5. JSX를 사용하는 리액트 (2)

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

Division of Computer Engineering

한성대학교 HANSUNG UNIVERSITY

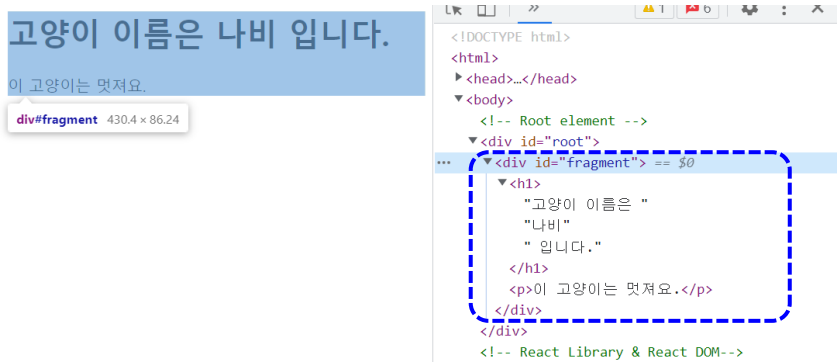
학습 목표: 5장. JSX를 사용하는 리액트

- JSX
- Babel
- JSX 활용 예제: Recipe
- React fragments
- 웹팩
- 프로젝트 구성하기

React Fragments

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

/* ch05-06-1-fragments.html */

const Cat = function( {name} ){
  return <h1>고양이 이름은 {name} 입니다.</h1>;
}

// Menu를 현재의 DOM 안에 렌더링
ReactDOM.render(
  <Cat name="나비" />,
  document.getElementById('root')
)
```

컴포넌트의 최상위 노드는 하나




```
/* ch05-06-2-fragments.html */
const Cat = function( {name} ){
  return (
    <h1>고양이 이름은 {name} 입니다.</h1>
    <p>이 고양이는 멋져요.</p>
  );
}
```

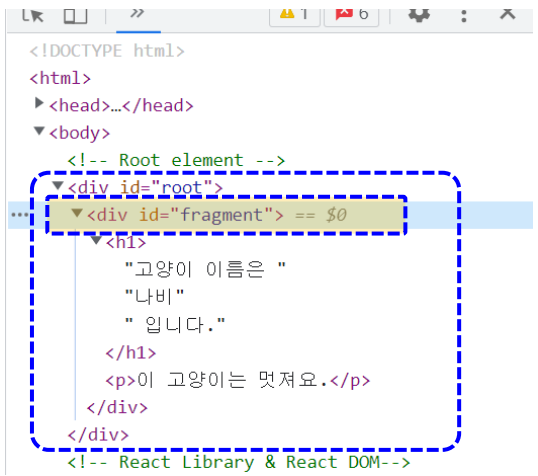
컴포넌트의 최상위 노드를 둘 이상 지정하려고 하면…

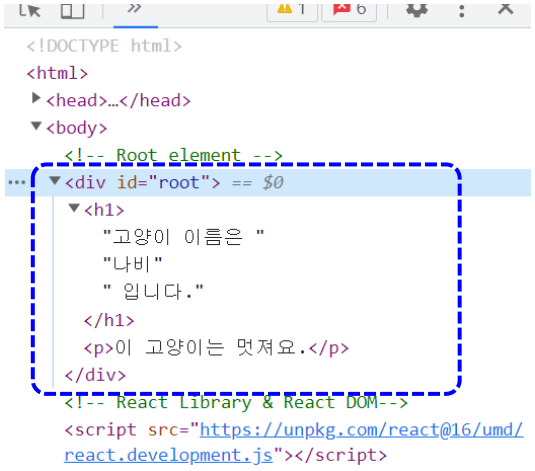
```
Uncaught SyntaxError: Inline Babel script:
Adjacent JSX elements must be wrapped in an enclosing tag
```

```
/* ch05-06-3-fragments.html */
const Cat = function( {name} ){
  return (
    <div id="fragment">
      <h1>고양이 이름은 {name} 입니다.</h1>
      <p>이 고양이는 멋져요.</p>
    </div>
  );
}
```

React Fragments

```
/* ch05-06-3-fragments.html */
const Cat = function( {name} ){
  return (
    <div id="fragment">
      <h1>고양이 이름은 {name} 입니다.</h1>
      <p>이 고양이는 멋져요.</p>
    </div>
  );
}
```






```
/* ch05-06-4-fragments.html */
const Cat = function( {name} ){
  return (
    <React.Fragment >
      <h1>고양이 이름은 {name} 입니다.</h1>
      <p>이 고양이는 멋져요.</p>
    </React.Fragment>
  );
}
```

