 한다
```

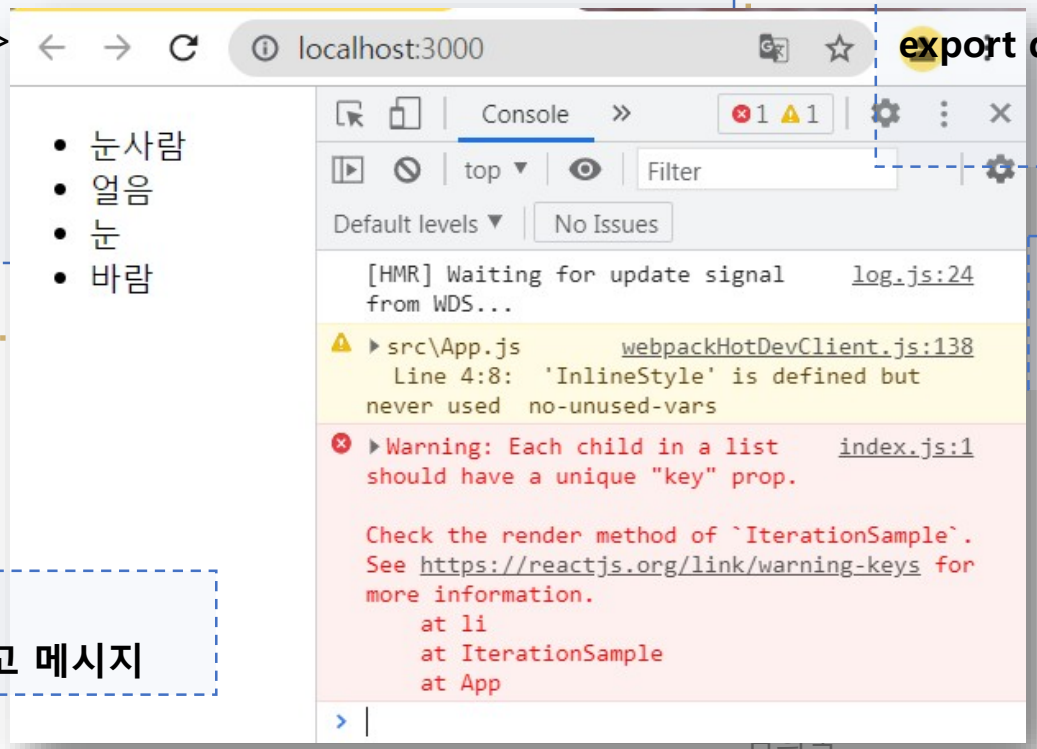
[실습] 반복(IterationSample.js)

map()

❖ App.js

```
//  ./src/App.js
import React from 'react';
import './App.css';
import IterationSample from './Components/IterationSample';

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

export default App;
```

```
// ./src/Components/IterationSample.js
import React from 'react';

const IterationSample = () => {
  const names = ['눈사람', '얼음', '눈', '바람'];
  const nameList = names.map(name => <li>{name}</li>);
  return <ul>{nameList}</ul>;
};

export default IterationSample;
```



// 리액트는 반복문 실행한 HTML 요소에는 반드시 key={}를 적으라고 권장

// 개발자 도구
// key prop 없다는 경고 메시지

[요약] key props

key

배열 ['a', 'b', 'c', d’] → 배열 ['a', 'b', 'z', 'c', d’] // b 다음에 z 가 삽입되는 경우

// key 가 없는 경우

```
<div>a</div>
<div>b</div>
<div>c</div>
<div>d</div>

⇒

<div>a</div>
<div>b</div>
<div>z</div>
<div>c</div>
                삽입 <div>d</div>
```

// key 가 있는 경우

```
<div key={0}>a</div>
<div key={1}>b</div>
<div key={2}>c</div>
<div key={3}>d</div>

