React 설치

Node.js 설치

https://nodejs.org/en/download

React 프로젝트 생성 명령

npx create-react-app (프로젝트명)

```
npx create-react-app ex01

#설치가 안될 경우
npm -g i npx create-react-app
```

터미널 창에서 위 명령을 실행하면, 현재 디렉토리에 프로젝트 폴더가 생성되고 프로젝트 파일이 생성된다.

npx는 javascript로 구현된 명령을 다운로드하여 실행해 주는 도구이다.
위 명령에 의해서 create-react-app 패키지 최신 버전이 다운로드 되고 실행된다.

따라서 위 명령을 처음 실행하면, create-react-app 패키지를 설치할 것인지 묻는다.

프로젝트 경로로 이동

```
cd ex01
```

프로젝트 구조

package.json 파일

모든 자바스크립트 프로젝트의 기본 설정 파일은 package.json 이다.
자바스크립트 프로젝트의 루트 폴더에 이 파일이 있어야 한다.

이 파일의 내용은 다음과 같다.

name

프로젝트 이름

version

프로젝트 버전

dependencies

프로젝트가 사용하는 패키지(라이브러리) 이름과 버전 목록.

dependency 패키지들은 node_modules 폴더 아래에 다운로드 된다.
따라서 node_modules 폴더의 크기는 상당하다.

프로젝트 폴더를 압축할 때, 먼저 node_modules 폴더를 삭제해야 한다.
다음 명령에 의해서 dependency 패키지들을 다시 전부 다운로드하여 설치할 수 있기 때문이다.

```
npm install
```

위 명령을 아래처럼 간단히 입력해도 된다.

```
npm i
```

scripts

프로젝트를 빌드하거나 실행하기 위한 명령 목록이다.

npm start : 개발하기 위해 프로젝트 실행하기 위한 명령

npm run build : 프로젝트를 빌드하여 배포 파일 생성하기 위한 명령

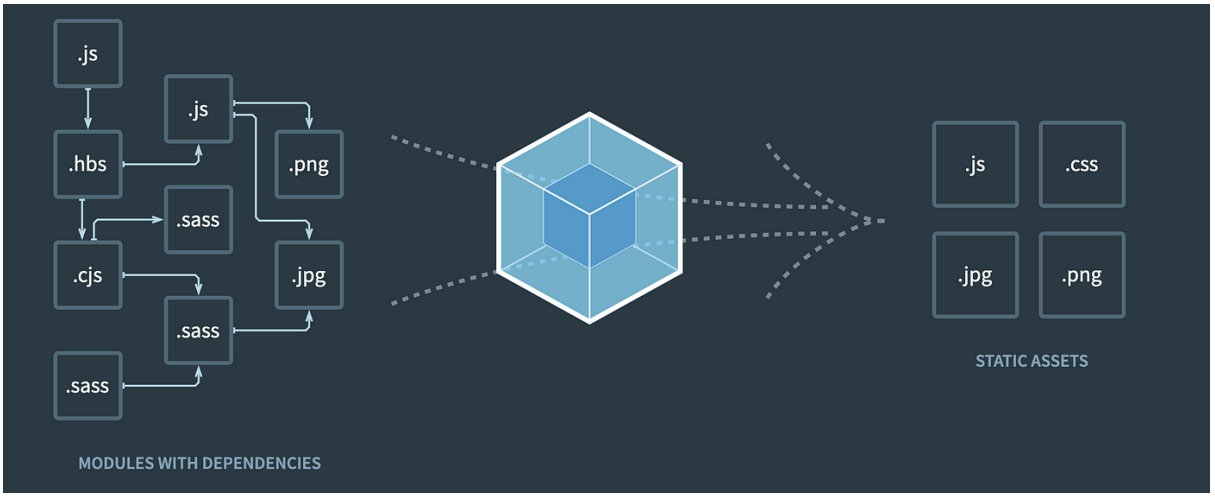
웹팩 (webpack)

자바 스크립트 프로젝트의 가장 대표적인 빌드 도구가 웹팩이다.
웹팩은 프로젝트의 소스코드 파일들을 기본적인 자바스크립트 문법으로 트랜스파일하고, 트랜스파일된 파일들을 묶어서 빌드 결과 파일 즉 배포 파일을 만들어 준다.

리액트로 개발된 프론트엔드 앱은 웹 브라우저에서 실행되어야 하므로,
프로젝트 빌드 결과 만들어진 배포 파일이 웹브라우저에 다운로드되어 실행된다.