```
/* ch05-06-5-fragments.html */
const Cat = function( {name} ){
  return (
    <>
      <h1>고양이 이름은 {name} 입니다.</h1>
      <p>이 고양이는 멋져요.</p>
    </>
  );
}
```

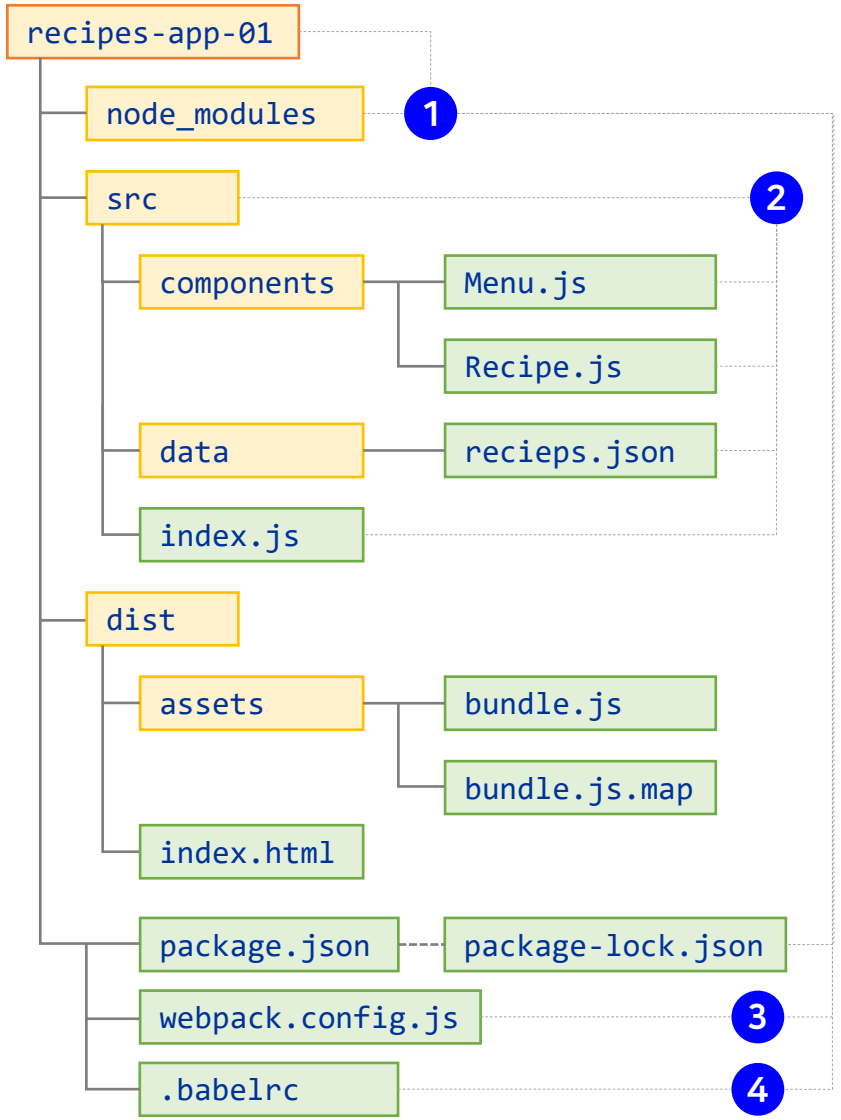
Webpack

- 웹팩
  - 모듈 번들러: 여러 다른 파일을 하나의 파일로 묶어 (dependency graph를 연결하여) 번들 생성

- 모듈화의 장점
  - 재사용성, 관리의 용이성
  - 성능: 의존 관계가 있는 여러 파일 번들로 묶으면, 브라우저가 한 번만 읽기 때문에 네트워크 성능에 유리

프로젝트 구성

reference code: ch05-05-2-recipe.html



1

```
> mkdir recipes-app-01; cd recipes-app-01
recipes-app-01> npm init -y
recipes-app-01> npm install react react-dom serve
```

프로젝트 생성 및 초기화
package.json 생성
필수 패키지 설치

https://www.npmjs.com/package/serve

2

```
src> mkdir src; cd src
src> mkdir components
src> mkdir data
src> New-Item -Path ./components/Menu.js -ItemType File
src> New-Item -Path ./components/Recipe.js -ItemType File
src> New-Item -Path ./data/recipes.json -ItemType File
src> New-Item -Path ./index.js -ItemType File
```

Menu 컴포넌트,
Recipe 컴포넌트,
데이터,
entry point 생성

3

