4. 리액트의 작동 원리

Prof. Seunghyun Park (sp@hansung.ac.kr)

Division of Computer Engineering

한성대학교 HANSUNG UNIVERSITY

학습 목표: 4장. 리액트의 작동 원리

- React element 생성
  - `React.createElement( type, props, children )`
- ReactDOM 렌더링
  - `ReactDOM.render( element, container )` element를 렌더링
- React component
  - 함수형 컴포넌트
  - 클래스 컴포넌트

페이지 설정

- React: 뷰를 만들기 위한 라이브러리

https://unpkg.com/react@16.14.0/umd/react.development.js
https://unpkg.com/react-dom@16.14.0/umd/react-dom.development.js

- ReactDOM: UI를 브라우저에 렌더링 할 때 사용하는 라이브러리

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>순수 리액트 예제</title>
</head>
<body>

<div id="react-container"></div>

<script src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
<script>

/* ch04/01/01-page-setup.html */
// 순수 리액트와 자바스크립트 코드

</script>

</body>
</html>
```

타겟 컨테이너

React와 ReactDOM 라이브러리

React element 생성과 ReactDOM 렌더링

https://ko.reactjs.org/docs/react-api.html#createelement
https://ko.reactjs.org/docs/react-dom.html#render

```
<!-- Target Container -->
<div id="react-container"></div>
```

<script> 자바스크립트 코드

```
/* ch04/02/01-elements.html */
const dish = React.createElement(
  "h1", { id: "recipe-0" }, "구운 연어"
)

ReactDOM.render(
    dish,
    document.getElementById('react-container')
)
```

</script>

```
console.log('dish', dish)
```

element 생성
- type: h1
- property: id="recipe-0"
- 자식노드: 텍스트 ("구운 연어")

ReactDOM 렌더링
- element (dish: h1)
- 대상: 'react-container'

[ ]: 필수X

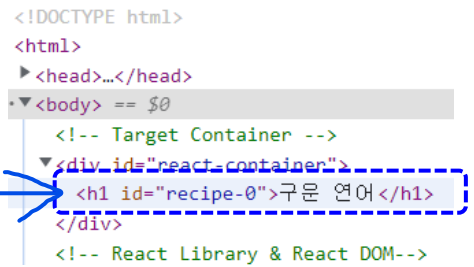
```
React.createElement( 리액트 element를 생성 후 반환
  type, [props], [...children]
) 타입, 프로퍼티, 자식요소
```

- 인자로 주어지는 타입에 따라 새로운 **리액트 엘리먼트를 생성하여 반환**

