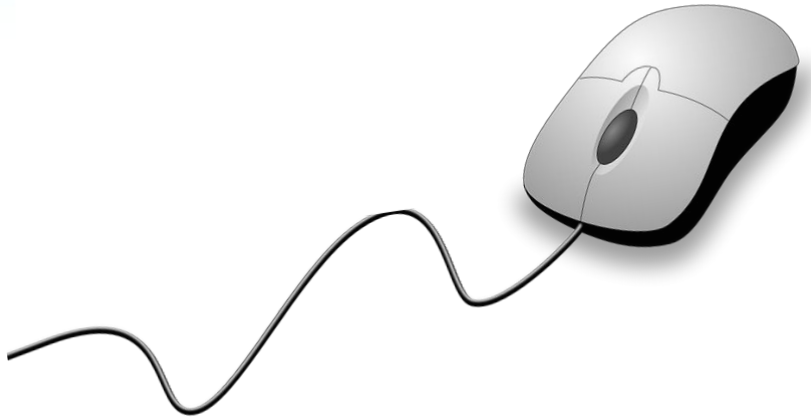
공개SW 솔루션 설치 & 활용
가이드



React 제대로 배워보자

How to Use Open Source Software

Open Source Software Installation & Application Guide

OPENUP
오픈소스 소프트웨어 통합지원센터
Open Source Software Support Center

CONTENTS

1. 개요

소개	• 사용자 인터페이스를 만들기 위한 진보적인 자바스크립트 프론트엔드 프레임워크 • 생태계가 넓고 사용하는 곳이 많음 • HTTP 클라이언트, 라우터, 심화적 상태 관리등의 기능들은 내장되어있지 않음, 따로 공식 라이브러리가 있는 것도 아니여서, 개발자가 원하는 스택을 마음대로 골라서 사용할 수 있음		
주요기능	• 리액트는 컴포넌트 기반의 라이브러이다. • Virtual Dom • JavaScript에 XML을 추가 확장한 문법인 JSX사용		
대분류	• 응용 SW	**소분류**	• 콘텐츠 배포
라이선스형태	• The MIT License (MIT)	**사전설치 솔루션**	• Node.js(NPM을 사용할 경우)
		버전	• v18.2.0(2022년 6월 14일 기준)
특징	• Component 기반으로 이루어짐 • Virtual DOM 기반을 사용하여 가벼움 • 단방향 데이터 바인딩으로 리액트는 데이터 흐름이 한방향으로만 흐름 • 다른 framework나 라이브러리와 병행해서 사용할 수 있음. 이는 개발이 이미 완료된 프로젝트에도 적절히 녹여낼 수 있는 확장성도 포함함. • 공식문서의 품질이 좋음		
개발회사/커뮤니티	• Meta Platforms / https://www.facebook.com/groups/react.ko		
공식 홈페이지	• https://ko.reactjs.org/		

2. 기능요약

React.js 주요 기능

컴포넌트	MVC의 뷰를 독립적으로 구성하여 재사용을 할 수 있고 이를 통해 새로운 컴포넌트를 쉽게 만들 수 있음.
단방향 바인딩	단방향 데이터 바인딩은 단 하나의 Watcher가 자바스크립트의 데이터 갱신을 감지하여 사용자의 UI 데이터를 갱신함. 사용자가 UI를 통해 자바스크립트의 데이터를 갱신할 때는, 이벤트를 통해 갱신하게 됨. 이처럼 단방향 데이터 바인딩은 하나의 Watcher를 사용하기 때문에 양방향 데이터 바인딩이 가지는 성능적인 이슈를 해결하고 더 확실하게 데이터를 추적할 수 있게 해줌.
JSX	JSX는 자바스크립트와 HTML을 동시에 사용하며, HTML에 자바스크립트의 변수들을 바로 사용할 수 있는 일종의 템플릿 언어(Template language)이다. 자바스크립트에서 HTML 태그를 사용할 수 있으며, 자바스크립트 변수를 HTML 태그에서 바로 호출하여 사용할 수 있음.