```
src> cd ..
> npm install -D webpack webpack-cli
> New-Item -Path ./webpack.config.js -ItemType File
> npm install -D babel-loader @babel/core
> vi ./webpack.config.js
```

개발 환경에 필요한 패키지 설치
> 웹팩 설정파일: entry, output 지정

웹팩 번들링에 필요한 패키지 설치
> 웹팩 설정파일: module 추가

4

```
> npm install -D @babel/preset/env @babel/preset-react
> New-Item -Path .babelrc -ItemType File
```

바벨 실행에 사용할 preset 지정

프로젝트 생성 및 필수 패키지 설치

```
> npm init -y
> cat package.json
```

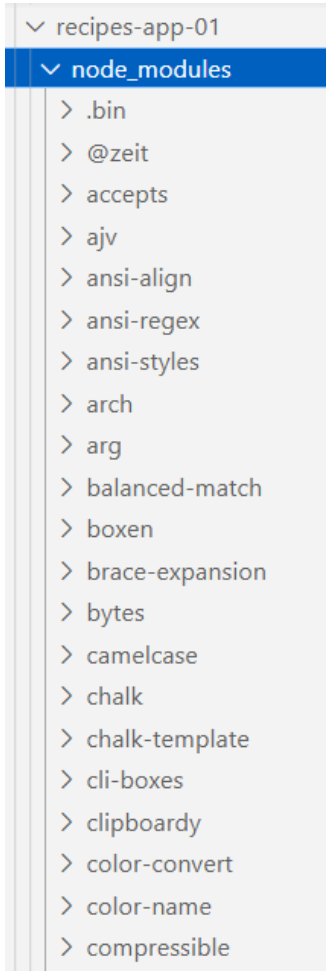
```
{
  "name": "recipes-app-01",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "build": "webpack --mode production"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

ch05/recipes-app-01/package.json

```
> npm install react react-dom serve
> cat package.json
```

```
{
  "name": "recipes-app-01",
  "version": "1.0.0",
  "description": "",
  ...
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "serve": "^14.0.1"
  }
}
```

ch05/recipes-app-01/package.json



컴포넌트 생성 (Menu, Recipe)

```
/* ch05/recipes-app-01/src/components/Menu.js */

import React from "react";
import Recipe from "./Recipe";

export default function Menu(props){
  return (
    <section>
      <header>
      <h1>{props.title}</h1>
      </header>
      <div className="recipes">
      {
        props.recipes.map( (recipe, i) => (
          <Recipe key={i} {...recipe} />
        ))
      }
      </div>
    </section>
  );
}
```

외부에서 활용할 수 있도록 모듈로 배포

```
/* ch05/recipes-app-01/src/components/Recipe.js */

import React from "react";

export default function Recipe({name, ingredients, steps}){
  return (
    <section id={name.toLowerCase().replace(/ /g, "-")}>
      <h1>{name}</h1>
      <ul className="ingredients">
        {ingredients.map( (ingrd, i) => (
          <li key={i}>{ingrd.name}</li>) )}
      </ul>
      <section>
        <h3>recipe</h3>
        <ul>
          {steps.map( (step, i) => (<p key={i}>{step}</p>) )}
        </ul>
      </section>
    </section>
  );
}
```

entry point (index.js), 데이터 생성 (recipes.json)

ch05/recipes-app-01/src/data/recipes.json

```
[
  {
    "name": "Baked Salmon",
    "ingredients": [
      { "name": "연어", "amount": 500, "measurement": "그램" },
      ...,
    ],
    "steps": [
      "오븐을 180도로 예열한다.",
      ...,
    ]
  },
  {
    "name": "Fish Tacos",
    "ingredients": [
      { "name": "흰살생선", "amount": 500, "measurement": "그램" },
      ...,
    ],
    "steps": [
      "생선을 그릴에 익힌다.",
      ...,
    ]
  }
]
```

