

React 기본 과정

inky4832@daum.net

1장. React 개요 및 환경설정



React 개요

React 환경설정

<https://reactjs.org/>

가. facebook에서 Reactjs 발표

나. UI상태 자동 관리

다. 가상 DOM을 이용한 빠른 DOM 조작 가능

라. 자바스크립트로 정의하는 화면

마. MVC의 ‘V’ 담당

Visual한 요소와 그 상태를 최신으로 유지하는데 중점을 둔 자바스크립트 라이브러리.



2. React 환경 설정

Reactjs

가. node.js 설치

<http://nodejs.org>

The screenshot shows the official Node.js website. At the top, there's a navigation bar with links: HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. Below the navigation, a green banner says "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine." Another green banner below it says "Join us at OpenJS World, a free virtual event on June 2-3, 2021". A large button labeled "Download for Windows (x64)" is centered. Underneath it, two buttons are shown: "14.17.0 LTS" (Recommended For Most Users) and "16.2.0 Current" (Latest Features). Both buttons have a red border around them. At the bottom of the page, there's a note about the Long Term Support (LTS) schedule.

```
C:\WINDOWS\system32>node -v  
v14.17.0  
C:\WINDOWS\system32>
```

나. 개발 툴 설치 (Visual Studio Code)

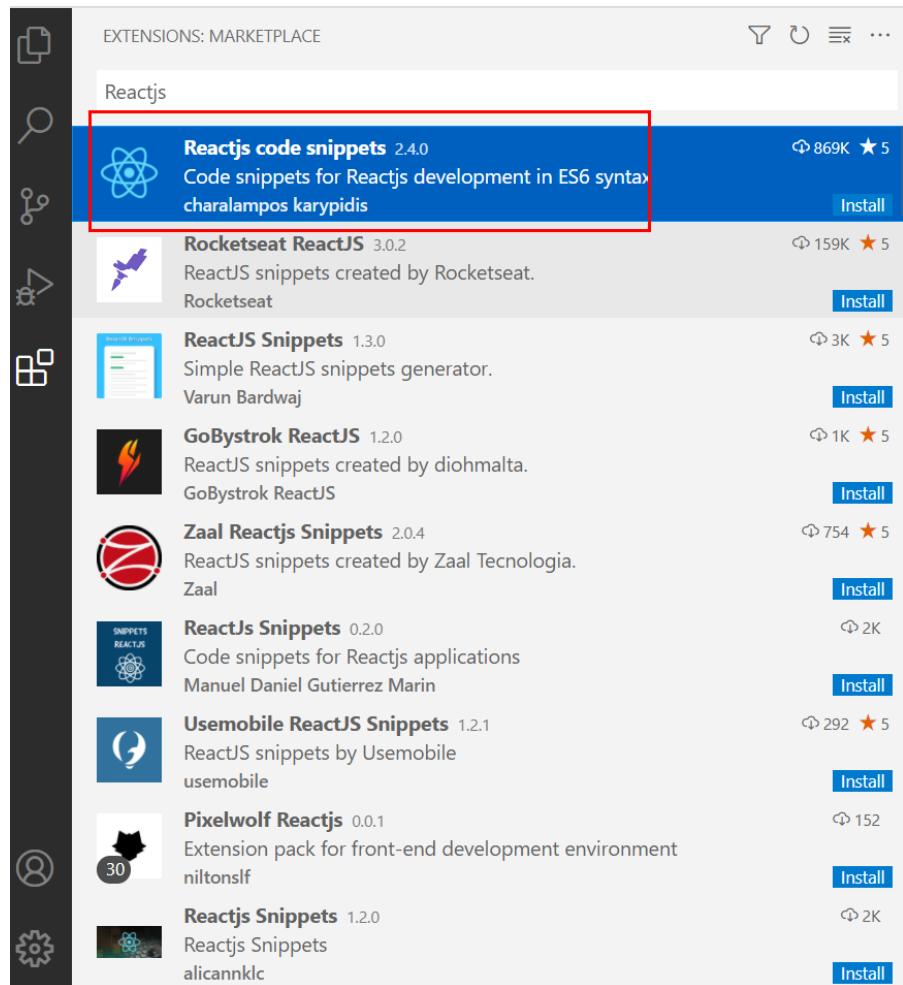
<https://code.visualstudio.com>

The screenshot shows the Visual Studio Code website. The header includes links for Visual Studio Code, Docs, Updates, Blog, API, and Extensions. A message at the top says "Version 1.56 is now available! Read all about it". The main content features the tagline "Code editing. Redefined." and "Free. Built on open source. Runs everywhere." Below this, a large blue button says "Download for Windows" with a dropdown arrow. A red box highlights this button. Below it, a table lists download options: "macOS" (Universal), "Windows x64" (User Installer), and "Linux x64". The "Windows x64" row is also highlighted with a red box. On the right side, there's a sidebar titled "EXTENSIONS: MARKETPLACE" listing various extensions like Python, GitLens, ESLint, Debug, Language, vscode-icons, and Vetur.

2. React 환경 설정

Reactjs

다. VS Code 확장 프로그램 설치



EXTENSIONS: MARKETPLACE

Reactjs

Reactjs code snippets 2.4.0
Code snippets for Reactjs development in ES6 syntax
charalampous karypidis

869K ⭐ 5

Install

Rocketseat ReactJS 3.0.2
ReactJS snippets created by Rocketseat.
Rocketseat

159K ⭐ 5

Install

ReactJS Snippets 1.3.0
Simple ReactJS snippets generator.
Varun Bardwaj

3K ⭐ 5

Install

GoBystrok ReactJS 1.2.0
ReactJS snippets created by diohmalta.
GoBystrok ReactJS

1K ⭐ 5

Install

Zaal Reactjs Snippets 2.0.4
ReactJS snippets created by Zaal Tecnologia.
Zaal

754 ⭐ 5

Install

ReactJs Snippets 0.2.0
Code snippets for Reactjs applications
Manuel Daniel Gutierrez Marin

2K

Install

Usemobile ReactJS Snippets 1.2.1
ReactJS snippets by Usemobile
usemobile

292 ⭐ 5

Install

Pixelwolf Reactjs 0.0.1
Extension pack for front-end development environment
niltonslf

152

Install

Reactjs Snippets 1.2.0
Reactjs Snippets
alicannklc

2K

Install

Release Notes: 1.56.2

Extension: Reactjs code snippets

Reactjs code snippets xabikos.reactsnippets
charalampous karypidis | 869,594 | ★★★★★ | Repo

Code snippets for Reactjs development in ES6 syntax

Install

Details **Feature Contributions**

Spaces: 2 Ln 12, Col 35 UTF-8 CRLF JavaScript 1.8.2

Snippets

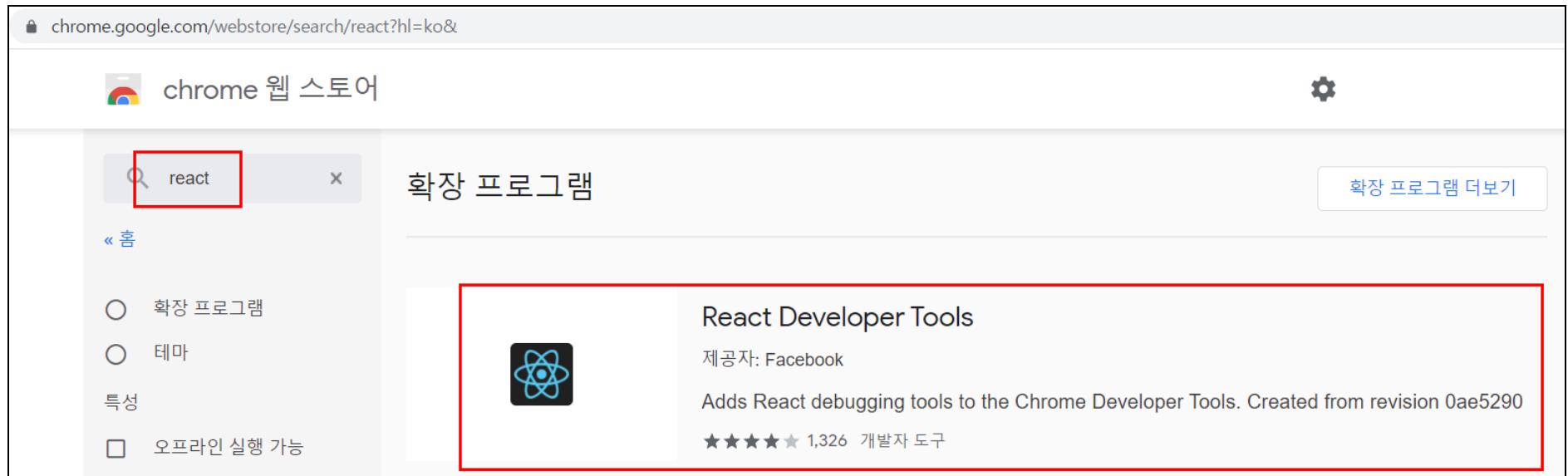
Below is a list of all available snippets and the triggers of each one. The → means the TAB key.

Trigger	Content
rcc→	class component skeleton
rrc→	class component skeleton with react-redux connect
rrdc→	class component skeleton with react-redux connect and dispatch
rccp→	class component skeleton with prop types after the class
rcjc→	class component skeleton without import and default export lines
rcfc→	class component skeleton that contains all the lifecycle methods
rwrd→	class component without import statements

2. React 환경 설정

Reactjs

라. 크롬 브라우저에 React 플러그인 설치



2장. my-app 프로젝트 실습



React toolchain 설치
my-app 프로젝트 실행 및 구조

<https://reactjs.org/docs/create-a-new-react-app.html>

Create a New React App

Use an integrated toolchain for the best user and developer experience.

This page describes a few popular React toolchains which help with tasks like:

- Scaling to many files and components.
- Using third-party libraries from npm.
- Detecting common mistakes early.
- Live-editing CSS and JS in development.
- Optimizing the output for production.

The toolchains recommended on this page **don't require configuration to get started**.

Recommended Toolchains

The React team primarily recommends these solutions:

- If you're **learning React** or **creating a new single-page app**, use [Create React App](#).
- If you're building a **server-rendered website with Node.js**, try [Next.js](#).
- If you're building a **static content-oriented website**, try [Gatsby](#).
- If you're building a **component library** or **integrating with an existing codebase**, try [More Flexible Toolchains](#).

1. React toolchain 설치

Reactjs

Create React App

Create React App is a comfortable environment for **learning React**, and is the best way to start building **a new single-page application** in React.

It sets up your development environment so that you can use the latest JavaScript features, provides a nice developer experience, and optimizes your app for production. You'll need to have Node >= 10.16 and npm >= 5.6 on your machine. To create a project, run:

```
npx create-react-app my-app  
cd my-app  
npm start
```

```
C:\WINDOWS\system32>cd C:\WSK_React  
C:\WSK_React>npx create-react-app my-app  
npx: installed 67 in 5.04s  
Creating a new React app in C:\WSK_React\my-app.
```

Success! Created `my-app` at `C:\WSK_React\my-app`
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 dependencies, configuration files and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

```
cd my-app  
npm start
```