⇒

<div key={0}>a</div>
<div key={1}>b</div>
<div key={4}>z</div>
<div key={2}>c</div>
<div key={3}>d</div>
```

[요약] key props

배열 ['a', 'b', 'c', d'] → 배열 ['b', 'c', 'd'] // a 가 삭제되는 경우

// key 가 없는 경우

1번째 a
2번째 b
3번째 c
4번째 d

⇒

삭제 a
1번째 b
2번째 c
3번째 d

// key 가 있는 경우

a
b
c
d

⇒

a 삭제
b
c
d

[요약] Key

Key 문법

- Key: (리액트)
  - 컴포넌트 배열을 렌더링 했을 때 어떤 원소에 변동이 있었는지 알아내기 위해 사용
  - 예) 유동 데이터 다룰 때(원소 새로 생성, 삭제, 수정)

    Key가 없을 때 Virtual DOM을 비교하는 과정에서 리스트를 순차적으로 비교하면서 변화를 감지
  - Key 값은 언제나 유일해야 한다(데이터가 가진 고유한 값을 key 값으로 설정)

```
// 게시판의 게시물을 렌더링한다면
// 게시물 번호를 key 값으로 설정
```

```
const articleList = articles.map(article => {
        <Article
                title={article.title}
                writer={article.writer}
                key={article.id}
        />
});
```

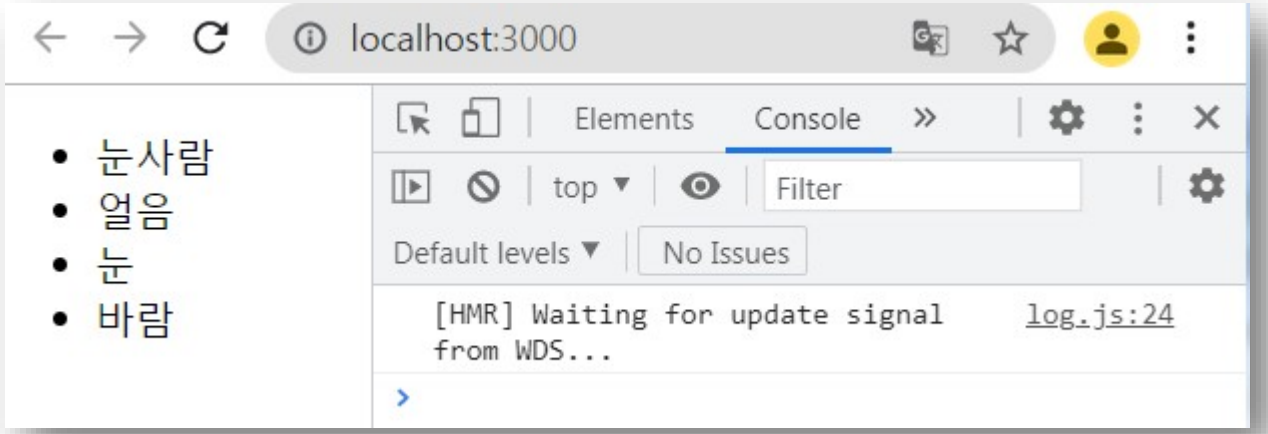
```
// key 값을 설정할 때는 map 함수의 인자로 전달되는
    함수 내부에서
    컴포넌트 props를 설정하듯이 설정한다
```

[실습] 반복(IterationSample.js)

Key 문법

❖ Key 설정

```
// ./src/Components/IterationSample.js
import React from 'react';

const IterationSample = () => {
  const names = ['눈사람', '얼음', '눈', '바람'];
  const nameList = names.map((name, index) =>
                    <li key={index}>{name}</li>);
  return <ul>{nameList}</ul>;
};