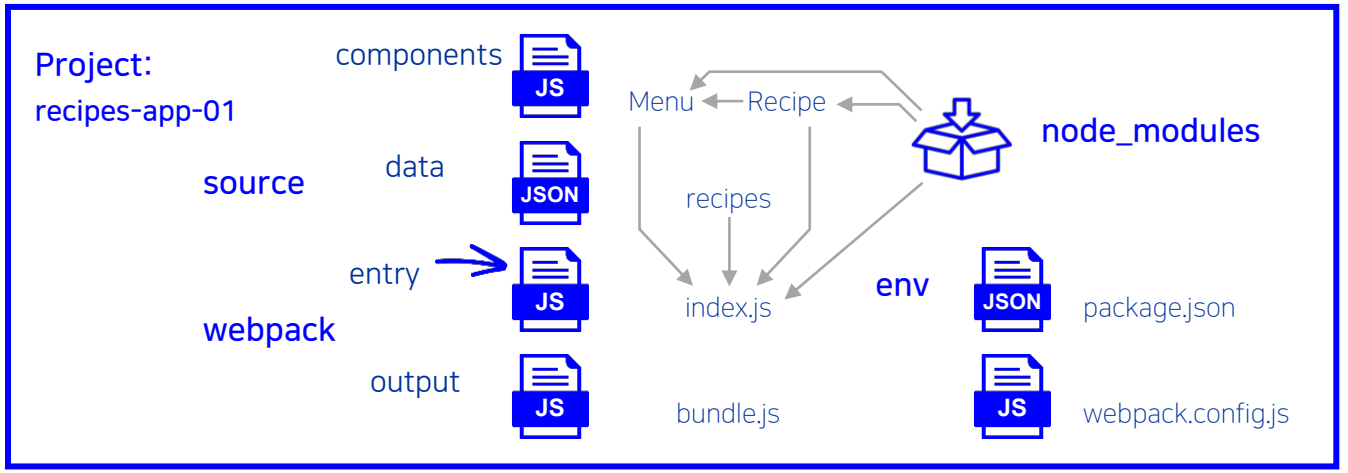
```
/* ch05/recipes-app-01/src/index.js */

import React from "react";
import {render} from "react-dom";
import Menu from "./components/Menu";
import data from "./data/recipes.json";

render(
    <Menu recipes={data} />,
    document.getElementById("root")
);
```

개발환경 구성 (webpack)

```
> npm install -D webpack webpack-cli
> cat ./package.json
> vi ./webpack.config.js
```

```
{
  ...
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "serve": "^14.0.1"
  },
  "devDependencies": {
    "webpack": "^5.74.0",
    "webpack-cli": "^4.10.0"
  }
}
```

ch05/recipes-app-01/package.json




```
/* ch05/recipes-app-01/webpack.config.js */

const path = require("path");

module.exports = {
  entry: "./src/index.js",
  output: {
    path: path.join(__dirname, "dist", "assets"),
    filename: "bundle.js",
  }
}
```

개발 환경에 필요한 패키지 (웹팩) 설치

웹팩 설정파일 webpack.config.js에 entry와 output 지정

- entry point: ./src/index.js
- output: ./dist/assets/bundle.js

개발환경 구성 (babel)

```
> npm install -D @babel-loader @babel/core
> cat ./package.json
> vi ./webpack.config.js
```

```
{
  ...,
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "serve": "^14.0.1"
  },
  "devDependencies": {
    "@babel/core": "^7.19.1",
    "babel-loader": "^8.2.5",
    "webpack": "^5.74.0",
    "webpack-cli": "^4.10.0"
  }
}
```

ch05/recipes-app-01/package.json

```
/* ch05/recipes-app-01/webpack.config.js */

const path = require("path");

module.exports = {
    entry: "./src/index.js",
    output: {
        path: path.join(__dirname, "dist", "assets"),
        filename: "bundle.js",
    },
    module: {
        rules: [{
            test: /\.js$/,
            exclude: /node_modules/,
            loader: "babel-loader"
        }]
    }
}
```

웹팩에 사용할 loader를 지정 (예: babel-loader 사용)

loader가 적용할 대상을 찾기 위한 정규 표현식
(단, ./node_module/ 하위의 .js 파일은 제외)

babel을 실행할 때 사용할 환경 필요
→ .babelrc: presets

개발환경 구성 (babel: presets) 및 번들 생성

```
> npm install -D @babel/preset-env @babel/preset-react
> cat ./package.json
> vi .babelrc
```

```
{
  ...,
  "scripts": {
    "build": "webpack --mode production"
  }
  ...,
  "dependencies": {
    ...,
  },
  "devDependencies": {
    "@babel/core": "^7.19.1",
    "@babel/preset-env": "^7.19.1",
    "@babel/preset-react": "^7.18.6",
    "babel-loader": "^8.2.5",
    "webpack": "^5.74.0",
    "webpack-cli": "^4.10.0"
  }
}
```

ch05/recipes-app-01/package.json

```
{
    "presets": ["@babel/preset-env", "@babel/preset-react"]
}
```

ch05/recipes-app-01/.babelrc

개발

```
> npx webpack --mode development
```

운영

```
> npm run build  ⟶  npx webpack --mode production
```

```
{
  ...,
  "scripts": {
    "build": "webpack --mode production"
  },
  ...,
}
```