Happy hacking!

```
C:\WSK_React>
```

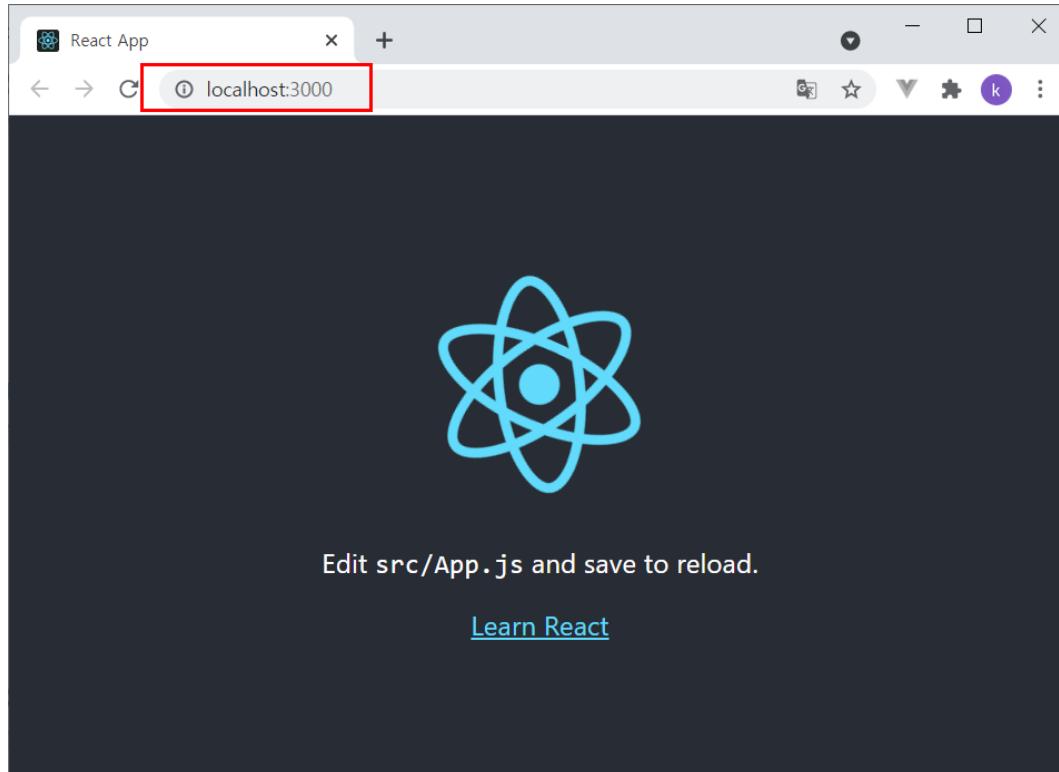
2. my-app 프로젝트 실행

Reactjs

```
관리자: Windows PowerShell
Compiled successfully!
You can now view my-app in the browser.

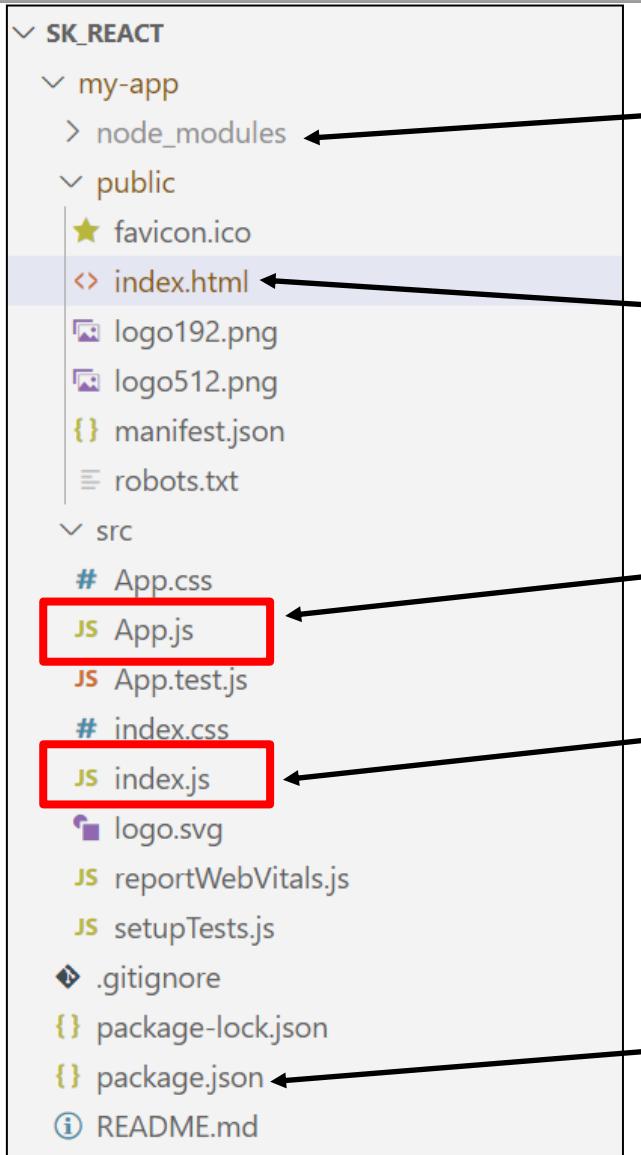
Local:          http://localhost:3000
On Your Network:  http://172.28.160.1:3000

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



3. my-app 프로젝트 구조

Reactjs



node.js는 node_modules 폴더를 생성하고 이곳에 package.json 파일에서 명시된 외부 모듈을 다운로드로 저장함.

메인 홈페이지 파일. 편집할 필요 없다.
어플리케이션을 빌드 할 때 자동으로 모든 js 및 css 파일을
동적으로 추가된다.

React컴포넌트 파일

Starting point. 어플리케이션의 App 컴포넌트를 로딩하여
웹브라우저에서 실행시킴.

Nodejs의 모듈 관리 설정 파일
프로젝트에서 사용하는 third party용 패키지를 나열

3장. 컴포넌트 (Component)



컴포넌트 개요 및 종류

컴포넌트 구성요소

컴포넌트 랜더링

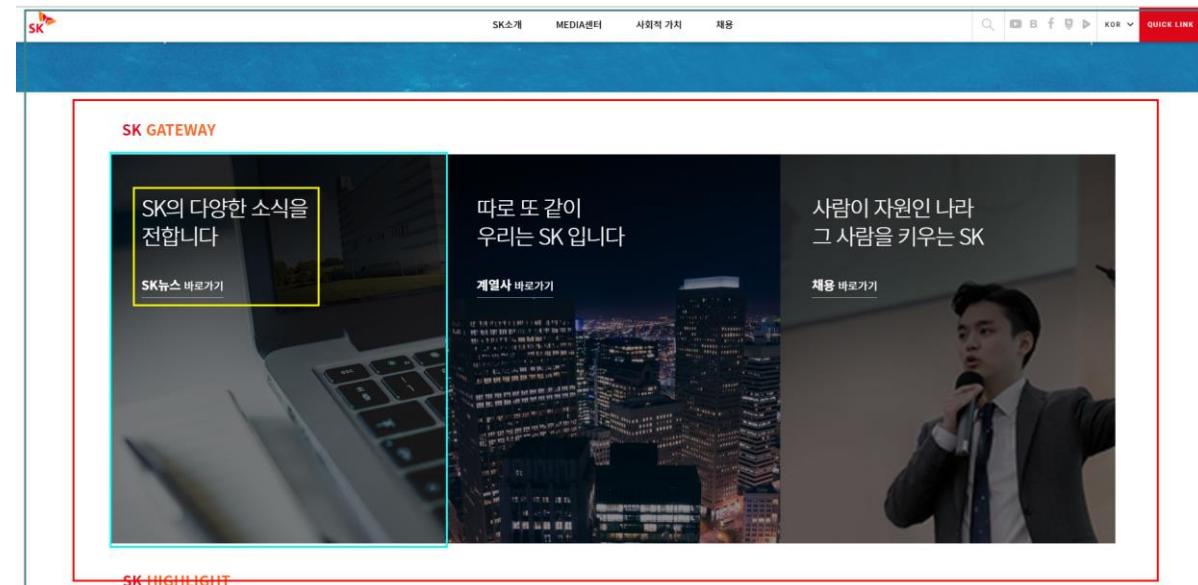
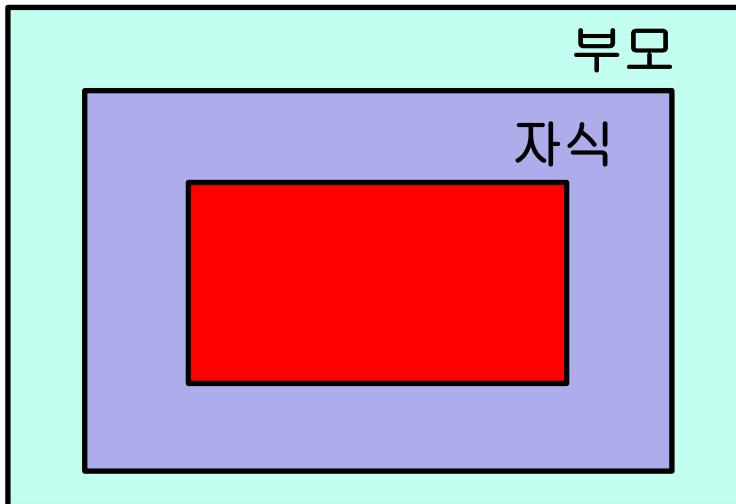
1. 컴포넌트 (Component) 개요

Reactjs

<https://ko.reactjs.org/docs/react-component.html>

○ 컴포넌트 개념

웹 화면에서 보여지는 개별적인 화면 블록을 의미하고, 일반적으로 중첩된 형태로 사용된다.



2. 컴포넌트 (Component) 생성 방법

Reactjs

<https://ko.reactjs.org/docs/components-and-props.html>

○ 컴포넌트 종류

가. 클래스형 컴포넌트

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

나. 함수형 컴포넌트

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

주의: 컴포넌트의 이름은 항상 대문자로 시작합니다.

React는 소문자로 시작하는 컴포넌트를 DOM 태그로 처리합니다. 예를 들어 `<div />`는 HTML div 태그를 나타내지만, `<Welcome />`은 컴포넌트를 나타내며 범위 안에 `Welcome`이 있어야 합니다.

3. 클래스형 컴포넌트 (Component) 구성요소

Reactjs

<https://ko.reactjs.org/docs/react-component.html>

가. 라이프 사이클 메서드

- `constructor()`
- `static getDerivedStateFromProps()`
- `render()`
- `componentDidMount()`
- `static getDerivedStateFromProps()`
- `shouldComponentUpdate()`
- `render()`
- `getSnapshotBeforeUpdate()`
- `componentDidUpdate()`
- `componentWillUnmount()`

다. 클래스 프로퍼티

- `defaultProps`
- `displayName`

라. 인스턴스 프로퍼티

- `props`
- `state`

나. 기타 메서드

- `setState()`
- `forceUpdate()`

함수형 컴포넌트는
props 만 사용 가능하다.

4. 컴포넌트 (Component) 랜더링

Reactjs

<https://ko.reactjs.org/docs/components-and-props.html#composing-components>

○ 컴포넌트 랜더링

클래스 또는 함수로 생성된 컴포넌트는 다음과 같이 중첩하여 사용된다.

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

function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

4. 컴포넌트 실습

Reactjs

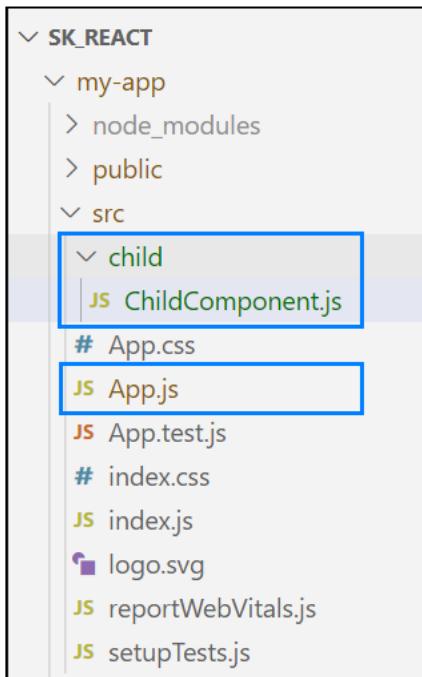
* React 컴포넌트 생성 실습

가. extends React.Component

나. render() 메서드 구현

다. return (html 형식의 JSX코드);

라. export default 컴포넌트명;



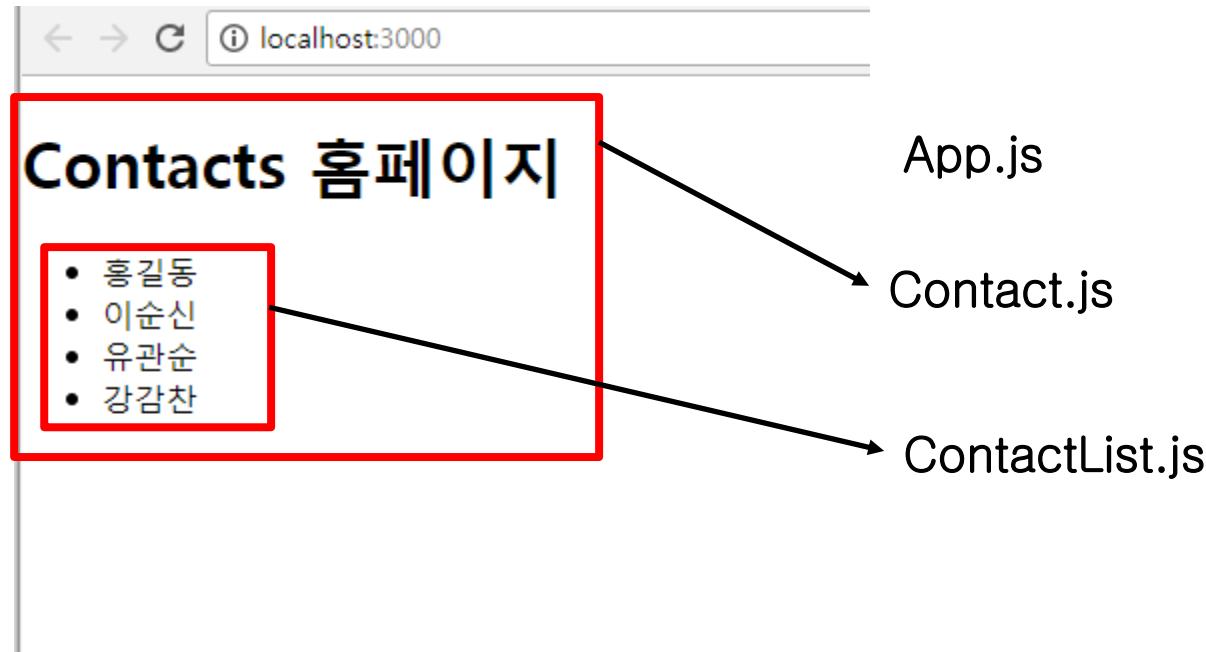
```
JS ChildComponent.js U ×  
my-app > src > child > JS ChildComponent.js > ...  
1 import React, { Component } from 'react';  
2  
3 class ChildComponent extends Component {  
4   render() {  
5     return (  
6       <div>  
7         |   자식 컴포넌트 입니다.  
8       </div>  
9     );  
10   }  
11 }  
12  
13 export default ChildComponent;
```