export default IterationSample;
```



// index 를 key 로 사용
// 고유번호 없으므로 index 로 key 사용

⇒ 고유한 값이 없을 때만 index 값을 key 로 사용한다

// 배열이 변경될 때 효율적으로 리렌더링 되지 못한다

map() 함수

map()

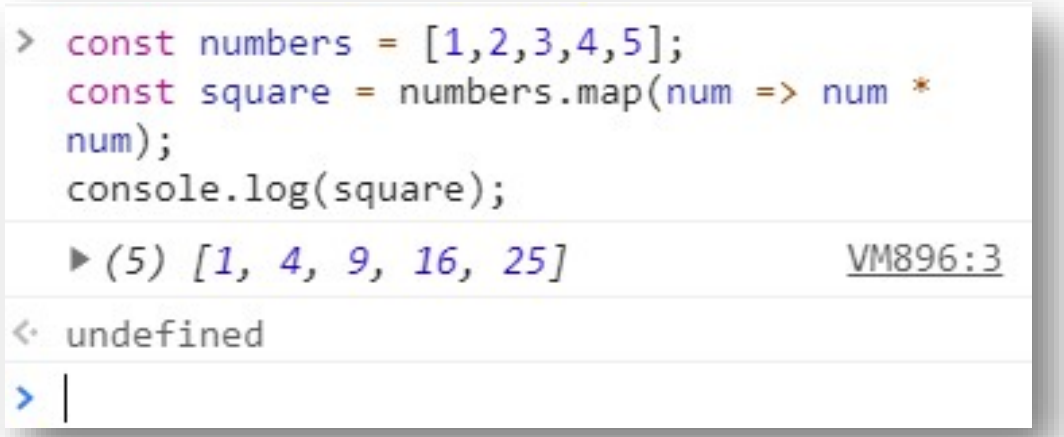
❖ map() 함수

- 배열의 모든 원소마다 특정 작업을 하는 함수를 작용하고 그 함수가 반환한 결과를 모아서 배열로 반환한다
- 자바스크립트 배열 객체의 내장함수 map() 함수
  반복되는 컴포넌트를 렌더링
- 파라미터로 전달된 함수를 사용해서
  배열 내 각 요소를 원하는 규칙에 따라 변환한 후
  그 결과로 새로운 배열을 생성

```
// map() 함수
var numbers = [1,2,3,4,5];
var processed = numbers.map(function(num) {
   return num * num
});
console.log(processed);
```

```
// map() 함수
const numbers = [1,2,3,4,5];
const square = numbers.map(num => num * num);
console.log(square);
```

```
// 배열의 값을 제곱하여 새로운 배열 만든다
```

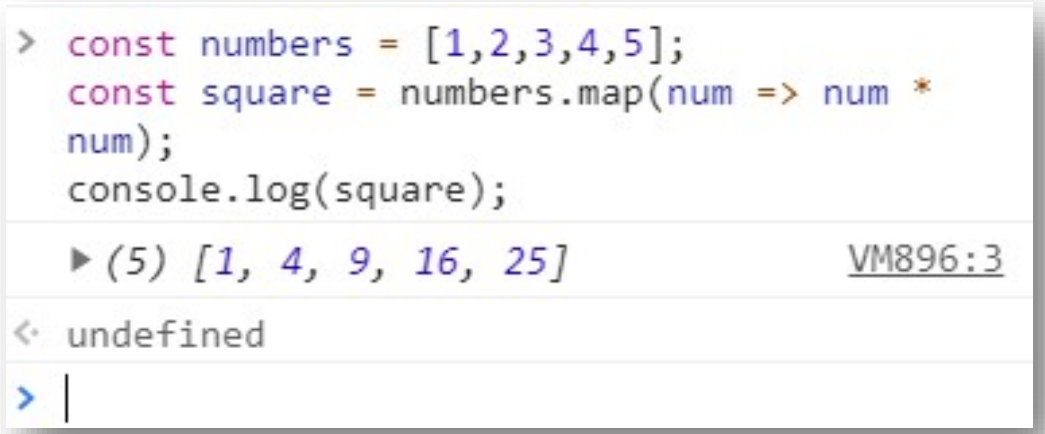


map() 함수

❖ map() 함수

- 배열의 모든 원소마다 특정 작업을 하는 함수를 작용하고 그 함수가 반환한 결과를 모아서 배열로 반환한다

```
// map() 함수

// 배열의 값을 제곱하여 새로운 배열 만든다
const numbers = [1,2,3,4,5];
const square = numbers.map(num => num * num);
console.log(square);
```



map() 함수

map()

❖ map() 함수

- 배열의 모든 원소마다 특정 작업을 하는 함수를 작용하고 그 함수가 반환한 결과를 모아서 배열로 반환한다

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

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

// [0,0,0] friends.map() 이 최종적으로 반환한 값