번들 생성 `./dist/assets/bundle.js` webpack.config.js 설정의 output 확인

실행 (번들 로딩)

ch05/recipes-app-01/dist/index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>proj: recipes-app-01</title>
</head>
<body>

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

<script src="assets/bundle.js"></script>

</body>
</html>
```

Baked Salmon

- 연어
- 잣
- 버터 상추
- 옐로 스쿼시
- 올리브 오일
- 마늘

recipe

오븐을 180도로 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.

Fish Tacos

```
<!DOCTYPE html>
<html>
▶ <head>…</head>
▼ <body>
    <!-- Root element -->
… ▼ <div id="root"> == $0
    ▼ <section>
      ▶ <header>…</header>
      ▼ <div class="recipes">
        ▼ <section id="baked-salmon">
            <h1>Baked Salmon</h1>
          ▶ <ul class="ingredients">…</ul>
          ▼ <section>
              <h3>recipe</h3>
            ▶ <ul>…</ul>
            </section>
          </section>
        ▶ <section id="fish-tacos">…</section>
        </div>
      </section>
    </div>
    <script src="assets/bundle.js"></script>
  </body>
</html>
```

소스 맵

https://webpack.kr/configuration/devtool

```
/* ./webpack.config.js */

const path = require("path");

module.exports = {
  entry: "./src/index.js",
  output: {
    path: path.join(
      __dirname, "dist", "assets"),
    filename: "bundle.js",
  },
  module: {
    rules: [{
      test: /\.js$/,
      exclude: /node_modules/,
      loader: "babel-loader"
    }]
  },
  devtool: 'source-map'
}
```

- 소스 맵: 원본 소스와 난독화 된 소스를 매핑 ※ devtool 옵션으로 소스 맵 선택

Baked Salmon

- 연어
- 잣
- 버터 상추
- 옐로 스쿼시
- 올리브 오일
- 마늘

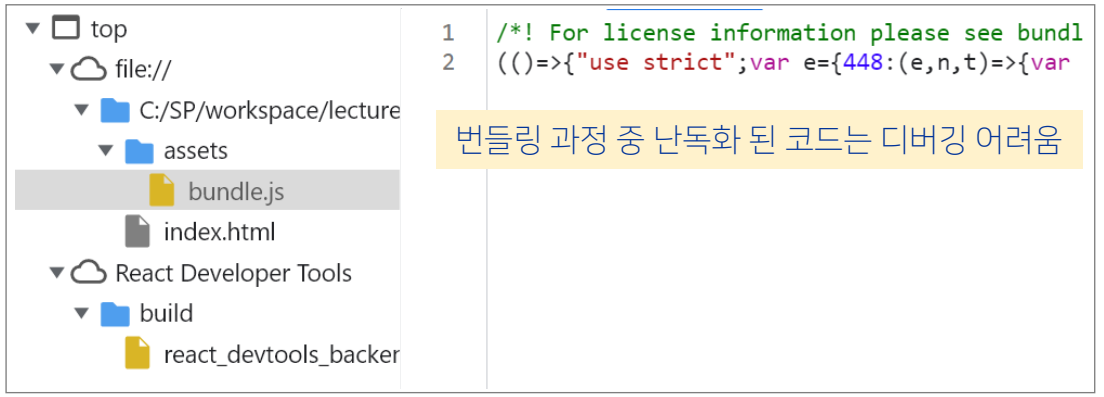
recipe

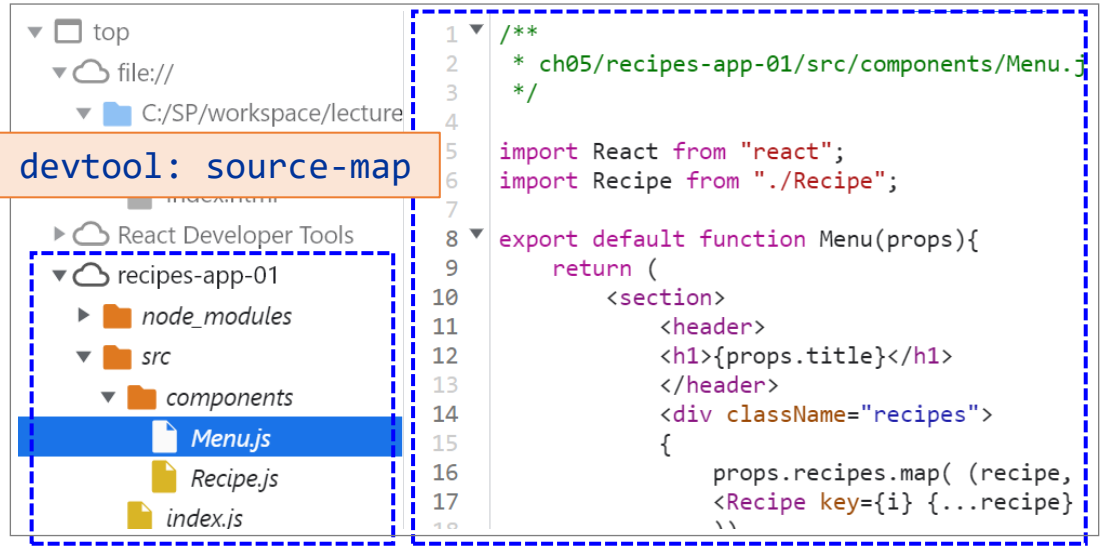
오븐을 180도로 예열한다.

유리 베이킹 그릇에 올리브
두른다.

연어, 마늘, 잣을 그릇에 담

오븐에서 15분간 익힌다.

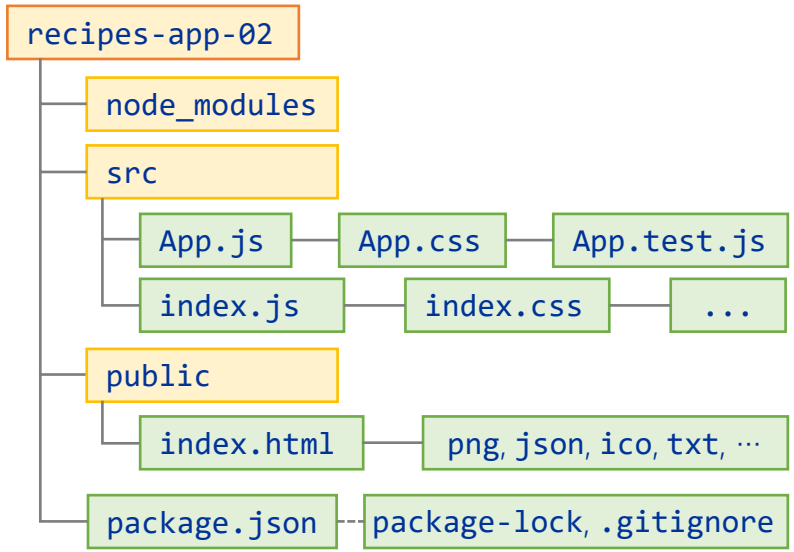




create-react-app (프로젝트 생성)

- create-react-app
  - 리액트 프로젝트를 생성하는 도구
  - 프로젝트 구조와 패키지별 의존관계 설정
  - npm 패키지 설치 필요