```
App.js M ×  
/app > src > JS App.js > ...  
1 import logo from './logo.svg';  
2 import './App.css';  
3 import ChildComponent from './child/ChildComponent';  
4  
5 function App() {  
6   return (  
7     <div className="App">  
8       |   부모 컴포넌트 입니다.  
9       <ChildComponent />  
10    </div>  
11  );  
12}  
13  
14 export default App;
```

5. 실습 1

Reactjs

다음과 같은 레이아웃을 가진 컴포넌트를 작성하시오.



4장. JSX



JSX 개요

JSX 사용시 주의할 점

- JSX는 JavaScript XML의 줄임말로 ‘자바스크립트에 XML을 추가한 확장형’ 문법이라고 할 수 있다.
- 일반 자바스크립트 문법이 아닌 JSX 문법을 사용하여 UI를 구현한다.
- JSX는 컴파일링 되면서 최적화되기 때문에 빠르다.
- 컴파일 단계에서 에러를 확인할 수 있다.
- HTML 문법과 비슷하여 더 쉽고 빠르게 UI 템플릿 작성 가능하다.

```
export class Contact extends Component {  
  render() {  
    return (  
      <h1>Contacts 홈페이지</h1>  
    );  
  }  
}
```

2. JSX 사용시 주의할 점

Reactjs

1) JSX에서 모든 태그는 종료태그가 필수이다.

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

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

2) JSX에서는 반드시 단 하나의 root 태그가 필수이다.

```
function App() {  
  return (  
    <div>  
      <ChildComponent />  
      <ChildComponent2 />  
    </div >  
  );  
}
```

function App() {
 return (**ERROR**
 <ChildComponent />
 <ChildComponent2 />
);
}

2. JSX 사용시 주의할 점

Reactjs

3) JSX에서 변수값 출력할 때 {} 사용한다.

```
class ChildComponent3 extends Component {  
  render() {  
    let mesg = "Hello";  
    return (  
      <div>  
        <h2>3)JSX: 변수값 출력시 인터플레이션을 사용한다.</h2>  
        메시지: {mesg}  
      </div>  
    );  
  }  
}
```

4) JSX에서 자바스크립트 코드 사용시 {} 사용한다.

```
class ChildComponent4 extends Component {  
  render() {  
    let names = ["홍길동", "이순신", "유관순"];  
    return (  
      <div>  
        <h2>4)JSX: 자바스크립트 코드 사용시 인터풀레이션을 사용한다.</h2>  
        <ul>  
          {  
            names.map((row, idx) => {  
              return <li key={idx}>{row}</li>  
            })  
          }  
        </ul>  
      </div>  
    );  
  }  
}
```

2. JSX 사용시 주의할 점

Reactjs

5) JSX에서 spread 연산자 사용 가능하다. (*)

```
class ChildComponent5 extends Component {
  render() {
    let attr = {
      href: "http://www.google.com",
      target: "_blank"
    };
    return (
      <div>
        <h2>5)JSX: spread 연산자 사용 가능하다.</h2>
        <a {...attr}>구글</a>
      </div>
    );
  }
}
```

6) JSX에서 이벤트 처리시 camel 표기법 필수이다.

```
class ChildComponent6 extends Component {
  render() {
    return (
      <div>
        <h2>6)JSX: 이벤트 처리시 camel 표기법 필수이다.</h2>
        <button onClick={() => console.log("OK")}>OK</button>
      </div>
    );
  }
}
```

2. JSX 사용시 주의할 점

Reactjs

7) JSX에서는 class 대신에 className 속성을 사용한다.

```
import './child.css';

class ChildComponent7 extends Component {
  render() {
    return (
      <div>
        <h2>7) JSX: class 대신에 className 속성을 사용한다.</h2>
        <p className="xyz">Hello</p>
      </div>
    );
  }
}
```

```
.xyz{
  font-size: large;
  color: red;
}
```

8) JSX에서 style은 객체 형식으로 사용한다.

```
class ChildComponent8 extends Component {
  render() {
    return (
      <div>
        <h2>8) JSX: class 대신에 className 속성을 사용한다.</h2>
        <p style={{ fontSize: '20px', backgroundColor: 'red' }}>Hello</p>
      </div>
    );
  }
}
```

9) JSX에서 주석은 {/* */} 형식이다.

```
return (
  <div>
    {/* 이것은 주석입니다*/}
    <p style={{fontSize: '20px', background: 'red'}}>world</p>
    <p style={myStyle}>happy</p>
  </div>
);
```

5장. Props 속성



Props 개요 및 사용방법

Props 타입 검사

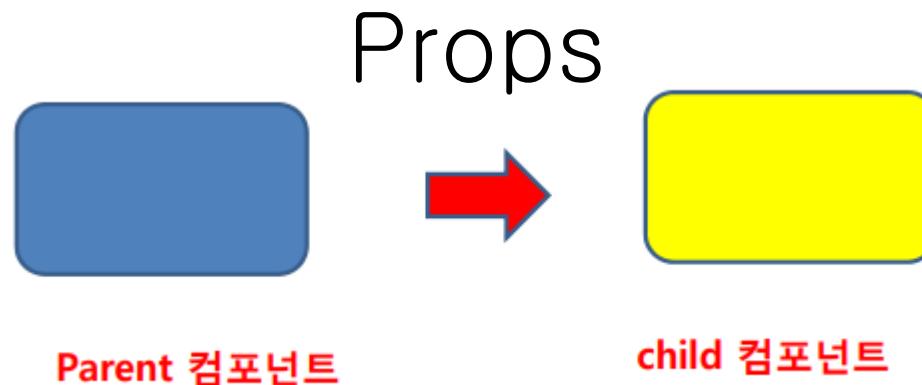
Default Props 값 설정

Spread 연산자 활용

<https://ko.reactjs.org/docs/components-and-props.html#props-are-read-only>

☞ **props는 읽기 전용입니다.**

- 컴포넌트에서 사용할 데이터 중에서 **변동되지 않는 (immutable)** 데이터를 처리 할 때 사용한다.
- 일반적으로 부모(parent)컴포넌트에서 자식(child) 컴포넌트로 데이터(함수포함)를 전달 할 때 사용된다.
- React에서는 모든 데이터의 흐름은 ‘**단 방향**’으로 처리해야 된다.



2. Props 사용1 – 속성에 값 설정

Reactjs

가. 부모(parent)에서 자식(child)태그의 속성에 값을 설정하여 전달한다.

propsName="value"

```
function App() {  
  return (  
    <div>  
      <ChildComponent name="홍길동" age={20} />  
    </div>  
  );  
}
```

문자열 이외의 값을 전달할 때는 반드시 {값} 형식으로 사용한다.

Props는 필수가 아니다.

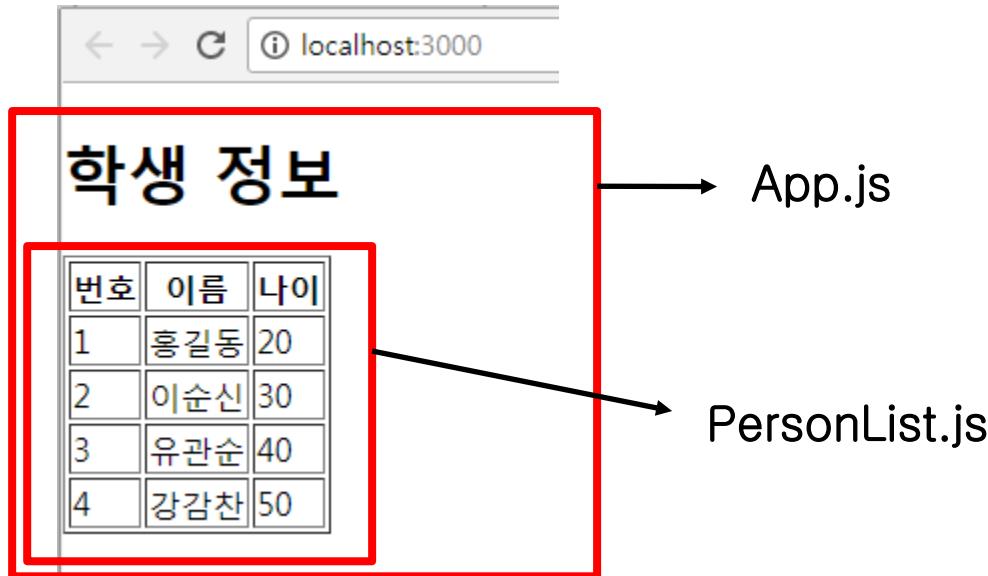
나. 자식(child)에서 속성값 얻기. (render() 메서드 안에서)

{ this.props.propsName }

```
class ChildComponent extends Component {  
  render() {  
    // Props 얻기  
    const { name, age } = this.props;  
    return (  
      <div>  
        이름:{name}<br />  
        나이:{age}  
      </div>  
    );  
  }  
}
```

3. 실습 2

Reactjs



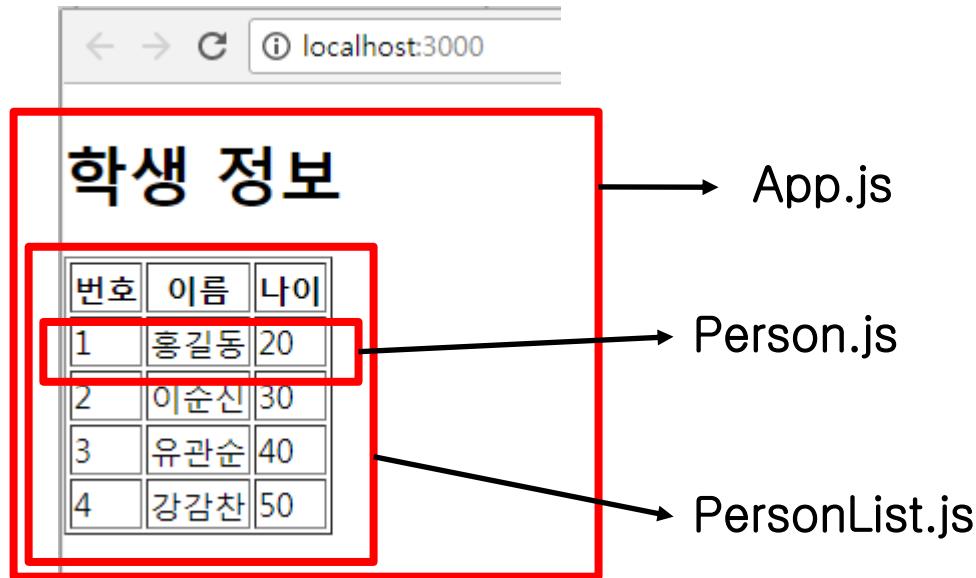
```
import logo from './logo.svg';
import './App.css';
import PersonList from './child/PersonList';

let persons = [
  { name: "홍길동", age: 20 },
  { name: "이순신", age: 30 },
  { name: "유관순", age: 40 },
  { name: "강감찬", age: 50 },
];

function App() {
  return (
    <div>
      <h1>학생 정보</h1>
    </div>
  );
}

export default App;
```

3. 실습 3



4. Props 사용2 – body에 값 설정

Reactjs

가. 부모(parent)에서 자식(child)태그의 몸체(body)에 값을 설정하여 전달.