```
// 이름 뒤에 하트 붙이기
friends.map(function(friend) {  // 익명함수
  return friend + " ♥";
})      // 익명함수의 friend 에 friends 배열의 원소가
          하나씩 넘어오고 그 원소에 하트를 붙여 반환
```

// map() 함수 특징:
1) map() 함수 인자로 전달한 함수는 배열 friends 의 원소를 대상으로 실행 (즉, 원소 3개이므로 함수 3번 실행)
2) 그 함수가 반환한 값이 모여 배열이 되고
그 배열이 map() 함수의 반환값이 된다

[실습] [./src/App.js] map()

Map()

App.js

```
// ./src/App.js
import React from 'react';
// import './App.css';
import ReturnMap from ‘./Components/ReturnMap'

function App() {
  return (
    <div>
      <h1>Start React!</h1>
      <p>CSS 적용하기</p>
      <ReturnMap />
    </div>
  );
}

export default App;
```

ReturnMap.js

```
// ./src/Components/ReturnMap.js
import React, { Component } from 'react';

class ReturnMap extends Component {
  render() {
    const element_Array = [
        <li>Start</li>
      , <li>react</li>
      , <li>Array map</li>
    ]
    return (
      <ul>
        {element_Array.map((array_val) => array_val)}
      </ul>
    )
  }
}

export default ReturnMap;
```

[실습] [.src/index.js][./src/App.js][./public/index.html]

App.js

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

function Food(props) {
  return <h3>I Like {props.fav}!</h3>
}

function App() {
  return (
    <div>
      <h1>Hello! </h1>
      <Food fav = "kimchi" />
    </div>
  );
}

export default App;
```

App.js 수정

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

function Food({fav}) {
  return <h3>I Like {fav}!</h3>
}

function App() {
  return (
    <div>
      <h1>Hello! </h1>
      <Food fav = "kimchi" />
    </div>
  );
}

export default App;
```

[실습] [.src/index.js][./src/App.js][./public/index.html]

map()

App.js

```
// const foodILike = [];
const foodILike = [
  {
    name: 'Kimchi',
    image: 'http://aeriskitchen.com/wp-content/uploads/2008/09/kimchi_bokkeumbap_02-.jpg'
  },
  {
    name: 'Samgyeopsal',
    image:
      'https://3.bp.blogspot.com/-
hKwIBxIVcQw/WfsewX3fhJI/AAAAAAAAALk/yHxnxFXcfx4ZKSfHS_RQNKjw3bAC03AnACLcBGAs/s400/DSC07624.jpg'
  },
  {
    name: 'Bibimbap',
    image:
      'http://cdn-image.myrecipes.com/sites/default/files/styles/4_3_horizontal_-
_1200x900/public/image/recipes/ck/12/03/bibimbop-ck-x.jpg?itok=RoXlp6Xb'
  },
 {
    name: 'Doncasu',
    image: 'https://s3-media3.fl.yelpcdn.com/bphoto/7F9eTTQ_yxaWIRytAu5feA/ls.jpg'
  },
  {
    name: 'Kimbap',
    image: 'http://cdn2.koreanbapsang.com/wp-content/uploads/2012/05/DSC_1238r-e1454170512295.jpg'
  },
];
```

react 예제 (props)
npx create-react-app react-ex01-03-props

[실습] [.src/index.js][./src/App.js][./public/index.html]

map()

App.js

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

function Food({name}) {
  return <h3>I Like {name}!</h3>
}

// 생략

function App() {
  return (
    <div>
      <h1>Food</h1>
      {foodILike.map(dish => (<Food name = {dish.name} />))}
    </div>
  );
}

export default App;
```

react 예제 (props)
npx create-react-app react-ex01-03-props

[실습] [.src/index.js][./src/App.js][./public/index.html]

map()