```
> npm install -g create-react-app
```
npm 전역 패키지 필요



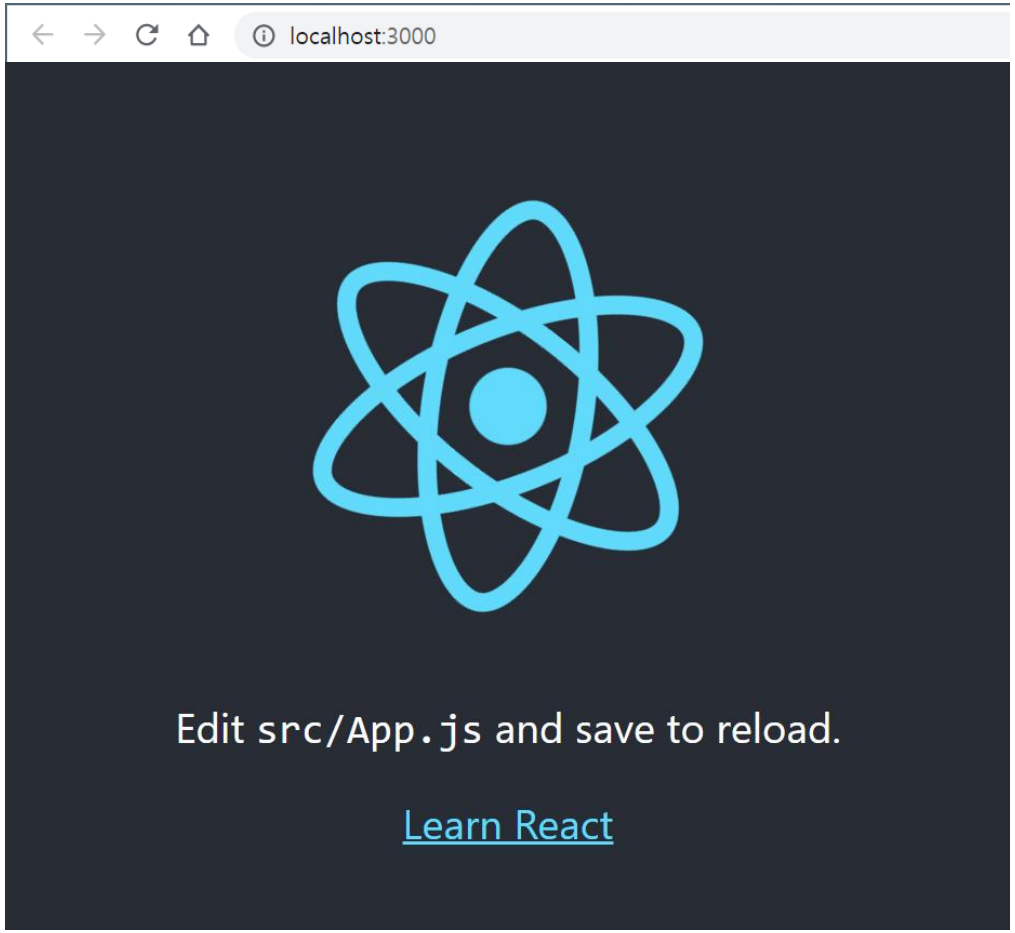