```
function App() {
  return (
    <div>
      <ChildComponent name="홍길동" age={20}>
        Hello
        <span>World</span>
      </ChildComponent>
    </div>
  );
}
```

The diagram illustrates the flow of props from the parent component's render method to the child component's body. An arrow labeled "속성" (Props) points from the highlighted prop declaration in the parent's render method to the highlighted child component. Another arrow labeled "몸체(body)" points from the highlighted child component to its internal structure, specifically the span element.

나. 자식(child)에서 속성값 얻기. (render() 메서드 안)

{ this.props.children }

```
class ChildComponent extends Component {
  render() {
    // Props 얻기
    const { name, age, children } = this.props;
    return (
      <div>
        <h1>Hello</h1>
        <h2>Name: {name}</h2>
        <h2>Age: {age}</h2>
        {children}
      </div>
    );
  }
}
```

5. default Props 값 설정

Reactjs

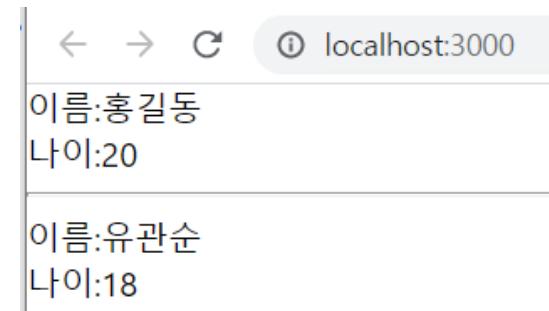
- 기본적으로 Props 속성은 필수가 아니다.
- 기본값을 설정하기 위해서는 컴포넌트 클래스 하단에

`className.defaultProps = { propName: value }` 를 삽입:

```
class ChildComponent extends Component {
  render() {
    // Props 얻기
    const { name, age } = this.props;
    return (
      <div>
        이름:{name}<br />
        나이:{age}
      </div >
    );
  }
}

// 기본 Props 설정
ChildComponent.defaultProps = {
  name: "유관순",
  age: 18
}
export default ChildComponent;
```

```
function App() {
  return (
    <div>
      <ChildComponent name="홍길동" age={20} />
      <hr />
      <ChildComponent />
    </div>
  );
}
```



6. Props 타입 검사

Reactjs

<https://reactjs.org/docs/typechecking-with-proptypes.html>

- 컴포넌트에서 원하는 Props의 type과 전달된 type 일치 여부 검증 가능.
- 추가로 필수(required) Props 설정도 가능.

```
import PropTypes from 'prop-types';
```

```
import PropTypes from 'prop-types';

MyComponent.propTypes = {
  // You can declare that a prop is a specific JS type. By default, these
  // are all optional.
  optionalArray: PropTypes.array,
  optionalBool: PropTypes.bool,
  optionalFunc: PropTypes.func,
  optionalNumber: PropTypes.number,
  optionalObject: PropTypes.object,
  optionalString: PropTypes.string,
  optionalSymbol: PropTypes.symbol,

  // Anything that can be rendered: numbers, strings, elements or an array
  // (or fragment) containing these types.
  optionalNode: PropTypes.node,

  // A React element.
  optionalElement: PropTypes.element,

  // A React element type (ie. MyComponent).
  optionalElementType: PropTypes.elementType,

  // You can also declare that a prop is an instance of a class. This uses
  // JS's instanceof operator.
  optionalMessage: PropTypes.instanceOf(Message),

  // You can ensure that your prop is limited to specific values by treating
  // it as an enum.
  optionalEnum: PropTypes.oneOf(['News', 'Photos']),
}
```

6. Props 타입 검사

Reactjs

```
function App() {
  return (
    <div>
      <ChildComponent
        boolValue={false}
        numValue={10}
        arrayValue={[1, 2, 3]}
        objValue={{ name: '홍길동', age: 20 }}
        nodeValue={<h1>노드</h1>}
        funcValue={() => { console.log("func") }}
        oneOfTypeValue={10}
        oneOfValue="여"
      />
    </div>
  );
}
```

```
// Props 타입 지정 : rpt
ChildComponent.propTypes = {
  boolValue: PropTypes.bool, //ptb
  numValue: PropTypes.number, //ptn
  arrayValue: PropTypes.array, //pta
  objValue: PropTypes.object, //pto
  nodeValue: PropTypes.node, //ptnd
  funcValue: PropTypes.func, //ptf
  oneOfTypeValue: PropTypes.oneOfType([
    PropTypes.string,
    PropTypes.number,
  ]), //ptet
  oneOfValue: PropTypes.oneOf(['남', '여']), // pte
};
```

```
<hr />
<ChildComponent />
</div>
```

```
// Props 타입 지정 : rpt
ChildComponent.propTypes = {
  boolValue: PropTypes.bool.isRequired, //ptb
  numValue: PropTypes.number, //ptn
  arrayValue: PropTypes.array, //pta
```

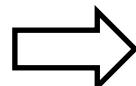
```
[HMR] Waiting for update signal from WDS
✖ ▶ Warning: Failed prop type: The prop `boolValue` is marked as required in `ChildComponent`, but its value is `undefined`.
  at ChildComponent (http://localhost:3000/static/js/main.chunk.js:312:1)
  at App
⚠ ▶ src\App.js
  Line 1:8:  'logo' is defined but never used  no-unused-vars
  >
```

7. static 키워드 이용한 설정

Reactjs

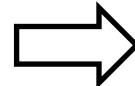
* transform-class-properties 방식

```
// 기본값 설정  
// Contact.defaultProps={  
//   mesg:"유관순",  
//   mesg2:200  
// }
```



```
export class Contact extends Component {  
  
  /*  
  If you are using a Babel transform like tr  
  , you can also declare defaultProps as st  
  */  
  static defaultProps={  
    mesg:"유관순",  
    mesg2:200  
  }  
  
  render(){  
    return(  
      <div>  
        <p>{mesg}</p>  
        <p>{mesg2}</p>  
      </div>  
    )  
  }  
}
```

```
//validate  
// Contact.propTypes={  
//   mesg: PropTypes.string,  
//   mesg2: PropTypes.number  
//   //mesg2: PropTypes.number.isRequired  
// }
```



```
export class Contact extends Component {  
  
  // 기본값 설정  
  static defaultProps={  
    mesg:"유관순",  
    mesg2:200  
  }  
  
  //validate  
  static propTypes={  
    mesg: PropTypes.string,  
    mesg2: PropTypes.number  
    //mesg2: PropTypes.number.isRequired  
  }  
}
```

8. spread 연산자 활용

Reactjs

Props를 여러 단계의 하위 컴포넌트로 전달하는 방법이다.

```
function App() {
  return (
    <div>
      <FirstChild name="홍길동" age={20} />
    </div>
  );
}
```

```
class FirstChild extends Component {
  render() {
    return (
      <div>
        <SecondChild {...this.props} address="서울" />
      </div>
    );
  }
}
```

```
class SecondChild extends Component {
  render() {
    // Props 얻기
    const { name, age, address } = this.props;
    return (
      <div>
        이름:{name}<br />
        나이:{age}<br />
        주소:{address}<br />
      </div>
    );
  }
}
```

6장. 이벤트 처리



이벤트 처리

콜백함수에서의 this 사용

계층구조 이벤트 처리

<https://ko.reactjs.org/docs/handling-events.html>

React 엘리먼트에서 이벤트를 처리하는 방식은 DOM 엘리먼트에서 이벤트를 처리하는 방식과 매우 유사합니다. 몇 가지 문법 차이는 다음과 같습니다.

- React의 이벤트는 소문자 대신 캐멀 케이스(camelCase)를 사용합니다.
- JSX를 사용하여 문자열이 아닌 함수로 이벤트 핸들러를 전달합니다.

기존 HTML의 이벤트 처리

```
<button onclick="activateLasers()">  
  Activate Lasers  
</button>
```

React의 이벤트 처리

```
<button onClick={activateLasers}>  
  Activate Lasers  
</button>
```

2. 콜백 함수에서의 this 사용

Reactjs

JSX 콜백 안에서 `this`의 의미에 대해 주의해야 합니다. JavaScript에서 클래스 메서드는 기본적으로 바인딩되어 있지 않습니다. `this.handleClick`을 바인딩하지 않고 `onClick`에 전달하였다면, 함수가 실제 호출될 때 `this`는 `undefined`가 됩니다.

이는 React만의 특수한 동작이 아니며, JavaScript에서 함수가 작동하는 방식의 일부입니다. 일반적으로 `onClick={this.handleClick}`과 같이 뒤에 `()`를 사용하지 않고 메서드를 참조할 경우, 해당 메서드를 바인딩 해야 합니다.

바인딩 없이 사용하는 경우1

```
class LoggingButton extends React.Component {
  // 이 문법은 `this`가 handleClick 내에서 바인딩되도록 합니다.
  // 주의: 이 문법은 *실험적인* 문법입니다.
  handleClick = () => {
    console.log('this is:', this);
  }

  render() {
    return (
      <button onClick={this.handleClick}>
        Click me
      </button>
    );
  }
}
```

바인딩 없이 사용하는 경우2

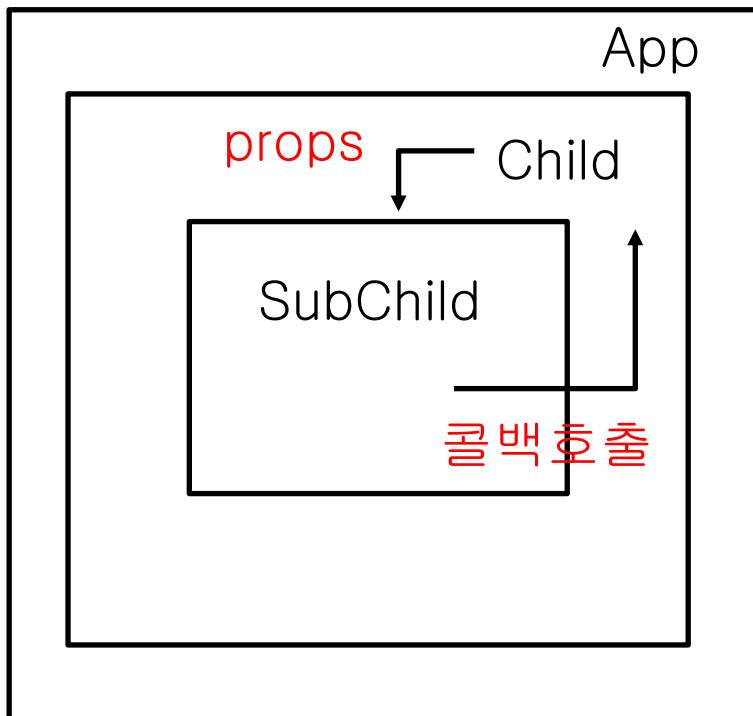
```
class LoggingButton extends React.Component {
  handleClick() {
    console.log('this is:', this);
  }

  render() {
    // 이 문법은 `this`가 handleClick 내에서 바인딩되도록 합니다.
    return (
      <button onClick={() => this.handleClick()}>
        Click me
      </button>
    );
  }
}
```

3. 계층구조 Events 처리

Reactjs

부모의 콜백 함수를 자식에서 호출할 수 있는 방법으로서 Props를 활용한다.