3. 실행환경

React 를 사용하는 개발환경 (Node.js 혹은 yarn 을 설치해야함)

- Windows
- Linux
- maxos등

React.js 지원 브라우저

- Safari 6 이상
- Chrome 23 이상
- Firefox 21 이상
- Edge 12 이상
- IE 9 이상

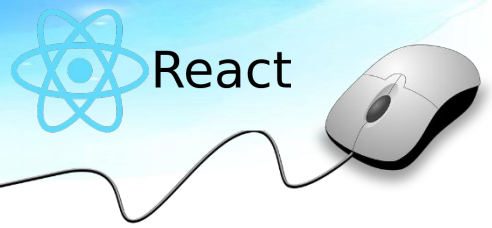
4. 설치 및 실행

1. Node.js 설치
2. React 설치 및 프로젝트 생성

본 가이드에서는 코드 에디터로 Vs Code를 사용하지만, 다른 텍스트 에디터를 사용해도 코드를 보고 편집할 수 있습니다. VS Code 다운로드 : https://code.visualstudio.com/download

4. 설치 및 실행

4.1. Node.js 설치 및 실행 (1 / 2)



- OS version: Windows 10
- Node.js version: v18.10.0
- React 를 이용하려면 Node.js가 설치 되어있어야함.
- Windows의 경우 https://nodejs.org/ko/download/ 에서 Node.js 설치
- Ubuntu의 경우 아래 순서대로 bash에서 진행
  - `sudo apt-get install curl`
  - `curl -fsSL https://deb.nodesource.com/setup_current.x | sudo -E bash -`
  - `sudo apt-get install -y nodejs`
- Node.js와 NPM 버전 확인 (Windows의 경우 cmd / Linux의 경우 bash)
  - node -v
  - npm -v
  - 버전의 숫자가 나오면 잘 설치된 것

```
soyun@soyun-Latitude-3520:/$ node -v
v18.10.0
soyun@soyun-Latitude-3520:/$ npm -v
8.19.2
```

4. 설치 및 실행

4.2. React 설치 및 프로젝트 생성(2 / 2)

1. React 설치

   `npm install -g create-react-app`

2. 프로젝트 생성

   `create-react-app “프로젝트 명”`

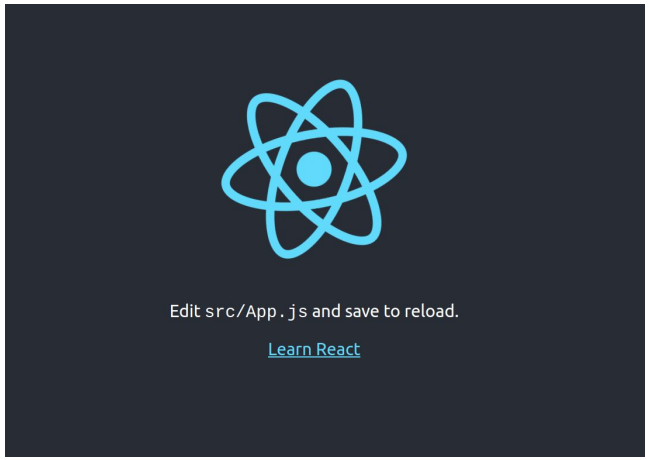
```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
```

3. vscode를 실행한 후 package.json의 scripts 확인

4. package.json에 있는 명령어를 사용하여 프로젝트를 로컬에서 실행

   `npm start`

5. React App 이 자동으로 실행



5. 기능소개

세부 목차

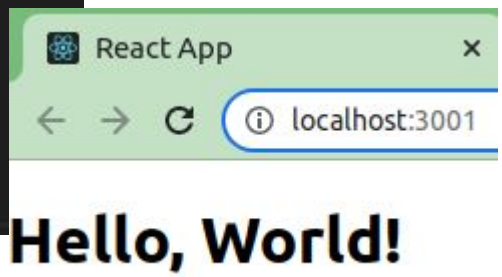
1. JSX
2. 컴포넌트
3. props
4. 라이프사이클
5. Hook