App.js

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

function Food({name, picture}) {
  return (
     <div>
        <h3>I Like {name}!</h3>
        <img src = {picture}  width = "50%" alt = {name} />
      </div>
  )
}
// 생략

function App() {
  return (
    <div>
      <h1>Food</h1>
      {foodILike.map(dish => (
           <Food name = {dish.name} picture = {dish.image} />
       ))}
    </div>    </div>
  );
}

export default App;
```

react 예제 (props)
npx create-react-app react-ex01-03-props

JSX

담당: 김희숙
(jasmin11@hanmail.net)

[Quiz] JSX (stu)

❖ *조리법(데이터 2건)*

맛있는 조리법

구운 연어

- 연어
- 잣
- 버터 상추
- 옐로 스쿼시(Yellow Squash, 호박의 한 종류)
- 올리브 오일
- 마늘

조리 절차

오븐을 350도로 예열한다.
유리 베이킹 그릇에 올리브 오일을 두른다.
연어, 마늘, 잣을 그릇에 담는다.
오븐에서 15분간 익힌다.
옐로 스쿼시를 추가하고 다시 30분간 오븐에서 익힌다.
오븐에서 그릇을 꺼내서 15분간 식힌다음에 상추를 곁들여서 내놓는다.

생선 타코

- 흰살생선
- 치즈
- 아이스버그 상추
- 토마토
- 또띠야

조리 절차

생선을 그릴에 익힌다.
또띠야 3장 위에 생선을 얹는다.
또띠야에 얹은 생선 위에 상추, 토마토, 치즈를 얹는다.

[실습] data 배열

조리법 배열

```
// 두 가지 조리법
const data = [
  {
    "name": "구운 연어",
    "ingredients": [
      { "name": "연어", "amount": 500, "measurement": "그램" },
      { "name": "잣", "amount": 1, "measurement": "컵" },
      { "name": "버터 상추", "amount": 2, "measurement": "컵" },
      { "name": "옐로 스쿼시(Yellow Squash, 호박의 한 종류)", "amount": 1, "measurement": "개" },
      { "name": "올리브 오일", "amount": 0.5, "measurement": "컵" },
      { "name": "마늘", "amount": 3, "measurement": "쪽" }
    ],
    "steps": [
      "오븐을 350도로 예열한다.",
      "유리 베이킹 그릇에 올리브 오일을 두른다.",
      "연어, 마늘, 잣을 그릇에 담는다.",
      "오븐에서 15분간 익힌다.",
      "옐로 스쿼시를 추가하고 다시 30분간 오븐에서 익힌다.",
      "오븐에서 그릇을 꺼내서 15분간 식힌다음에 상추를 곁들여서 내놓는다."
    ]
  },
```

react 예제 (props)
npx create-react-app recipe-app

[실습] data 배열

조리법 배열

```
// 두 가지 조리법

  {
    "name": "생선 타코",
    "ingredients": [
      { "name": "흰살생선", "amount": 500, "measurement": "그램" },
      { "name": "치즈", "amount": 1, "measurement": "컵" },
      { "name": "아이스버그 상추", "amount": 2, "measurement": "컵" },
      { "name": "토마토", "amount": 2, "measurement": "개(큰것)"},
      { "name": "또띠야", "amount": 3, "measurement": "개" }
    ],
    "steps": [
      "생선을 그릴에 익힌다.",
      "또띠야 3장 위에 생선을 얹는다.",
      "또띠야에 얹은 생선 위에 상추, 토마토, 치즈를 얹는다."
    ]
  }
]
```

react 예제 (props)
npx create-react-app recipe-app

[Quiz] JSX (stu)

❖ 조리법을 JSX로 작성하기: 빈칸을 완성하시오

```
// [Quiz] App 컴포넌트

function App() {
  return (
    <div className="App">
      <Menu recipes={_____________}></Menu>
    </div>
  );
}

export default App;
```

```
// [Quiz] 다음 <조건>을 적용하여 실습하시오.