```
> create-react-app project_name
```

```
> create-react-app recipes-app-02

Creating a new React app in ~\recipes-app-02.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

Initialized a git repository.

Installing template dependencies using npm...

Created git commit.
Success! Created recipes-app-02 at ~\recipes-app-02
Inside that directory, you can run several commands:

  npm start - Starts the development server.
  npm run build - Bundles the app into static files for production.
  npm test - Starts the test runner.
  npm run eject - Removes this tool and copies build dep..

We suggest that you begin by typing:
  cd recipes-app-02
  npm start
Happy hacking!
```

프로젝트 생성 중 의존관계 설정
(react, react-dom, react-scripts)

create-react-app (실행)

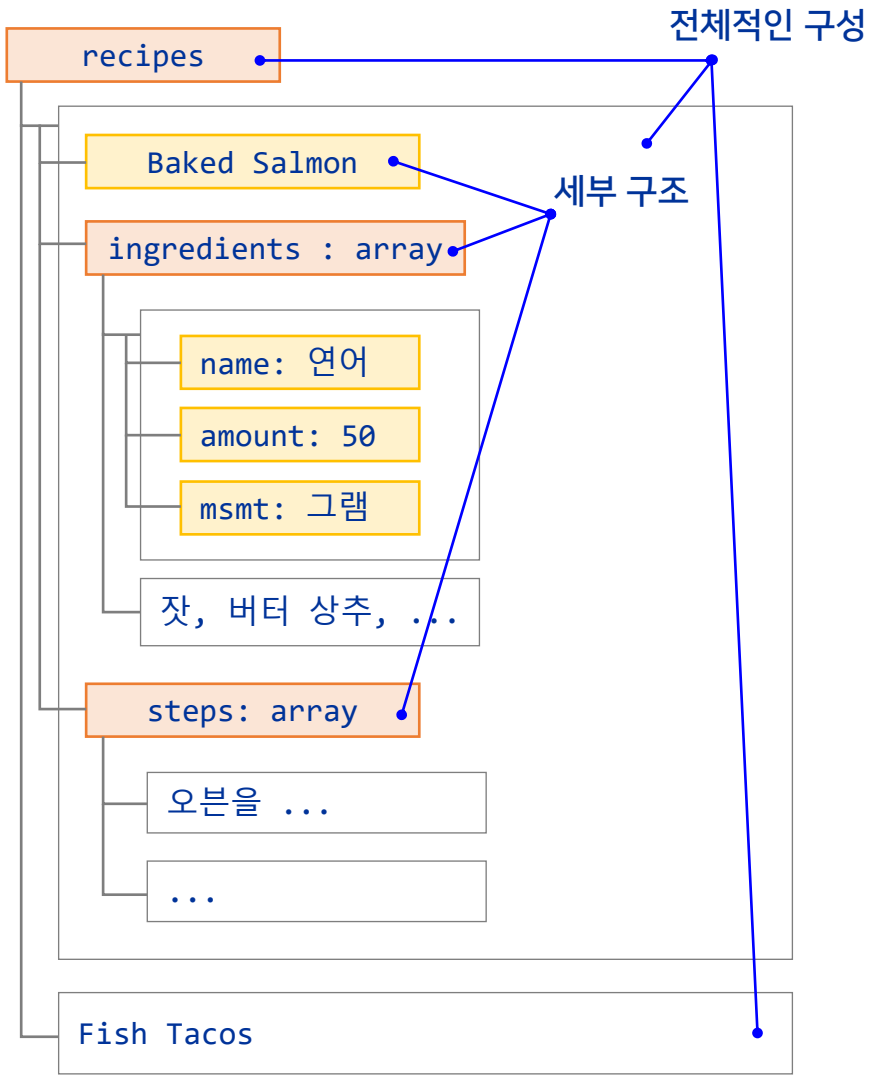
```
> cd recipes-app-02
recipes-app-02> npm start

Compiled successfully!