5.1. JSX란?(1 / 2)

- JSX란.
  - JavaScript에서 HTML 형식을 그대로 사용할 수 있습니다.
  - React element를 생성합니다. React element는 브라우저 DOM 엘리먼트와 달리 일반 객체입니다.
  - JS 코드 내부에서 UI관련 작업이 가능하므로 시각적으로 더 도움이 됩니다.
  - React가 더욱 도움이 되는 에러 및 경고 메시지를 표시할 수 있게 해줍니다.

```jsx
//JSX 예시
function App() {
  return (
    <h1>Hello, World!</h1>
  );
}

export default App;
```

JSX 문법의 예시

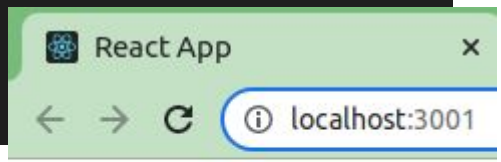


출력결과

- JSX문법
  - 반드시 부모 요소 하나가 감싸는 형태여야한다.
  - JSX내에서 자바스크립트 표현식을 사용할 수 있다. 자바스크립트 표현식을 작성하기 위해서 JSX 내부에서 코드를 {}로 감싸주면된다.
  - if문과 for문 대신 삼항연산자 사용해야한다.
  - React DOM은 HTML 어트리뷰트 이름 대신 camelCase 프로퍼티 명명 규칙을

```
4  //JSX 문법예시
5  function App() {
6    const name = 'react'
7    return (
8      <div className='react'>
9        {name == 'react'? (<h1>Hello, {name}</h1>) :  (<h1>Bye, {name}</h1>)}
10     </div>
11
12     );
13 }
```

JSX문법 예시



Hello, World!

출력결과

- 컴포넌트
  - React로 만들어진 앱을 이루는 최소한의 단위
  - MVC의 뷰를 독립적으로 구성하여 재사용 할 수 있으며 새로운 컴포넌트를 쉽게 만들 수 있다.
  - 데이터(props)를 입력받아 view(state) 상태에 따라 DOM node를 출력하는 함수
  - 컴포넌트 이름은 항상 대문자로 시작
  - UI를 재사용 가능한 개별적인 여러 조각으로 나누고, 각 조각을 개별적으로 나눈다.
  - 컴포넌트의 종류에는 함수형 컴포넌트와 클래스형 컴포넌트가 존재한다.
    - 리액트에서 정의하는 컴포넌트의 종류 :

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

- 함수형 컴포넌트
  - 가장 간단하게 컴포넌트를 정의하는 방법은 자바스크립트 함수를 이용하는 것이다.
  - export 구문은 작성한 Main 파일을 다른 파일에서 import할때 Main으로 불러올 수 있도록 정의해주는 부분이다.
  - import할때 js, jsx 와 같은 확장자를 생략해도 자동으로 찾을 수 있다.

```
import React from 'react';

function Main(props){
    return(
        <div>
            <main>
                <h1>안녕하세요. {props.name}입니다.</h1>
            </main>
        </div>
    )
}
export default Main;
```

함수형 컴포넌트의 예시

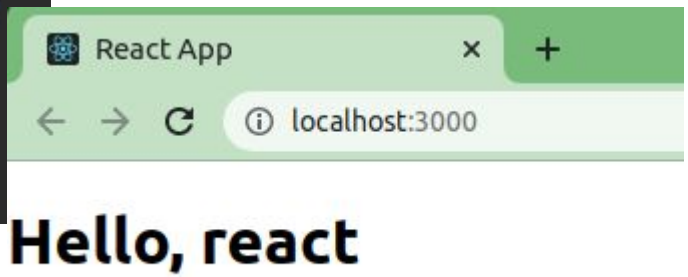
5. 기능소개

5.2. 컴포넌트 종류 (3 / 3)

- 클래스형 컴포넌트
  - 컴포넌트 구성요소, 리액트 생명주기를 모두 포함하고 있다.
  - 프로퍼티,state,생명주기 함수가 필요한 구조의 컴포넌트를 만들 때 사용한다.
  - render함수가 무조건 존재해야하며, 내부에서 보여주어야 할 jsx를 반환한다.

```
import React, {Component} from 'react';

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

클래스형 컴포넌트의 예시



출력 결과

5. 기능소개

5.3. props(1 / 2)

- 프로퍼티,props(properties의 줄임말) 이라고 한다.
- props는 상위 컴포넌트에서 하위컴포넌트로 데이터를 내려주는 방법이다.
- 상위 컴포넌트가 하위 컴포넌트에 값을 전달할때 사용한다. (단방향 데이터 흐름을 가지고 있음)
- HTML의 속성 문법과 같은 형태로 하위에 전달
  - `<Main name= “react”/>`
- 프로퍼티는 수정할 수 없다는 특징이 있다.