```
ReactDOM.render(
  element, container[, ...callback]
) 뿌릴 엘리먼트, 뿌릴 위치,
```

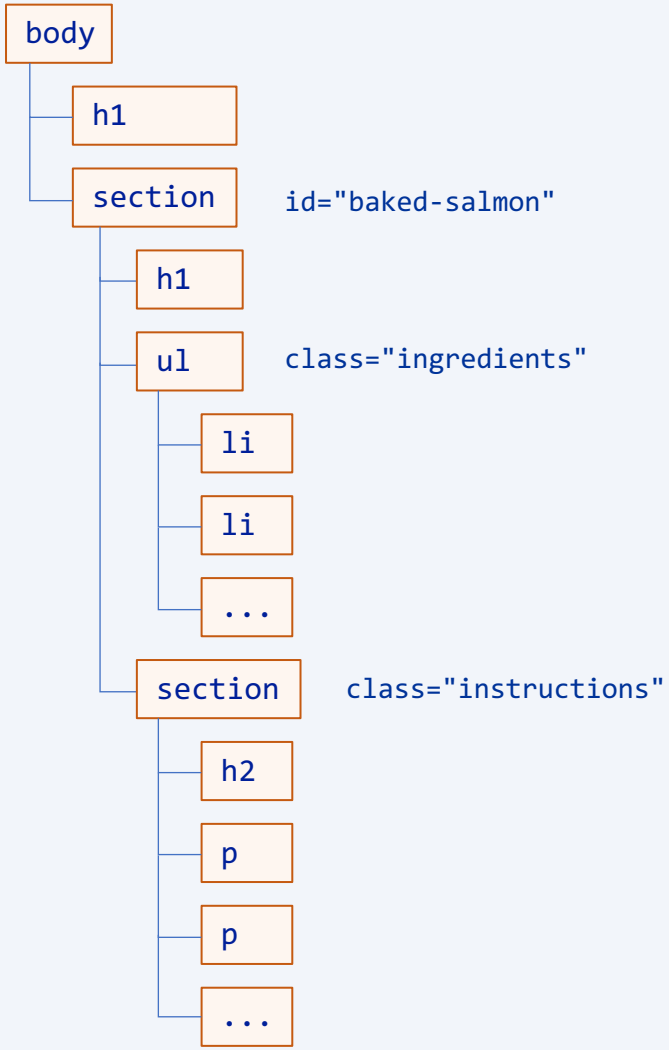
- 인자로 주어지는 렌더링 할 **리액트 엘리먼트**를 제공된 컨테이너의 **DOM** (렌더링이 일어날 대상 DOM)**에 렌더링**,
- 구성요소에 대한 참조를 반환

구운 연어

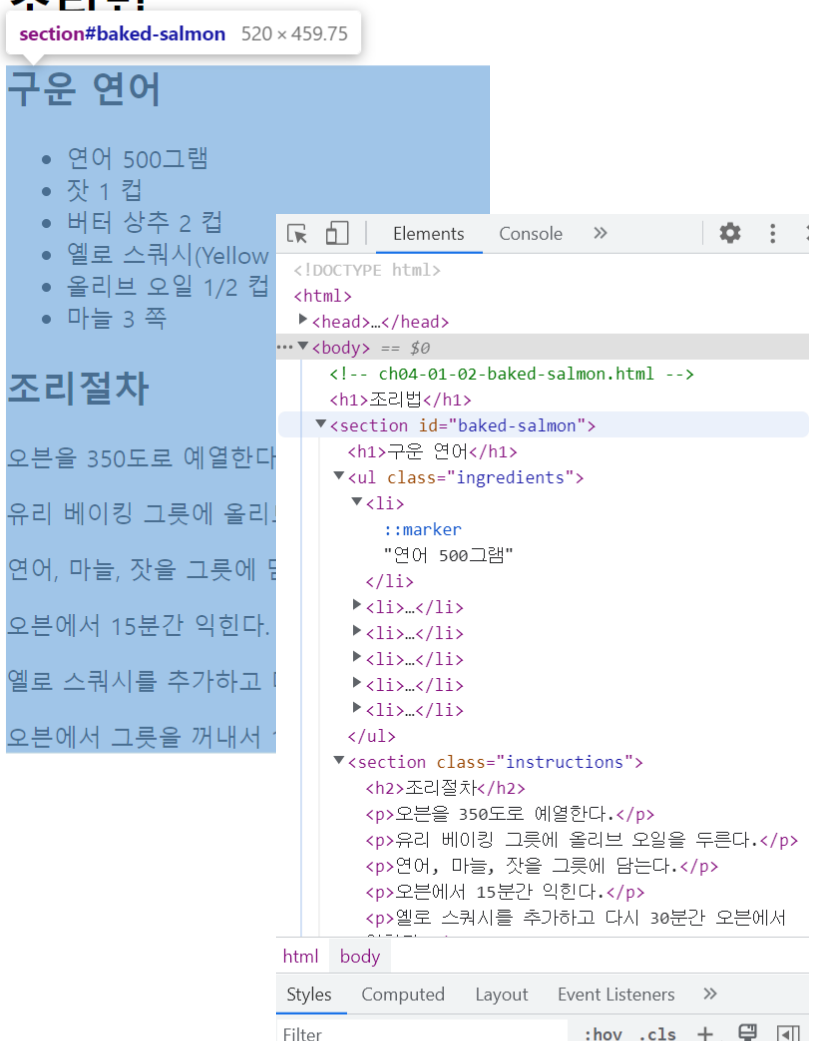