// <조건>
// 1. 다음 컴포넌트를 저장하기 위한 프로그램을 작성한다
// Menu 컴포넌트: 조리법 목록
// Recipe 컴포넌트: 각 조리법의 UI

// 2. App.js 를 완성
// (DOM에 렌더링하는 대상은 Menu 컴포넌트)
// (데이터를 Menu 컴포넌트의 recipes 프로퍼티로 넘긴다)
```

❖ 조리법 배열(요리이름, 재료, 조리절차)

```
const data = [
 {  name,
    ingredients:
            [ name, amount, measurement],
    steps
 },
 ...
]
```

[Quiz] JSX (stu)

❖ 조리법을 JSX로 작성하기: 빈칸을 완성하시오

```
// [Quiz] Menu 컴포넌트

// recipe-app/src/Components/Menu.js
import React from 'react';
import Recipe from './Recipe';
import '../../stylesheets/Menu.css';

const Menu = (                ) => {
    return (
            <article>
                <header>
                    <h1>맛있는 조리법</h1>
                </header>
                <div className="recipes">
                    {
                            <Recipe                    />)
                    }
                </div>          </article>
    )
};

export default Menu
```

```
// [Quiz] 다음 <조건>을 적용하여 실습하시오.

// <조건>
// 1. 다음 컴포넌트를 저장하기 위한 프로그램을 작성한다
// Menu 컴포넌트: 조리법 목록
// Recipe 컴포넌트: 각 조리법의 UI

// 2. App.js 를 완성
// (DOM에 렌더링하는 대상은 Menu 컴포넌트)
// (데이터를 Menu 컴포넌트의 recipes 프로퍼티로 넘긴다)
```

❖ 조리법 배열(요리이름, 재료, 조리절차)

```
const data = [
 {  name,
    ingredients:
            [ name, amount, measurement],
    steps
 },
 ...
]
```

[Quiz] JSX (stu)

❖ 조리법을 JSX로 작성하기: 빈칸을 완성하시오

```
// [Quiz] Menu 컴포넌트
// recipe-app/src/Components/Menu.js
import React from 'react';
import Recipe from './Recipe';
import '.././stylesheets/Menu.css';

const Menu = (                ) => {
    return (
 <article>
              <header>
                 <h1>{props.title}</h1>
              </header>
              <div className="recipes">
                 {
                          <Recipe                          />)
                 }
              </div>
           </article>    )
};
export default Menu
```

```
// [Quiz] 다음 <조건>을 적용하여 실습하시오.

// <조건>
// 1. 다음 컴포넌트를 저장하기 위한 프로그램을 작성한다
// Menu 컴포넌트: 조리법 목록
// Recipe 컴포넌트: 각 조리법의 UI

// 2. App.js 를 완성
// (DOM에 렌더링하는 대상은 Menu 컴포넌트)
// (데이터를 Menu 컴포넌트의 recipes 프로퍼티로 넘긴다)
```

❖ 조리법 배열(요리이름, 재료, 조리절차)

```
const data = [
 {  name,
    ingredients:
          [ name, amount, measurement],
    steps
 },
 ...
]
```

[Quiz] JSX (stu)

❖ 조리법을 JSX로 작성하기: 빈칸을 완성하시오

```
// [Quiz] Recipe 컴포넌트

// recipe-app/src/Components/Recipe.js
const Recipe = (_____) =>
    <section id={name.toLowerCase().replace(/ /g, '-')}>
        <h1>{name}</h1>
        <ul className="ingredients">
            {ingredients.map((ingredient, i) =>
                <li key={i}>{ingredient.name}</li>
            )}
        </ul>
        <section className="instructions">
            <h2>조리 절차</h2>
            {steps.map((step, i) =>
                <p key={i}>{step}</p>
            )}
        </section>
    </section>
export default Recipe
```

```
// [Quiz] 다음 <조건>을 적용하여 실습하시오.

// <조건>
// 1. 다음 컴포넌트를 저장하기 위한 프로그램을 작성한다
// Menu 컴포넌트: 조리법 목록
// Recipe 컴포넌트: 각 조리법의 UI

// 2. App.js 를 완성
// (DOM에 렌더링하는 대상은 Menu 컴포넌트)
// (데이터를 Menu 컴포넌트의 recipes 프로퍼티로 넘긴다)
```

❖ 조리법 배열(요리이름, 재료, 조리절차)

```
const data = [
 {  name,
    ingredients:
            [ name, amount, measurement],
    steps
 },
 ...
]
```

[Quiz] JSX (ans)

❖ 조리법을 JSX로 작성하기: 빈칸을 완성하시오

```
// [Quiz] Recipe 컴포넌트

// recipe-app/src/Components/Recipe.js
const Recipe = (_____) =>
    <section id={name.toLowerCase().replace(/ /g, '-')}>
        <h1>{name}</h1>
        <IngredientsList list={ingredients} />
        <Instructions title="조리 절차"
                      steps={steps} />

export default Recipe
```

```
// [Quiz] 다음 <조건>을 적용하여 실습하시오.