You can now view recipes-app-02 in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.6:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



정리: 5장. JSX를 사용하는 리액트

- JSX
- Babel
- JSX 활용 예제
  - recipes-app
- 웹팩
- 프로젝트 구성하기
  - create-react-app 활용

참조. JSX 활용 예제: Recipes




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

<!-- React Library, React DOM & babel-->
<script src=react.js, react-dom.js, babel.js></script>

<script type="text/babel">

/* ch05-05-1-recipe.html */

// data
const data = [ ... ];

// component: describing a recipe
const Recipe = function( ... ){ ... };

// component: describing a menu
const Menu = function(...){ ... };

// Menu를 현재의 DOM 안에 렌더링
ReactDOM.render(
    <Menu recipes={data} title="recipes" />,
    document.getElementById('root')
)

</script>
```

세부 구조

전체적인 구성

참조. JSX 활용 예제: Recipes

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

```
<script type="text/babel">
/* ch05-05-1-recipe.html */

// data
const data = [ ... ];

// component: Menu
const Menu = function;

// component: Recipe
const Recipe = function;

// rendering
ReactDOM.render();
```

```
// component: describing a menu
const Menu = function(props){
  return (
    <section>
      <header>
        <h1>{props.title}</h1>
      </header>
      <div className="recipes">
        { props.recipes.map( (recipe, i) =>
          (<Recipe key={i}
            name={recipe.name}
            ingredients={recipe.ingredients}
            steps={recipe.steps} />) ) }
      </div>
    </section>
  );
}
```

```
// component: describing a recipe
const Recipe = function({name, ingredients, steps}){
  return (
    <section>
      <h1>{name}</h1>
      <ul>
        {ingredients.map( (ingrd, i) =>
          (<li key={i}>{ingrd.name}</li>) )}
      </ul>
      <section>
        <h3>recipe</h3>
        <ul>
          {steps.map( (step, i) =>
            (<p key={i}>{step}</p>) )}
        </ul>
      </section>
    </section>
  );
}
```

```
// Menu를 현재의 DOM 안에 렌더링
ReactDOM.render(
  <Menu recipes={data} title="recipes" />,
  document.getElementById('root')
)
```

```
<div id="root">
  <Menu />
</div>
```

```
<section>
  <header>
    <h1></h1>
  </header>
  <div>
    <Recipe />
  </div>
</section>
```

```
<section>
  <h1></h1>
  <ul>  <li><li>...<li>  </ul>
  <section>
    <h3></h3>
    <ul> <p><p>...<p> </ul>
  </section>
</section>
```

참조 코드: ch05-05-2-recipe.html


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

```
const data = [   // data
 {
  "name": "Baked Salmon",
  "ingredients": [
   { "name": "연어", "amount": 500, ... },
   ...,
  ],
  "steps": [
   "오븐을 180도로 예열한다.",
   ...,
  ]
 },
 {
  "name": "Fish Tacos",
  "ingredients": [
   { "name": "흰살생선", "amount": 500, ... },
   ...,
  ],
  "steps": [
   "생선을 그릴에 익힌다.",
   ...,
  ]
 }
]
```

```
// component: describing a recipe
const Recipe = function({name, ingredients, steps}){
  return ( <section id={name.toLowerCase().replace(/ /g, "-")}>
      <h1>{name}</h1>
      <ul className="ingredients">
        { ingredients.map( (ingrd, i) => (<li key={i}>{ingrd.name}</li>) ) }
      </ul>
      <section>
        <h3>recipe</h3>
        <ul>
          { steps.map( (step, i) => (<p key={i}>{step}</p>) ) }
        </ul>
      </section>
    </section> );
}
```

```
// component: describing a menu
const Menu = function(props){
  return (
    <section>
      <header>
        <h1>{props.title}</h1>
      </header>
      <div className="recipes">
        { props.recipes.map( (recipe, i) => (
            <Recipe key={i} {...recipe} /> )) }
      </div>
    </section>
  );
}
```

```
// Menu를 현재의 DOM 안에 렌더링
ReactDOM.render(
  <Menu recipes={data}
        title="recipes" />,
  document.getElementById('root')
)
```