```
dish Object
  $$typeof: Symbol(react.element)
  key: null
  props: {id:'recipe-0', children: '구운 연어'}
  ref: null
  type: "h1"
  _owner: null
  _store: {validated: false}
  _self: null
  _source: null
  [[Prototype]]: Object
```

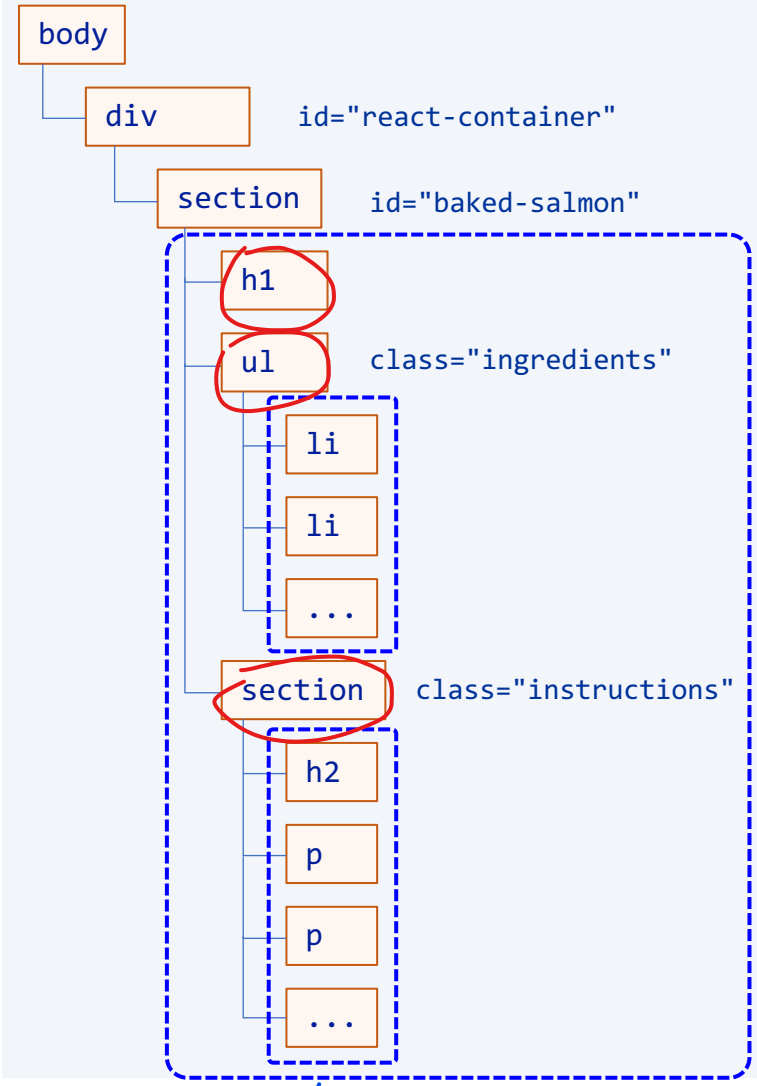
예제 1-2. baked-salmon (html)