```jsx
import React from 'react';

function Main(props){
    return(
        <div>
            <main>
                <h1>안녕하세요. {props.name}입니다.</h1>
            </main>
        </div>
    )
}
export default Main;
```

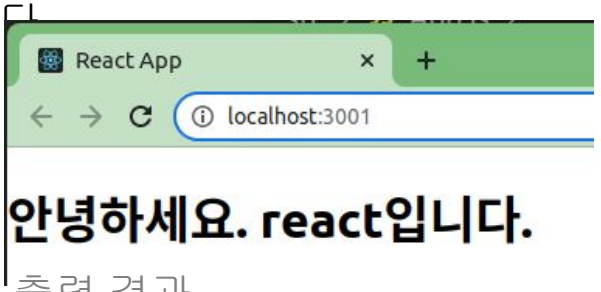
props문법 예시 - Main.js

```jsx
import './App.css';
import Main from './main';

//props 예시
function App() {
  return (
    <div>
      <Main name="react"/>
    </div>
    );
}

export default App;
```

props문법 예시 - App.js

5.3. props(2 / 2)

- 프로퍼티에 문자열을 전달할 때는 큰따옴표(" ")를, 문자열 외의 값을 전달할 때는 중괄호({ }) 를 사용 한다.
- 컴포넌트에게 전달되는 props는 파라미터를 통해 조회할 수 있다.
- props는 객체 형태로 전달 되며 이를 조회하고 싶다면 `{props.name}`을 사용하여 조회하면된다.



출력 결과

```jsx
import React from 'react';

function Main(props){
    return(
        <div>
            <main>
                <h1>안녕하세요. {props.name}입니다.</h1>
            </main>
        </div>
    )
}
export default Main;
```

props문법 예시 - Main.js

```jsx
import './App.css';
import Main from './main';

//props 예시
function App() {
  return (
    <div>
      <Main name="react"/>
    </div>
    );
}