// <조건>
// 3. Recipe 컴포넌트를 분리한다
// Instructions      컴포넌트

// 4. Instructions 컴포넌트 완성
```

❖ 조리법 배열(요리이름, 재료, 조리절차)

```
const data = [
 {  name,
    ingredients:
            [ name, amount, measurement],
    steps
 },
 ...
]
```

[Quiz] JSX (stu)

❖ 조리법을 JSX로 작성하기: 빈칸을 완성하시오

```
// [Quiz] Recipe 컴포넌트

// recipe-app/src/Components/Recipe.js
const Recipe = (_____) =>
    <section id={name.toLowerCase().replace(/ /g, '-')}>
        <h1>{name}</h1>
        <ul className="ingredients">
            {ingredients.map((ingredient, i) =>
                <li key={i}>{ingredient.name}</li>
            )}
        </ul>
        <section className="instructions">
            <h2>조리 절차</h2>
            {steps.map((step, i) =>
                <p key={i}>{step}</p>
            )}
        </section>
    </section>
export default Recipe
```

```
// [Quiz] 다음 <조건>을 적용하여 실습하시오.

// <조건>
// 1. 다음 컴포넌트를 저장하기 위한 프로그램을 작성한다
// Menu 컴포넌트: 조리법 목록
// Recipe 컴포넌트: 각 조리법의 UI

// 2. App.js 를 완성
// (DOM에 렌더링하는 대상은 Menu 컴포넌트)
// (데이터를 Menu 컴포넌트의 recipes 프로퍼티로 넘긴다)
```

❖ 조리법 배열(요리이름, 재료, 조리절차)

```
const data = [
 {  name,
    ingredients:
            [ name, amount, measurement],
    steps
 },
 ...
]
```

[Quiz] JSX (stu)

❖ 조리법을 JSX로 작성하기: 빈칸을 완성하시오

```
// [Quiz]  컴포넌트
// recipe-app/src/Components/Recipe.js
import React from 'react';
import Instructions from "./Instructions";

const Recipe = ( {                                  } ) =>
    <section id={name.toLowerCase().replace(/ /g, '-')}>
        <h1>{name}</h1>
        <ul className="ingredients">
            {ingredients.map((ingredient, i) =>
                <li key={i}>{ingredient.name}</li>
            )}
        </ul>
        <Instructions title="조리 절차" steps={steps} />
    </section>


export default Recipe;
```

```
// [Quiz] 다음 <조건>을 적용하여 실습하시오.

// <조건>
// 3. Recipe 컴포넌트를 분리한다
// Instructions      컴포넌트

// 4. Instructions 컴포넌트 완성
```

❖ 조리법 배열(요리이름, 재료, 조리절차)

```
const data = [
 {  name,
    ingredients:
            [ name, amount, measurement],
    steps
 },
 ...
]
```

[Quiz] JSX (stu)

❖ 조리법을 JSX로 작성하기: 빈칸을 완성하시오

```
// [Quiz] Instructions 컴포넌트

// recipe-app/src/Components/Instructions.js

import React from 'react';

const Instructions = ( {                }) => {
    return (
        <section className="instructions">
            <h2>{title}</h2>
            {
                steps.map((step, i) =>
                    <p key={i}>{step}</p>)
            }
        </section>
    )
}

export default Instructions;
```

```
// [Quiz] 다음 <조건>을 적용하여 실습하시오.

// <조건>
// 3. Recipe 컴포넌트를 분리한다
// Instructions      컴포넌트

// 4. Instructions 컴포넌트 완성
```

❖ 조리법 배열(요리이름, 재료, 조리절차)

```
const data = [
 {  name,
    ingredients:
            [ name, amount, measurement],
    steps
 },
 ...
]
```