```html
<!-- ch04/01/02-baked-salmon.html -->
<h1>조리법</h1>

<section id="baked-salmon">
  <h1>구운 연어</h1>
  <ul class="ingredients">
    <li>연어 500그램</li>
    <li>잣 1 컵</li>
    <li>...</li>
    <li>...</li>
    <li>...</li>
    <li>...</li>
  </ul>
  <section class="instructions">
    <h2>조리절차</h2>
    <p>오븐을 350도로 예열한다.</p>
    <p>...</p>
    <p>...</p>
    <p>...</p>
    <p>...</p>
    <p>...</p>
  </section>
</section>
```



예제 1-2. baked-salmon (react)



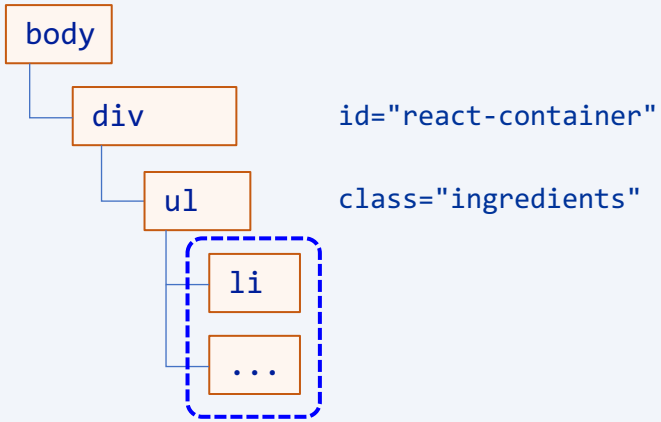

```
<!-- Target Container -->
<div id="react-container"></div>

/* ch04/02/03-elements.html */
const dish = React.createElement(
  "section", {id: "baked-salmon"},
    React.createElement("h1", null, "구운 연어"),
    React.createElement(
      "ul", {"className": "ingredients"},
        React.createElement("li", null, "연어 500그램"),
        React.createElement("li", null, "잣 1 컵"),
        React.createElement("li", null, "..."),
        React.createElement("li", null, "...")
      ),
    React.createElement(
      "section", {"className": "instructions"},
        React.createElement("h2", null, "조리절차"),
        React.createElement("p", null, "오븐을..."),
        React.createElement("p", null, "유리..."),
        React.createElement("p", null, "..."),
        React.createElement("p", null, "...")
    )
)

ReactDOM.render(dish, document.getElementById('react-container'))
console.log('dish element', dish)
```

section은 자식 엘리먼트가 3개: h1, ul, section

ul은 자식 엘리먼트가 6개: li, …

section은 자식 엘리먼트가 7개: h2, p, …

예제 1-2. baked-salmon (react: *props.children*, 계속)



bad

```
/* ch04/02/04-1-elements.html */
React.createElement(
  "ul", { className: "ingredients" },
    React.createElement("li", null, "연어"),
    React.createElement("li", null, "잣"),
    ...
);
```

props.children 을 array로 생성

```
/* ch04/02/04-elements.html */
const items = [
  "연어 500그램",
  "잣 1 컵",
  "버터 상추 2 컵",
  "옐로 스쿼시(Yellow Squash, 호박의 한 종류) 1개",
  "올리브 오일 1/2 컵",
  "마늘 3 쪽"
]
```

map : 배열 내 모든 요소에 대해 함수의 결과를 모아 새로운 배열로 반환

good

```
const ingredients = React.createElement(
  "ul", { className: "ingredients" },
    items.map( (ingredient) =>
      React.createElement("li", null, ingredient)
    )
)
```

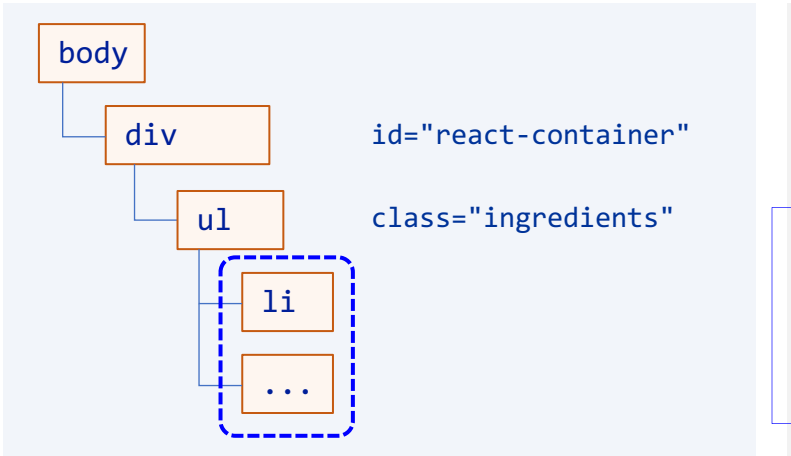
반복 작업 단순화를 위해 array.map() 활용

items 배열의 모든 요소를 활용해 React.createElement() 호출