export default App;
```

props문법 예시 - App.js

- 라이프사이클
  - React는 컴포넌트 단위로 개발하며, 이때 각 컴포넌트들은 라이플사이클을 가지고 있다.
  - 클래스형 컴포넌트에서는 주로 생명주기 메서드를 통해 라이프사이클에 따라 컴포넌트를 조작한다.
  - 함수형 컴포넌트에서는 생명주기 메서드가 따로 존재하지 않기 때문에 리액트 훅을 사용하여 생명주기 메서드와 비슷하게 동작하도록 구현한다.
  - 라이프사이클을 생성, 업데이트, 제거 총 세 가지 단계로 나뉘며 각 단계마다 메서드가 존재한다.
  - 총 9가지 메서드가 있으며 자세한 라이프사이클은 다음을 참조하도록한다.
    https://github.com/wojtekmaj/react-lifecycle-methods-diagram

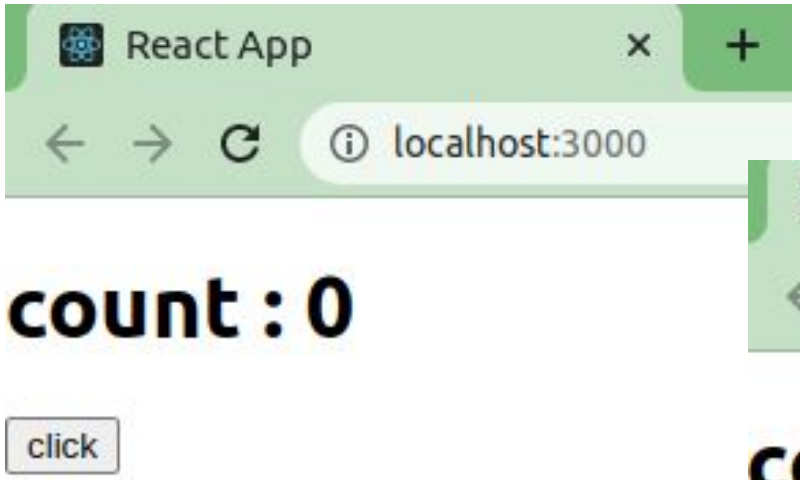
- React의 Hook 은 함수형 컴포넌트에서 React state와 생명주기 기능을 “연동(hook into)“할 수 있게 해주는 함수이다.
- Hook은 class 안에서는 동작하지 않고, class 없이 React를 사용할 수 있게 해준다.
- 최상위에서만 Hook을 호출해야한다.
  - 반복문이나 조건문 혹은 중첩된 함수 내에서 Hook을 호출하면 안된다.
  - React 혹은 호출되는 순서에 의존하기에 조건문이나 반복문 안에서 실행하게 될 경우 해당 부분을 건너뛰는 일이 발생할 수도 있기에 순서가 꼬여 버그발생 가능성이 높아진다.
- React 함수 컴포넌트에서만 Hook을 호출해야하고, 일반 JS함수에서는 Hook을 호출해서는 안된다.
  - hook은 일반적인 js 함수에서는 호출하면 안된다.

React Hook의 종류는 다양하며 자세한건 해당 링크에서 확인할 수 있다.

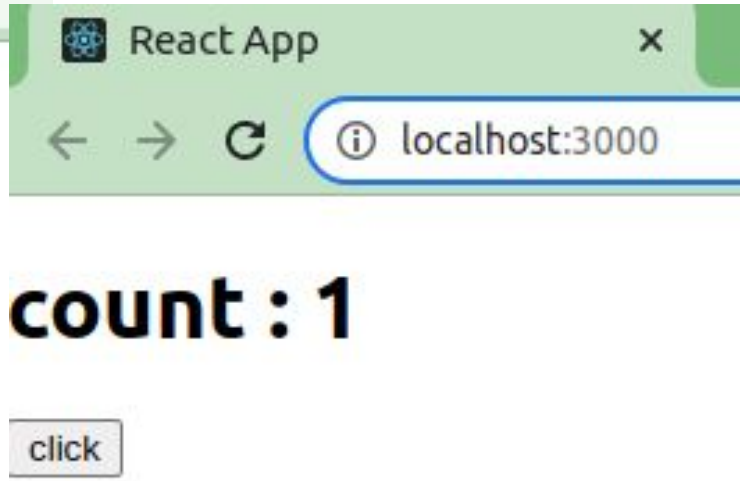
https://ko.reactjs.org/docs/hooks-intro.html

6. 버튼을 추가하여 이벤트 핸들링을 추가

- 출력 결과 예시



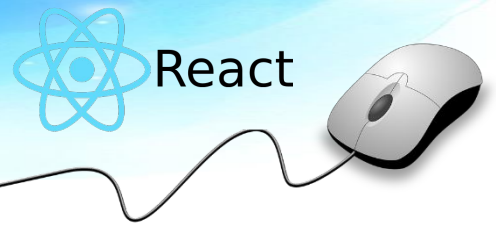
출력 결과



Click 버튼을 클릭하였을 때

6. 활용예제

세부 목차



6. 버튼을 추가하여 이벤트 핸들링을 추가

- 템플릿에 버튼과 Count 변수를 나타내는 문장을 추가
- 생성자로 변수만들기

  (값을 증가시키기 위해 0으로 초기값을 설정하는 과정)
- 버튼을 클릭하면 증가하는 함수 만들기
- button에 추가한 onClick={onClick}으로 버튼을 클릭할때마다 onClick 함수가

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

//Button click
function App() {
  const [counter,setCounter] = React.useState(0);
  function onClick(){
    setCounter(counter+1);
  }
  return (
    <div>
      <h1>count : {counter}</h1>
      <button onClick={onClick}>click</button>
    </div>

  );
}
export default App;
```