```
class ChildComponent extends Component {  
  
    // 콜백함수  
    handleChildEvent() {  
        console.log("handleChildEvent");  
    }  
    render() {  
        return (  
            <div>  
                <SubChildComponent onEvent={this.handleChildEvent} />  
            </div>  
        );  
    }  
}
```

```
class SubChildComponent extends Component {  
    render() {  
        const { onEvent } = this.props;  
        return (  
            <div>  
                <h2>ChildComponent 콜백 호출</h2>  
                <button onClick={onEvent}>call1</button>  
            </div>  
        );  
    }  
}
```

7장. 컴포넌트 상태 관리



state 개요

자동으로 랜더링 되는 경우

State 사용시 주의할 점

State 특징

<https://ko.reactjs.org/docs/state-and-lifecycle.html>



1) state 개요

컴포넌트에서 사용할 데이터 중에서 **변경 가능한 데이터(mutable)**를 다룰 때 사용.

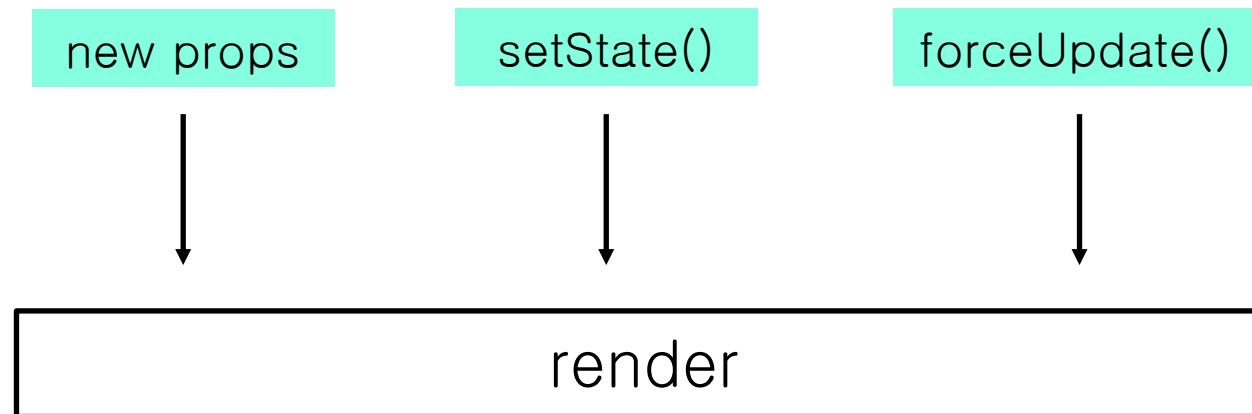
2) state 사용 방법

- 가. state 의 초기화는 constructor에서 **this.state={key:value}** 형식으로 설정.
- 나. state 를 랜더링하기 위해서는 **{ this.state.key }** 사용.
- 다. state를 업데이트 할 때는 반드시 **this.setState() {}** 메서드 사용
(**this.state=** 값 형식으로 직접 수정 불가)

2. 자동으로 랜더링 되는 경우

React에서 자동으로 화면을 re-rendering 하는 경우는 다음과 같다.

1. 새로운 Props 발생
2. setState() 호출하여 state가 변경된 경우
3. forceUpdate()



3. State내의 배열 사용시 주의할 점

```
constructor(){
  super();
  this.state={
    mem:[],
    inputValue:''
  };
  this.handleClick = this.handleClick.bind(this);
  this.handleChange = this.handleChange.bind(this);
}//end constructor
```

직접 state의 mem배열에 push하면 안되고
새로운 값을 할당하는 방식으로 사용해야 된다.

```
handleClick(){
  let addMem= this.state.mem;
  addMem.push(this.state.inputValue);
  this.setState({
    mem: addMem
  });
}
```

4. State내의 input 태그 사용시 주의할 점

Reactjs

<https://ko.reactjs.org/docs/forms.html#controlled-components>

```
class NameForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {value: ''};

    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }

  handleChange(event) {
    this.setState({value: event.target.value});
  }

  handleSubmit(event) {
    alert('A name was submitted: ' + this.state.value);
    event.preventDefault();
  }

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <label>
          Name:
          <input type="text" value={this.state.value} onChange={this.handleChange}>
        </label>
        <input type="submit" value="Submit" />
      </form>
    );
  }
}
```

onChange 이벤트를 이용하여 텍스트 입력시 state를 업데이트하도록 설정한다.

5. State 사용방법

Reactjs

가) 생성자에서 state 초기화

```
constructor(){
  super();
  this.state = {
    xyz: 0,
    number: 100
  };
  this.handleClick = this.handleClick.bind(this);
}//end constructor
```

나) state 값 랜더링

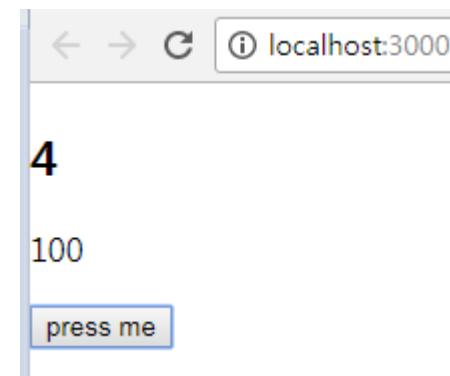
```
render(){
  return (
    <div>
      <h2>{this.state.xyz}</h2>
      <p>{this.state.number}</p>
      <button onClick={this.handleClick}>press me</button>
    </div>
  );
}
```

다) state 변경 (자동 re-rendering)

```
handleClick(){
  console.log(this);
  this.setState({
    xyz: this.state.xyz + 1
  });
}
```

라) 메서드 this 바인딩

```
this.handleClick = this.handleClick.bind(this);
```



6. Babel의 *transform-class-properties* 방식

Reactjs

```
export class Counter extends Component{  
  
    // 아래 생성자와 동일한 기능  
    state={  
        num:0  
    }  
  
    // constructor(){  
    //     super();  
    //     this.state={  
    //         num:0  
    //     };  
    // }  
    //arrow 함수 사용하면 bind 필요없다.  
    handleClick=(e)=>{  
        this.setState({  
            num: this.state.num + 1  
        });  
    }  
}
```

7. State 의 특징

Reactjs

1) State의 수정은 비동기적일 수 있다. <https://ko.reactjs.org/docs/faq-state.html>

React는 성능을 위해 여러 `setState()` 호출을 단일 업데이트로 한꺼번에 처리할 수 있습니다.

`this.props` 와 `this.state` 가 비동기적으로 업데이트될 수 있기 때문에 다음 state를 계산할 때 해당 값에 의존해서는 안 됩니다.

예를 들어, 다음 코드는 카운터 업데이트에 실패할 수 있습니다.

```
// Wrong
this.setState({
  counter: this.state.counter + this.props.increment,
});
```

이를 수정하기 위해 객체보다는 함수를 인자로 사용하는 다른 형태의 `setState()` 를 사용합니다. 그 함수는 이전 state를 첫 번째 인자로 받아들일 것이고, 업데이트가 적용된 시점의 props를 두 번째 인자로 받아들일 것입니다.

```
// Correct
this.setState((state, props) => ({
  counter: state.counter + props.increment
}));
```

7. State의 특징

Reactjs

2) State의 수정은 병합(merge)가 가능하다.

```
class Gateway extends Component {
  state = {
    error: null,
    isLoading: false,
    data: null,
  }

  componentDidMount() {
    this.fetchData()
  }

  fetchData() {
    // 데이터 요청 (비동기)
    // 로딩 표시
    this.setState({
      ...this.state,
      isLoading: true
    })
    // 데이터 요청
    // 1. fetch api
    fetch(`${process.env.PUBLIC_URL}/api/sk.json`)
      .then(
        (response) => response.json(),
        (error) => console.error(error.message)
      )
      .then((json) => {
        this.setState({
          data: json.gateway
        })
      })
  }
}
```

3) State 끌어 올리기

<https://ko.reactjs.org/docs/lifting-state-up.html>

State 끌어올리기

종종 동일한 데이터에 대한 변경사항을 여러 컴포넌트에 반영해야 할 필요가 있습니다. 이럴 때는 가장 가까운 공통 조상으로 state를 끌어올리는 것이 좋습니다.

8. 실습 4

Reactjs

1. 초기화면

state 실습

step: 1
num: 0

- +

2. Step 선택

state 실습

step: 1
num: 1

- +

3. + 선택

state 실습

step: 2
num: 6

- +

4. - 선택

state 실습

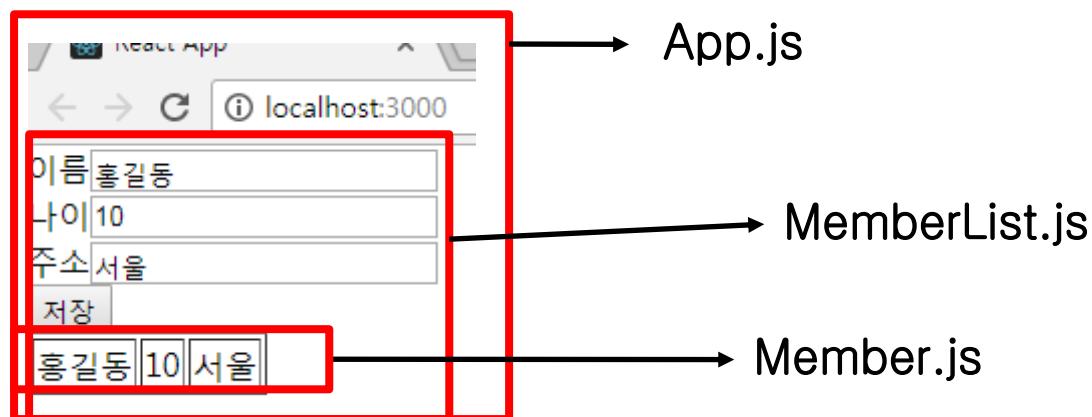
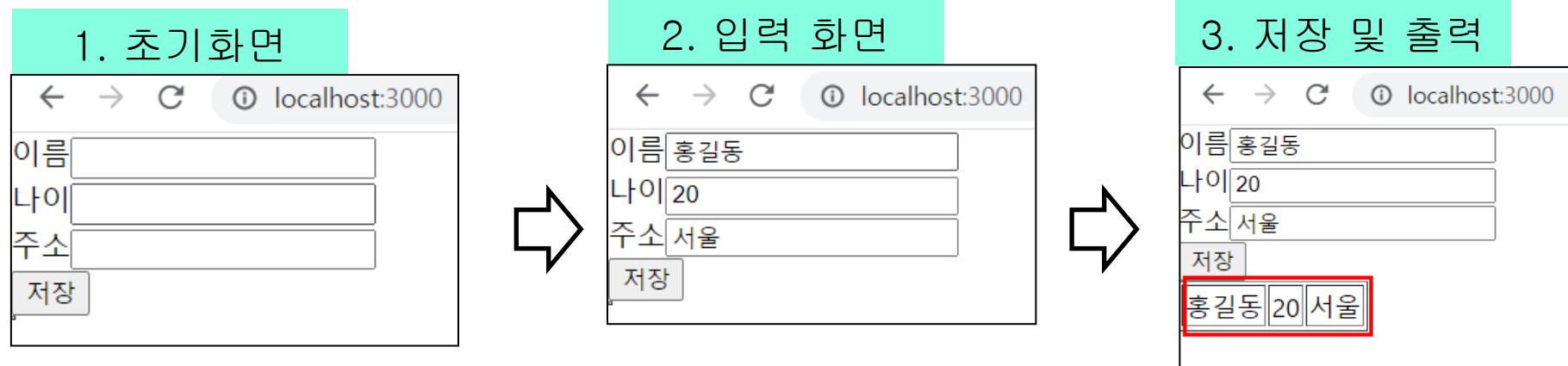
step: 2
num: 1

- +

음수값 허용 불가,
0으로 초기화

9. 실습 5

Reactjs