```
ReactDOM.render(ingredients, document.getElementById('react-container'))
console.log('ingredients', ingredients)
```

그러나 오류가 뜬다

예제 1-2. baked-salmon (react: *props.children*)




```
<!-- Target Container -->
<div id="react-container">
  <ul class="ingredients"> == $0
    <li>
      ::marker
      "연어 500그램"
    </li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
    <li>…</li>
  </ul>
</div>
<!-- React Library &
```

- 연어 500그램
- 잣 1 컵
- 버터 상추 2 컵
- 옐로 스쿼시(Yellow Squash, 호박의 한 종류) 1개
- 올리브 오일 1/2 컵
- 마늘 3 쪽

```
/* ch04/02/04-elements.html */
const items = [ ... ]
const ingredients = React.createElement( "ul", { className: "ingredients" },
  items.map( (ingredient, i) => React.createElement("li", null, ingredient) ) )

ReactDOM.render(ingredients, document.getElementById('react-container'))
console.log('ingredients', ingredients);
```

props.children

```
...
props:{className: 'ingredients', children: Array(6)}
 children:(6) [{…}, {…}, {…}, {…}, {…}, {…}]
  0:{$$typeof: Symbol(react.element), type: 'li', key: null, ref: null, props: {…}, …}
  1:{$$typeof: Symbol(react.element), type: 'li', key: null, ref: null, props: {…}, …}
  ...
 length:6
[[Prototype]]:Array(0)
```

```
Warning: Each child in a list should have a unique "key" prop.
Check the top-level render call using <ul>. See
https://fb.me/react-warning-keys for more information.
  in li
```

key 값을 넣어주러 가자

예제 1-2. baked-salmon (react: *props.children*)

```
/* ch04/02/05-elements.html */
const items = [ ... ]
const ingredients = React.createElement( "ul", { className: "ingredients" },
   items.map( (ingredient, i) => React.createElement("li", { key: i }, ingredient) )
)
ReactDOM.render(ingredients, document.getElementById('react-container'))
console.log('ingredients', ingredients)
```

```
...
props:{className: 'ingredients', children: Array(6)}
 children:(6) [{…}, {…}, {…}, {…}, {…}, {…}]
  0:{$$typeof: Symbol(react.element), type: 'li', key: '0', ref: null, props: {…}, …}
  1:{$$typeof: Symbol(react.element), type: 'li', key: '1', ref: null, props: {…}, …}
  ...
 length:6
[[Prototype]]:Array(0)
```

```
/* ch04-02-04-elements.html */
const ingredients = React.createElement( "ul", { className: "ingredients" },
   items.map( ingredient => React.createElement("li", null, ingredient) ) )
```

리액트 엘리먼트와 컴포넌트

- 리액트 엘리먼트
  - 리액트로 만들어진 앱을 구성하는 최소한의 단위
  - 화면에 표시할 내용을 기술하는 일반 객체 (plain object)

```
const element = React.createElement(
    "h1",
    null,
    "Hello, world"
  );
```

JSX

```
const element = <h1>Hello, world</h1>;
```

element

- 리액트 컴포넌트
  - UI를 재사용 가능한 개별적인 여러 조각으로 구성
  - 데이터를 입력 받고,
    view의 상태에 따라 DOM 노드를 출력하는 함수 또는 클래스

element들을 트리구조로 묶어서 재사용할 수 있게 만든

```
function IngredientsList(props){
  return <h1>Hello, {props.name}</h1>;
}
```

함수 컴포넌트

element

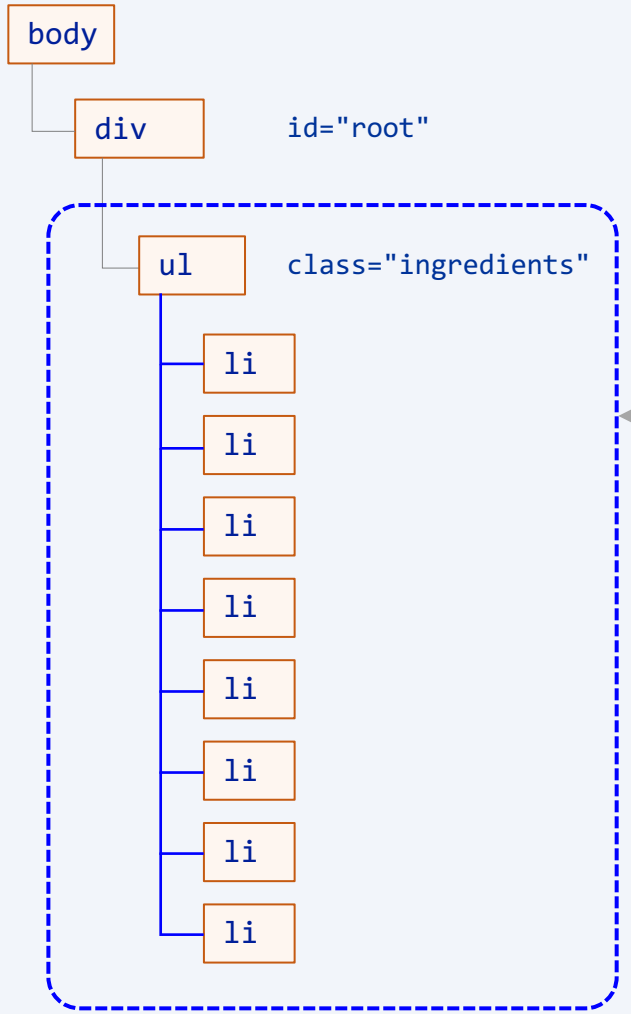
```
class IngredientsList extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

클래스 컴포넌트

리액트 컴포넌트 (함수)

리액트 엘리먼트는 변경 불가능한 객체다
리액트 컴포넌트는 변경가능!

함수이 element 넣으면 = 컴포넌트




```
<!-- Root element -->
<div id="root"></div>

/* ch04/03/01-1-functions.html */
function IngredientsList() {
  return React.createElement(
    "ul",
    { className: "ingredients" },
    React.createElement("li", null, "무염 버터 1 컵"),
    React.createElement("li", null, "크런치 땅콩 버터 1 컵"),
    React.createElement("li", null, "흑설탕 1 컵"),
    React.createElement("li", null, "백설탕 1 컵"),
    React.createElement("li", null, "달걀 2 개"),
    React.createElement("li", null, "일반 밀가루 2.5 컵"),
    React.createElement("li", null, "베이킹 소타 1 티스푼"),
    React.createElement("li", null, "소금 0.5 티스푼")
  );
}

ReactDOM.render(
    React.createElement(IngredientsList, null, null),
    document.getElementById("root")
);
```

함수 컴포넌트

리액트 컴포넌트 (함수)

```
/* ch04/03/01-2-functions.html */
const secIngredients = [
    "무염 버터 1 컵",          "크런치 땅콩 버터 1 컵",
    "흑설탕 1 컵",             "백설탕 1 컵",
    "달걀 2 개",               "일반 밀가루 2.5 컵",
    "베이킹 소타 1 티스푼",    "소금 0.5 티스푼"
];

function IngredientsList() {
  return React.createElement(
    "ul",
    { className: "ingredients" },
    secIngredients.map( (ingrd, i) =>
        React.createElement("li", {key: i}, ingrd) )
  );
}

ReactDOM.render(
  React.createElement(IngredientsList, null, null),
  document.getElementById("root")
);
```