```
class MemberList extends Component {  
  constructor(props) {  
    super(props);  
    this.state = {  
      memberData: [],  
      username: "",  
      age: "",  
      address: ""  
    }  
  }  
}
```

8장. 라이프사이클 메서드



라이프 사이클 메서드 종류 및 구조

1. 라이프 사이클 메서드 종류

Reactjs

<https://ko.reactjs.org/docs/react-component.html#the-component-lifecycle>

클래스 컴포넌트는 여러 종류의 ‘라이프사이클 메서드’를 가지며, 이 메서드를 재정의하여 특정 시점에 코드가 실행되도록 설정할 수 있다. (함수형 컴포넌트는 사용 불가)

1) 랜더링 전/후 (마운트 상태)

- `constructor()`
- `static getDerivedStateFromProps()`
- `render()`
- `componentDidMount()`

2) 수정 전/후 (업데이트 상태)

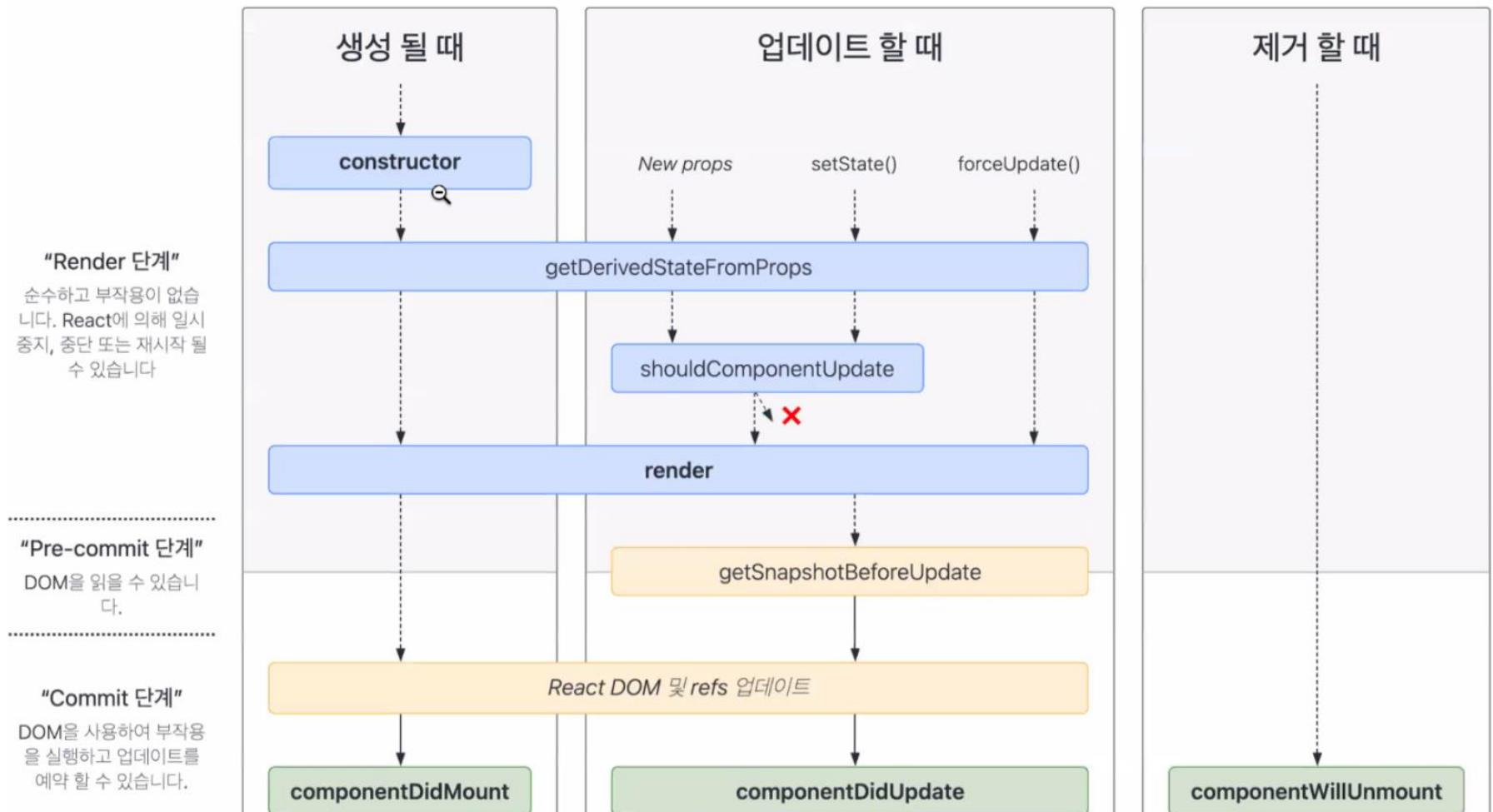
- `static getDerivedStateFromProps()`
- `shouldComponentUpdate()`
- `render()`
- `getSnapshotBeforeUpdate()`
- `componentDidUpdate()`

3) 컴포넌트 제거 (언마운트 상태)

- `componentWillUnmount()`

2. 라이프 사이클 메서드 구조

Reactjs



3. 마운트 상태의 라이프 사이클 메서드

Reactjs

1) 랜더링 전/후 (마운트 상태)

가. constructor

새로운 객체가 생성될 때마다 호출. super(props) 하여 this.props로 접근 가능, this.setState 또는 Ajax 불가

나. static getDerivedStateFromProps()

컴포넌트 초기화 또는 새로운 props를 받았을 때, this 키워드 사용 불가.

따라서 this.setState 대신에 state를 업데이트하기 위한 객체를 리턴하고 필요 없으면 null 값 반환.

다. render()

this.props와 this.state가 분석된다. state를 변경하면 안되고 브라우저와 직접 접근 불가

라. componentDidMount()

Ajax 요청 같은 데이터 로직 처리에 적합. DOM 접근 가능

2) 수정 전/후 (업데이트 상태)

`shouldComponentUpdate(nextProps, nextState){}`

state가 변경될 경우 `shouldComponentUpdate` 부터 lifecycle이 진행된다.

`getSnapshotBeforeUpdate(prevProps, prevState){}`

`render()` 메서드 호출 후 DOM에 변화를 반영하기 바로 전에 호출

여기에서 리턴된 값은 `componentDidUpdate()` 메서드의 세 번째 파라미터인 `snapshot`에서 받을 수 있다
주로 업데이트하기 직전의 값을 참고할 때 사용된다..

`componentDidUpdate (prevProps, prevState, snapshot){}`

모든 re-rendering이 완료된 후에 호출된다. 따라서 DOM관련 처리 가능하다.

`prevProps`와 `prevState` 값을 이용하여 컴포넌트가 이전에 가졌던 데이터에 접근할 수 있다.

5. 라이프 사이클 메서드가 필요한 이유

Reactjs

React의 성능 최적화를 위한 목적 또는 React가 컨트롤 할 수 없는 사이드 이펙트(Side effects, 부수효과)를 처리하기 위함이다.

다음은 대표적인 사이드 이펙트이다.

- 1) DOM 컨트롤 (실제 문서 접근 및 조작)
 - 대표적으로 jQuery 연동 및 웹 접근성 준수
- 2) 네트워크 통신 (비동기 통신 요청 및 응답)
 - Fetch API 또는 axios 라이브러리
- 3) 이벤트 핸들링 작업에서의 구독 및 취소 작업
 - 컴포넌트 생성시점에 이벤트를 구독했으면, 제거시점에 이벤트를 취소해야 한다.

9장. 함수형 컴포넌트



함수형 컴포넌트 개요 및 형태

1. 함수형 컴포넌트 (Functional Component) 개요

Reactjs

<https://ko.reactjs.org/docs/components-and-props.html#function-and-class-components>

다음과 같은 조건이 있는 경우에는 함수형 컴포넌트를 사용할 수 있다.

- 1) LifeCycle 메서드 없음
- 2) State도 사용하지 않음
- 3) Props만 사용하는 경우

1. 일반적인 Component 클래스 사용

```
class App extends Component {
  render() {
    return (
      <div>
        <ChildComponent mesg="ChildComponent"/>
        <ChildFunction mesg="ChildFunction"/>
        <ChildFunction2 mesg="ChildFunction2"/>
        <ChildDestructuring mesg="ChildDestructuring"/>
      </div>
    );
  }
}
```

클래스

```
export class ChildComponent extends Component{
  render(){
    return(
      <div>
        <h1>함수형 컴포넌트(Functional Component)1</h1>
        props:{this.props.mesg}
      </div>
    );
  }
}

//end Contact
```

2. 함수형 컴포넌트 (Functional Component) 형태

Reactjs

2. 함수형 컴포넌트 형태

일반 함수

```
export default function ChildFunction(props){  
    return(  
        <div>  
            <h1>함수형 컴포넌트(Functional Component)2</h1>  
            props:{props.msg}  
        </div>  
    );  
}
```

Arrow 함수

```
const ChildFunction2=(props)=>{  
    return(  
        <div>  
            <h1>함수형 컴포넌트(Functional Component)3</h1>  
            props:{props.msg}  
        </div>  
    );  
};  
  
export default ChildFunction2;
```

2. 함수형 컴포넌트 (Functional Component) 형태

Reactjs

3. Object Destructuring (*)

destructuring Arrow 함수

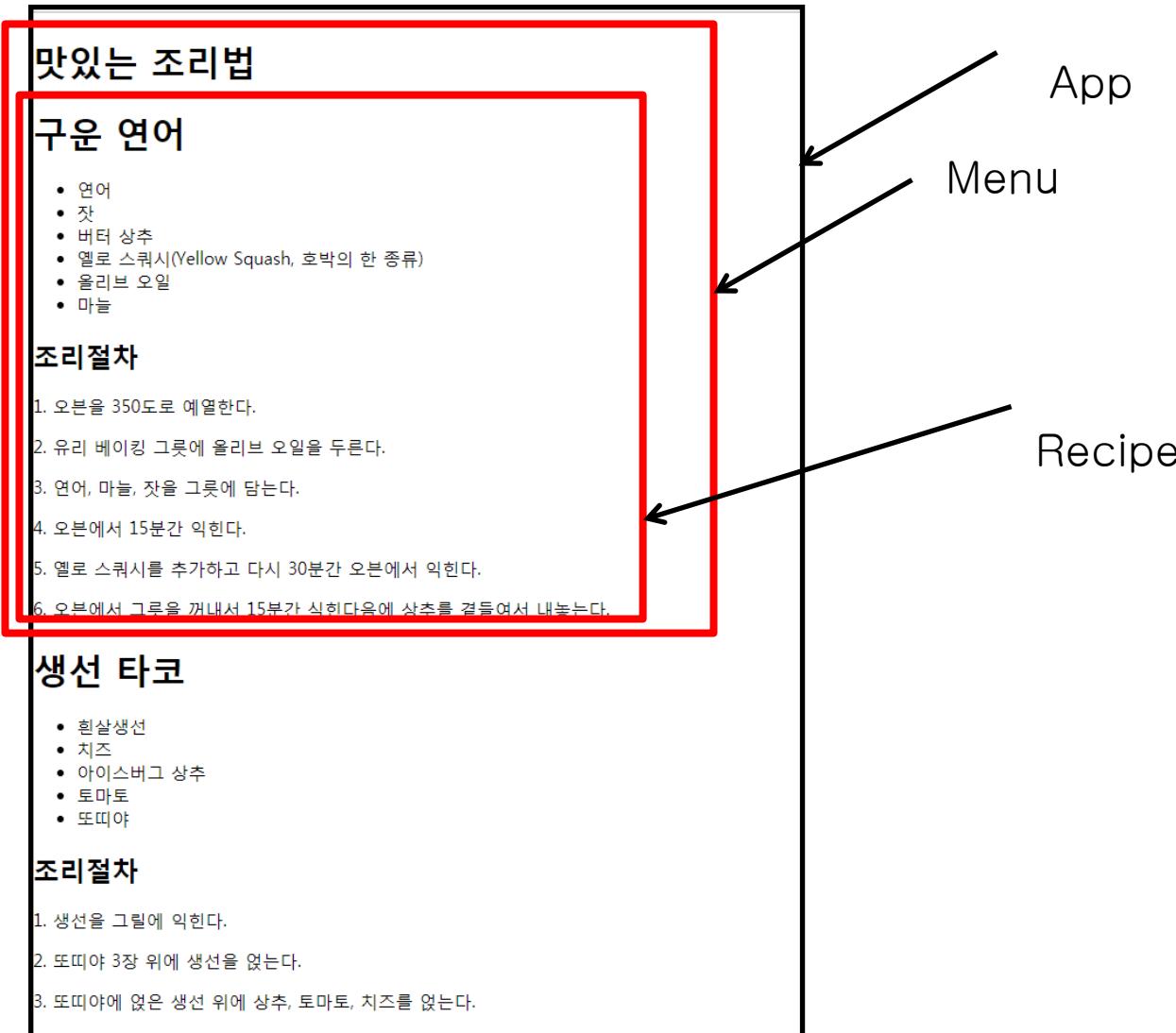
```
const ChildDestructuring = ({mesg})=>{
    return(
        <div>
            <h1>함수형 컴포넌트(Functional Component)</h1>
            props:{mesg}
        </div>
    );
}

export default ChildDestructuring;
```

3. 실습 6

Reactjs

제공된 App.js 파일을 활용하여 다음과 같이 구현 하시오.



10장. axios



axios 개요 및 설치

Json server 설치 및 실습

<https://ko.reactjs.org/docs/faq-ajax.html>

자바스크립트 환경에서 사용 가능한 비동기 Ajax 통신 방법은 다음과 같다.

- 1) XMLHttpRequest
- 2) Promise 함수
- 3) window.fetch 함수
- 4) axios 라이브러리 (*)

<https://github.com/axios/axios>

Installing

Using npm:

```
$ npm install axios
```

1. axios 개요

Reactjs

axios



Promise based HTTP client for the browser and node.js

New axios docs website: [click here](#)

Table of Contents

- Features
- Browser Support
- Installing
- Example
- Axios API
- Request method aliases
- Concurrency (Deprecated)
- Creating an instance
- Instance methods
- Request Config
- Response Schema
- Config Defaults
 - Global axios defaults
 - Custom instance defaults

axios API

Requests can be made by passing the relevant config to `axios`.

axios(config)

```
// Send a POST request
axios({
  method: 'post',
  url: '/user/12345',
  data: {
    firstName: 'Fred',
    lastName: 'Flintstone'
  }
});
```

Performing a `POST` request

```
axios.post('/user', {
  firstName: 'Fred',
  lastName: 'Flintstone'
})
.then(function (response) {
  console.log(response);
})
.catch(function (error) {
  console.log(error);
});
```

2. axios 함수

Reactjs

axios는 jQuery의 ajax 함수 만큼이나 편리하게 ajax를 처리할 수 있도록 다양한 API들을 제공한다.