element

props

children

Array.map():
배열의 모든 요소에 대해 *callback* 함수를 적용한 후, 새로운 배열을 반환

callback 함수:
secIngredients 배열의 요소를 이용하여 ul의 *children* 요소인 li 엘리먼트를 생성

```
/* ch04/03/01-3-functions.html */
const secIngredients = [ ... ];

function IngredientsList(props) {
  return React.createElement(
    "ul",
    { className: "ingredients" },
    props.items.map( (ingrd, i) =>
        React.createElement("li", {key: i}, ingrd ) )
  );
}

ReactDOM.render(
  React.createElement(IngredientsList,
    {items: secIngredients}, null),
  document.getElementById("root")
);
```

매개변수 추가 가능

데이터 *props*를 전달 받고, React 엘리먼트를 반환하는 함수 컴포넌트

component

props

리액트 컴포넌트 (함수)

```
/* ch04/03/01-4-functions.html */
const secIngredients = [ ... ];

function IngredientsList( { items } ) {
  return React.createElement(
    "ul",
    { className: "ingredients" },
    items.map( (ingrd, i) =>
      React.createElement("li", {key: i}, ingrd) )
  );
}

ReactDOM.render(
  React.createElement(IngredientsList,
    {items: secIngredients}, null),
  document.getElementById("root")
);
```

객체의 구조분해 할당
> *props* 대신 {items}로 Items를 변수로 활용

{ 속성
items : secIngredients
(배열을 객체에 담은 것)
}

```
/* ch04/03/01-5-functions.html */

const IngredientsList = ({ items }) =>
  React.createElement(
    "ul",
    { className: "ingredients" },
    items.map( (ingrd, i) =>
      React.createElement("li", {key: i}, ingrd) )
  )
```

화살표 함수, 동일한 표현 가능

함수 정의가 1문장:
{ }, return 생략

리액트 컴포넌트 (클래스)

사실 함수형 컴포넌트를 더 많이 쓴다

```
<!-- Target Container -->
<div id="root"></div>

/* ch04/03/02-1-components.html */
class IngredientsList extends React.Component {
  render() {
    return React.createElement(
      "ul",
      { className: "ingredients" },
      React.createElement("li", null, "연어 500그램"),
      React.createElement("li", null, "잣 1 컵"),
      React.createElement("li", null, "버터 상추 2 컵"),
      React.createElement("li", null, "옐로 스쿼시(Yellow... 1개"),
      React.createElement("li", null, "올리브 오일 1/2 컵"),
      React.createElement("li", null, "마늘 3 쪽")
    )
  }
}

ReactDOM.render(
  React.createElement(IngredientsList, null, null),
  document.getElementById('root')
)
```

클래스 컴포넌트

필수

리액트 컴포넌트 (클래스)

```
/* ch04/03/02-2-components.html */
const items = [...];

class IngredientsList extends React.Component {

  render() {
    return React.createElement(
      "ul",
      { className: "ingredients" },
      this.props.items.map( (ingrd, i) =>
        React.createElement("li", {key: i}, ingrd) )
    )
  }

}

ReactDOM.render(
  React.createElement(IngredientsList, {items}, null),
  document.getElementById('react-container')
)
```

```
/* ch04/03/02-3-components.html */

class IngredientsList extends React.Component {

  render() {
    return React.createElement(
      "ul",
      { className: "ingredients" },
      this.props.items.map(this.renderListItem)
    )
  }

  renderListItem(ingrd, i) {
    return React.createElement("li", { key: i }, ingrd)
  }

}

ReactDOM.render(
  React.createElement(IngredientsList, {items}, null),
  document.getElementById('react-container')
)
```

*callback*을 클래스의 메서드로 구현

학습 정리: 4장. 리액트의 작동 원리

- React element 생성
- ReactDOM 렌더링
- React component
  - 함수형 컴포넌트
  - 클래스 컴포넌트