```
axios.request(config)
```

```
axios.get(url[, config])
```

```
axios.delete(url[, config])
```

```
axios.head(url[, config])
```

```
axios.options(url[, config])
```

```
axios.post(url[, data[, config]])
```

```
axios.put(url[, data[, config]])
```

```
axios.patch(url[, data[, config]])
```

3. axios config 객체

Reactjs

axios 함수에 전달하는 config 객체는 다음과 같은 주요 속성을 갖는다.

```
let axios = axios({
  url: "./food.json", // 호출할 서버의 경로

  method: "get", // 사용하는 http method(post, get, put, delete)로 default는 get

  params: {
    name: "hong"
  }, // url 즉 쿼리스트링을 구성하는 파라미터 요소

  data: {
    age: 10,
    addr: "seoul"
  }, // request body를 통해서 서버로 전송되는 값(post, put, patch에서 사용)
});
```

요청에 대한 응답 결과는 then과 catch 콜백함수로 처리한다.

```
axios.then(
  success_callback
).catch(
  error_callback
).finally(
  finally_callback
);
```

```
{
  // 서버가 출력한 값은 언제나 data 속성 안에 존재한다.
  data: {},
  // HTTP status code
  status: 200,
  // HTTP status message from the server response
  statusText: 'OK',
  // `headers` the headers that the server responded with All header names are lower cased
  headers: {},
  // `config` is the config that was provided to `axios` for the request
  config: {}
}
```

4. axios 활용한 state 설정 가이드

Reactjs

컴포넌트의 생명주기 중 어디에서 AJAX 호출을 할 수 있나요?

AJAX 호출을 통한 데이터는 생명주기 메서드 중 `componentDidMount` 안에 추가되어야 합니다.

이는 데이터를 받아 올 때 `setState`를 통하여 컴포넌트를 업데이트하기 위함입니다.

```
{  
  "items": [  
    { "id": 1, "name": "Apples", "price": "$2" },  
    { "id": 2, "name": "Peaches", "price": "$5" }  
  ]  
}
```

```
class MyComponent extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = {  
      error: null,  
      isLoading: false,  
      items: []  
    };  
  }  
}
```

```
componentDidMount() {  
  fetch("https://api.example.com/items")  
    .then(res => res.json())  
    .then(  
      (result) => {  
        this.setState({  
          isLoading: true,  
          items: result.items  
        });  
        // 주의: 컴포넌트의 실제 버그에서  
        // 에러를 `catch()` 블록(block)에  
        // 이 부분에서 처리하는 것이 중요  
        (error) => {  
          this.setState({  
            isLoading: true,  
            error  
          });  
        }  
      }  
    )  
  }  
  
render() {  
  const { error, isLoading, items } = this.state;  
  if (error) {  
    return <div>Error: {error.message}</div>;  
  } else if (!isLoading) {  
    return <div>Loading...</div>;  
  } else {  
    return (  
      <ul>  
        {items.map(item => (  
          <li key={item.id}>  
            {item.name} {item.price}  
          </li>  
        ))}  
      </ul>  
    );  
  }  
}
```

<https://jsonplaceholder.typicode.com/>

The screenshot shows the homepage of the JSONPlaceholder API. The main content area features a large title '{JSON} Placeholder' and a subtitle 'Free fake API for testing and prototyping.' Below this, it states 'Powered by [JSON Server](#) + [LowDB](#)'. A note at the bottom indicates 'As of Dec 2020, serving ~1.8 billion requests each month.' To the right, a sidebar titled 'Resources' lists six common endpoints with their respective counts: /posts (100 posts), /comments (500 comments), /albums (100 albums), /photos (5000 photos), /todos (200 todos), and /users (10 users). The top navigation bar includes links for 'Guide', 'Sponsor this project', 'Blog', and 'My JSON Server'.

JSONPlaceholder

Guide Sponsor this project Blog My JSON Server

{JSON} Placeholder

Free fake API for testing and prototyping.

Powered by [JSON Server](#) + [LowDB](#)

As of Dec 2020, serving ~1.8 billion requests each month.

Resources

JSONPlaceholder comes with a set of 6 common resources:

/posts	100 posts
/comments	500 comments
/albums	100 albums
/photos	5000 photos
/todos	200 todos
/users	10 users

5. axios 실습

Reactjs

<https://jsonplaceholder.typicode.com/users>

```
jsonplaceholder.typicode.com/users

{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
},
{
  "id": 2,
  "name": "Ervin Howell",
  "username": "Antonette",
  "email": "Shanna@melissa.tv",
  "address": {
    "street": "Victor Plains",
    "suite": "Suite 879",
    "city": "Wisokyburgh",
    "zipcode": "90666-7771",
    "geo": {
      "lat": "-43.9509",
      "lng": "-34.4618"
    }
  },
  "phone": "010-692-6593 x09125",
  "website": "anastasia.net",
  "company": {
    "name": "Deckow-Crist",
    "catchPhrase": "Proactive didactic contingency",
    "bs": "synergize cross-media alternative ideas"
  }
},
```

```
constructor(props) {
  super(props);
  this.state = {
    error: null,
    isLoading: false,
    items: []
  }
}
```

```
componentDidMount() {
  axios.get("https://jsonplaceholder.typicode.com/users")
    .then((result) => {
      this.setState({
        ...this.state,
        isLoading: true,
        items: result.data
      })
    })
    .catch((error) => {
      this.setState({
        ...this.state,
        isLoading: true,
        error
      });
    });
}
```

```
render() {
  const { error, isLoading, items } = this.state;
  console.log("">>>>", items);
  if (error) {
    return <div>Error: {error.message}</div>;
  } else if (!isLoading) {
    return <div>Loading...</div>;
  } else {
    return (
      <ul>
        {items.map(item => (
          <li key={item.id}>
            {item.name} {item.email}
          </li>
        ))}
      </ul>
    );
  }
}
```

실행결과

localhost:3000

- Leanne Graham Sincere@april.biz
- Ervin Howell Shanna@melissa.tv
- Clementine Bauch Nathan@yesenia.net
- Patricia Lebsack Julianne.OConner@kory.org
- Chelsey Dietrich Lucio_Hettinger@annie.info
- Mrs. Dennis Schulist Karley_Dach@jasper.info
- Kurtis Weissnat Telly.Hoeger@billy.biz
- Nicholas Runolfsdottir V Sherwood@rosamond.me
- Glenna Reichert Chaim_McDermott@dana.io
- Clementina DuBuque Rey.Padberg@karina.biz

6. 로컬에 Json 서버 설치 및 활용

Reactjs

1) node.js 기반의 json 서버 설치

```
npm install json-server --save
```

2) my-app 프로젝트의 db.json 파일 수정



```
{}
db.json u x
my-app > {} db.json > [ ] todos
18 },
19 "todos": [
20 {
21   "id": 1,
22   "title": "Read SitePoint article",
23   "complete": false
24 },
25 {
26   "id": 2,
27   "title": "Clean inbox",
28   "complete": false
29 },
30 {
31   "id": 3,
32   "title": "Make restaurant reservation",
33   "complete": false
34 }
35 ]
36 }
```

3) package.json 파일 수정

```
"scripts": {  
  "start": "react-scripts start",  
  "build": "react-scripts build",  
  "test": "react-scripts test",  
  "eject": "react-scripts eject",  
  "json-server": "json-server --port 3001 --watch db.json"  
},
```

4) Json 서버 실행 및 요청

npm run json-server

```
Resources  
http://localhost:3001/posts  
http://localhost:3001/comments  
http://localhost:3001/profile  
http://localhost:3001/todos
```

Home

http://localhost:3001

실행결과

← → ⌂ i localhost:3000

- 1 Read SitePoint article false
- 2 Clean inbox false
- 3 Make restaurant reservation false

11장. 라우팅



SPA vs MPA

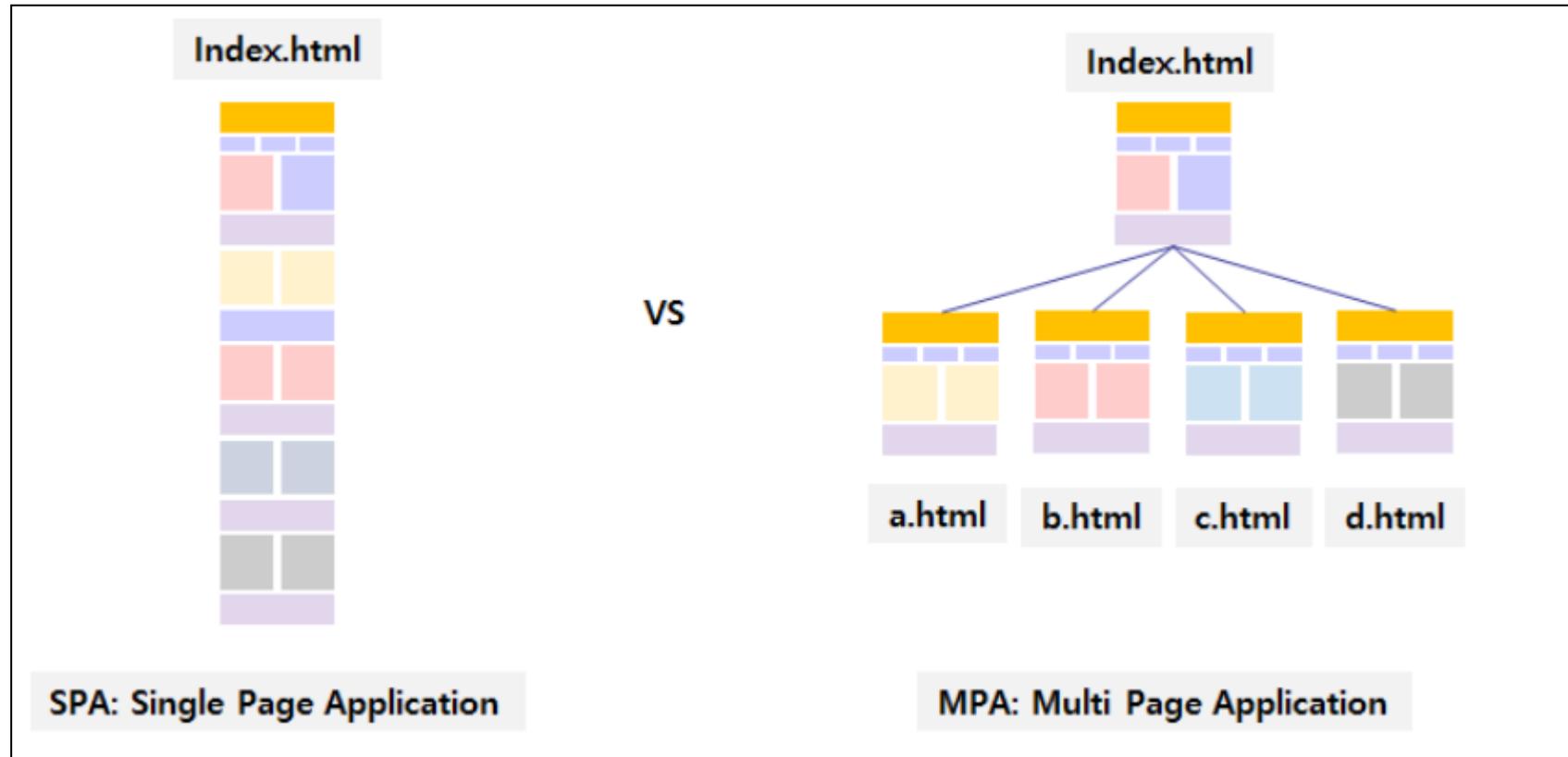
SPA 특징

React Router

1. SPA vs MPA

Reactjs

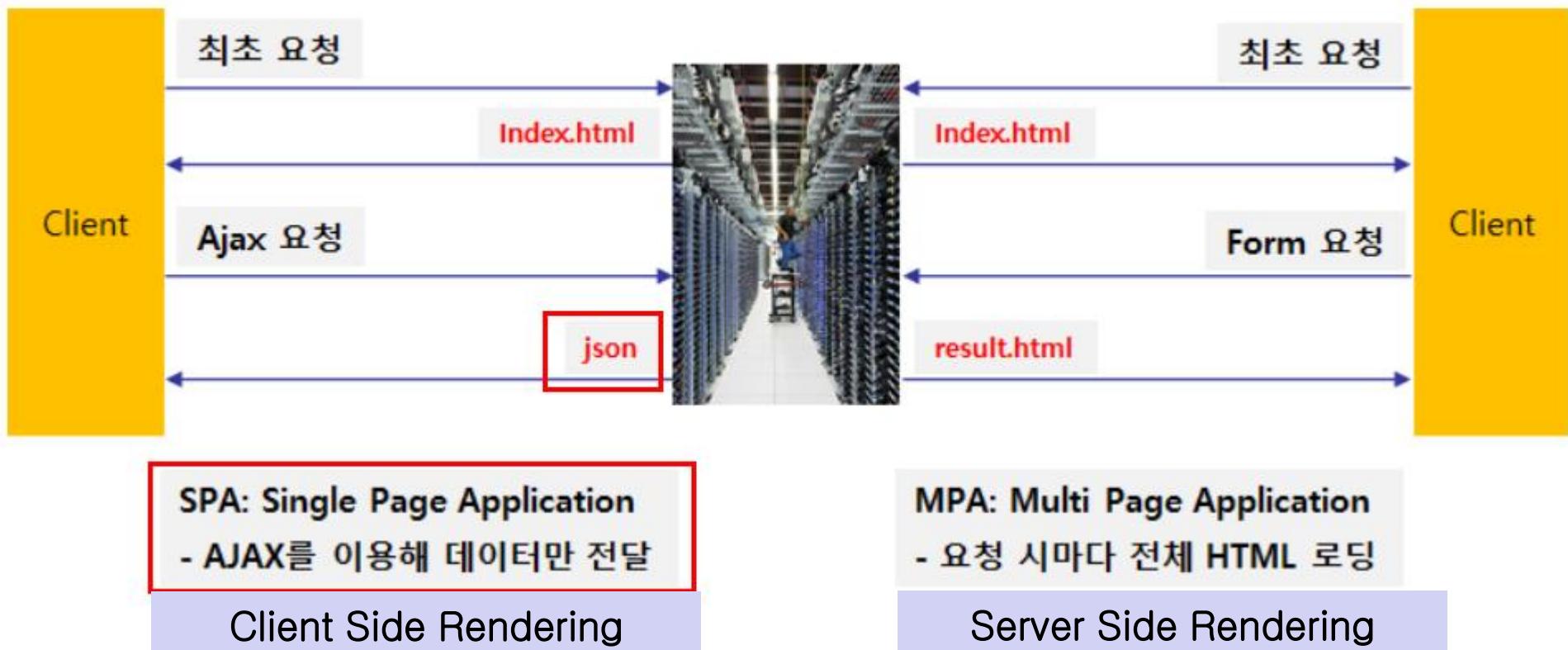
웹 어플리케이션의 UI 화면을 구현할 때 사용 가능한 형태는 SPA와 MPA가 있다.



1. SPA vs MPA

Reactjs

SPA와 MPA의 동작 방식의 가장 큰 차이점은 일반적인 Form 전송이냐 Ajax 동작이냐로 나눠볼 수 있다.



SPA는 Single Page Application으로 Client Side Rendering을 추구한다.

즉 UI화면과 관련된 리소스를 처음 요청시 서버로부터 몽땅 받아낸 후 클라이언트에서 모든 HTML/CSS/JS를 가지고 있다.

이후 Ajax 통신을 통해 변경하고자 하는 데이터만 받아오게 된다.

장점

- SPA는 사용 중 리소스 로딩이 없기 때문에 부드럽게 화면 전환이 이루어진다.
- 서버 입장에서는 템플릿(JSP)를 만드는 연산이 클라이언트로 분산되기 때문에 부담이 줄어든다.
- 컴포넌트별로 개발하기 때문에 생산성이 향상된다.
- 모바일 앱에서도 동일한 패턴의 Rest API 사용이 가능하다.

단점

- URL이 변경되지 않기 때문에 검색엔진의 색인화가 어렵다.
- 초기 구동 비용이 MPA 대비 상대적으로 비싸다.

3. react-router <https://reactrouter.com/web/api/Route>

Reactjs

1) react-router 설치

npm install react-router-dom

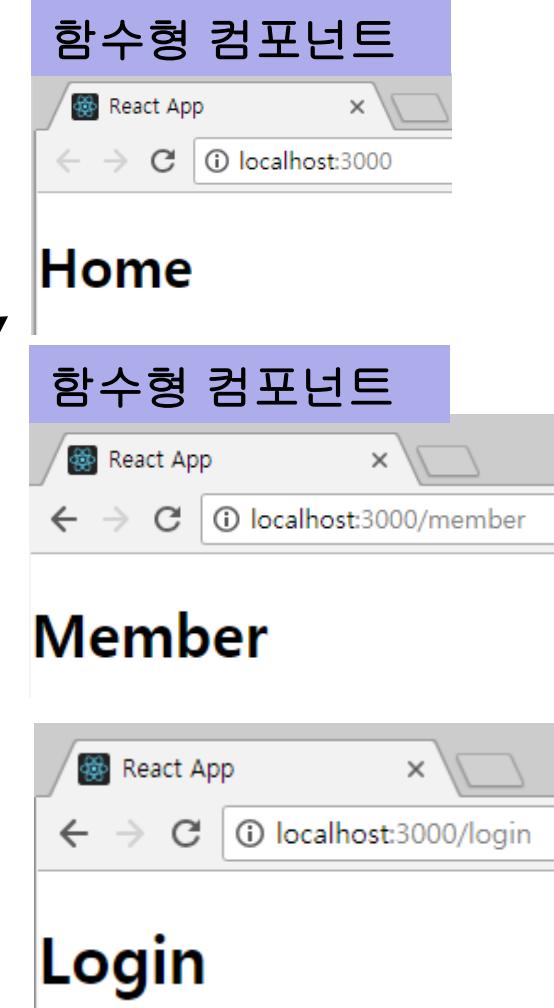
2) App.js에서 route 설정

```
import React, { Component } from 'react';
import './App.css';

import { BrowserRouter as Router, Route} from 'react-router-dom';
import Home from './Home';
import Member from './Member';
import Login from './Login';

class App extends Component {
  render() {
    return (
      <div>
        <Router>
          <div>
            <Route exact path="/" component={Home} />
            <Route exact path="/member" component={Member} />
            <Route exact path="/login" component={Login} />
          </div>
        </Router>
      </div>
    );
  }
}
```

- 77 -



3. react-router

Reactjs

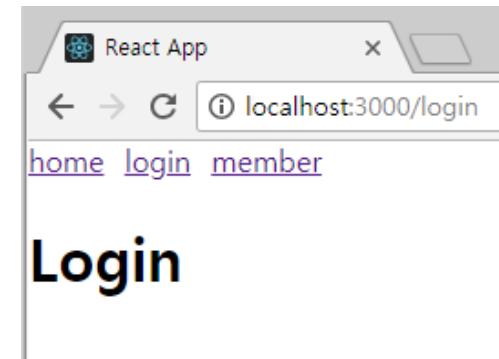
3) 링크 설정

```
import React, { Component } from 'react';
import {NavLink} from 'react-router-dom';

class Navi extends Component {
  render() {
    return (
      <div>
        <div>
          <NavLink to="/">home</NavLink>
          <NavLink to="/login">login</NavLink>
          <NavLink to="/member">member</NavLink>
        </div>
      </div>
    );
  }
}

export default Navi;
```

```
class App extends Component {
  render() {
    return (
      <div>
        <Router>
          <div>
            <Navi />
            <Route exact path="/" component={Home}></Route>
            <Route exact path="/member" component={Member}></Route>
            <Route exact path="/login" component={Login}></Route>
          </div>
        </Router>
      </div>
    );
  }
}
```



4) 파라미터 전달

- 대표적인 파라미터 전달 방법

가. queryString 방식

`http://localhost:3000/member?userid=hong`

나. Restful 방식 (*)

`http://localhost:3000/member/hong`

3. react-router

Reactjs

라우트로 설정한 컴포넌트는, 3가지의 props 를 전달받게 됩니다:

- `history` 이 객체를 통해 `push`, `replace` 를 통해 다른 경로로 이동하거나 앞 뒤 페이지로 전환 할 수 있습니다.
- `location` 이 객체는 현재 경로에 대한 정보를 지니고 있고 URL 쿼리 (`/about?foo=bar` 형식) 정보도 가지고 있습니다.
- `match` 이 객체에는 어떤 라우트에 매칭이 되었는지에 대한 정보가 있고 params (`/about/:name` 형식) 정보를 가지고 있습니다.

```
class Member extends Component {  
  render() {  
    console.log(this.props.match);  
    console.log(this.props.history);  
    console.log(this.props.location);  
    return (  
      <div>  
        <h1>Member</h1>  
      </div>  
    );  
  }  
}
```

```
const Member =({match,history,location})=>{  
  console.log(match);  
  console.log(history);  
  console.log(location);  
  return (  
    <div>  
      <h1>Member</h1>  
    </div>  
  );  
};
```

3. react-router

Reactjs

```
▼ {history: {...}, location: {...}, match: {...}, staticContext: undefined} ⓘ
  ▼ history:
    action: "PUSH"
    ► block: f block(prompt)
    ► createHref: f createHref(location)
    ► go: f go(n)
    ► goBack: f goBack()
    ► goForward: f goForward()
    length: 14
    ► listen: f Listen(listener)
    ► location: {pathname: "/member/hong", search: "", hash: "", state: null, key: "fxkjb6"}
    ► push: f push(path, state)
    ► replace: f replace(path, state)
    ► __proto__: Object
  ▼ location:
    hash: ""
    key: "fxkjb6"
    pathname: "/member/hong"
    search: ""
    state: null
    ► __proto__: Object
  ▼ match:
    isExact: true
    ► params: {userid: "hong"}
    path: "/member/:userid"
    url: "/member/hong"
    ► __proto__: Object
    staticContext: undefined
    ► __proto__: Object
```

3. react-router

Reactjs

5) 중첩 라우팅

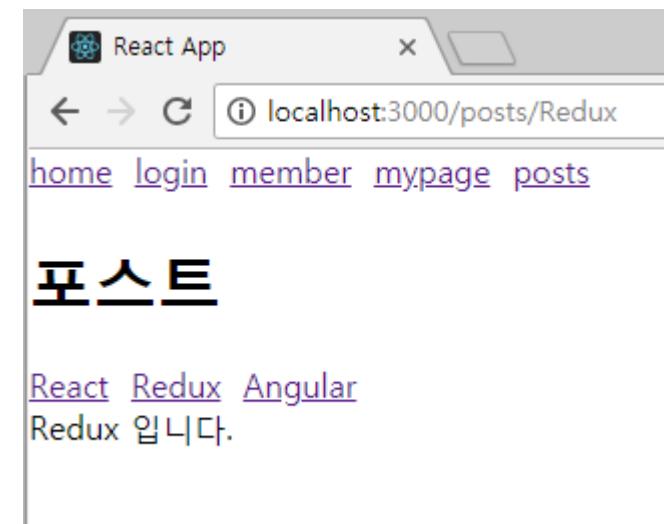
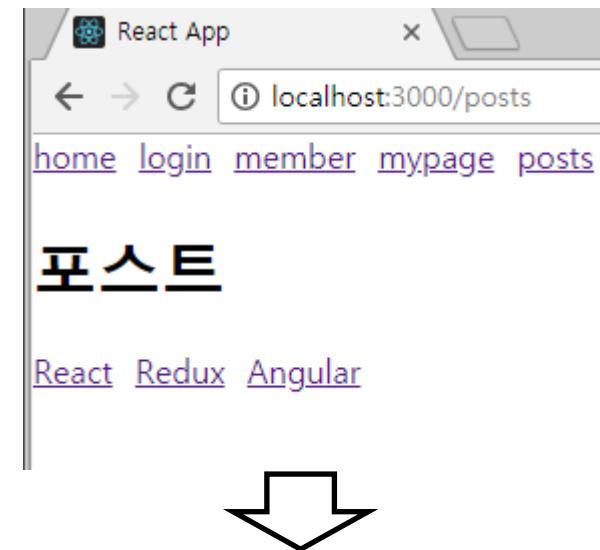
```
import React from 'react';

//중첩 라우터
import {Route,Link} from 'react-router-dom';

const Post = ({match}) => {
  return (
    <div>
      &{match.params.title} 입니다.
    </div>
  );
};

const Posts = () => {
  return (
    <div>
      <h1>포스트</h1>
      <Link to="/posts/React">React</Link>
      <Link to="/posts/Redux">Redux</Link>
      <Link to="/posts/Angular">Angular</Link>
      <Route path="/posts/:title" component={Post} />
    </div>
  );
};

export default Posts;
```



수고하셨습니다.