이 빌드 과정을 번들(bundle) 이라고도 한다.
번들은 묶음, 묶음을 만든다는 뜻인데,
많은 소스코드 파일들을 묶어서 소수의 배포 파일을 만드는 것을 의미한다.

그래서 웹팩을 번들러(bundler), 프로젝트 빌드 결과물을 번들(bundle) 이라고도 부른다.



빌드 해보기

다음 명령을 실행하면 웹팩에 의해 프로젝트가 빌드된다

```
npm run build
```

위 명령을 실행하면 프로젝트 루트 디렉토리 아래 build 디렉토리에 빌드 파일들이 생성된다.
프로젝트 개발이 끝나고 배포할 파일을 만들 때 위 명령을 실행하면 된다.

```
Compiled successfully.

File sizes after gzip:

  46.61 kB  build\static\js\main.4c740342.js
  1.77 kB   build\static\js\787.ea22f087.chunk.js
  541 B     build\static\css\main.073c9b0a.css

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
  serve -s build

Find out more about deployment here:

  https://cra.link/deployment
```

노란색 부분: 빌드 결과 만들어진 배포 파일들
빨간색 부분: 배포 파일들을 node.js에서 실행하기 위한 명령

React 프로젝트 실행

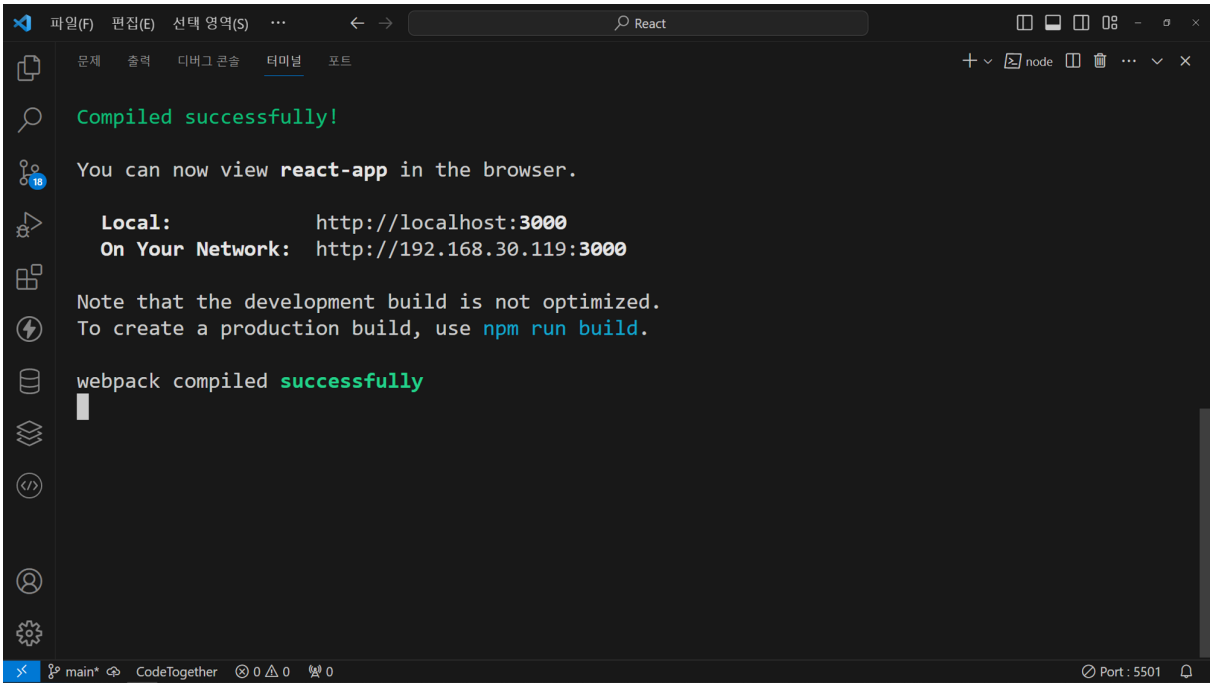
프로젝트 실행

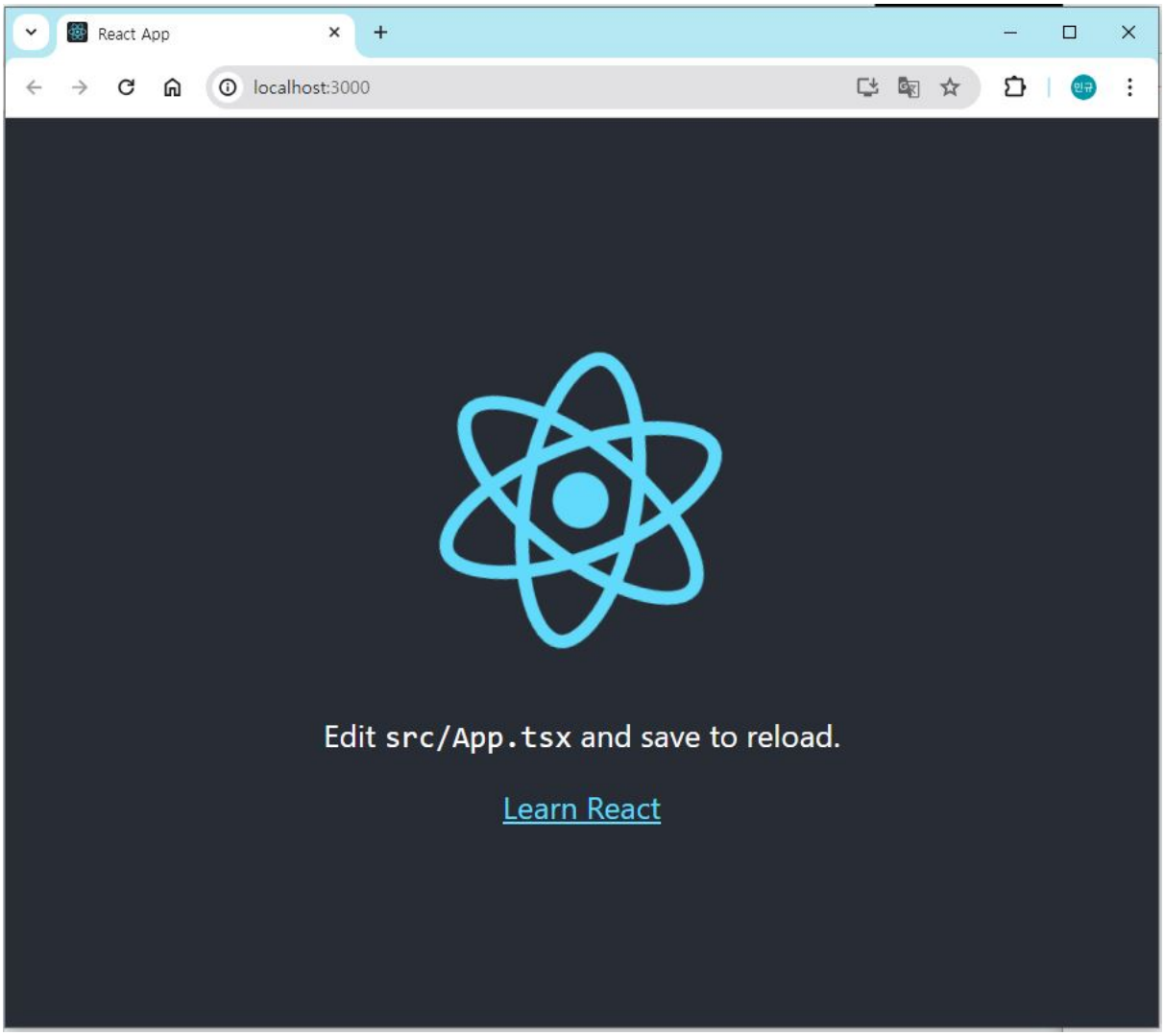
개발 모드로 프로젝트를 실행하려면, 빌드할 필요 없이 그냥 다음 명령만 실행하면 충분하다.
다음 명령에 의해서 소스코드 파일들이 번들되어 바로 실행된다.

```
npm start
```

위 명령에 의해서 구동되는 것은 웹팩 웹서버이다.

그림 1-25 개발 모드로 실행할 때의 웹팩





강제 종료

npm start 명령은 저절로 종료되지 않고 계속 실행된다.
따라서 실행을 종료하려면, 터미널에서 Ctrl+C 키를 눌러 강제 종료하거나, 터미널 창을 닫아야 한다.

그리고 npm start 명령이 실행 중인 터미널 창에서 다른 명령을 실행할 수 없기 때문에, 터미널 창을 하나 더 열어야 한다.

npm start 명령이 실행중인 터미널 창

빨간색 버튼을 클릭하면, 새 터미널 창이 열린다.
파란색 버튼을 클릭하면 현재 터미널 창이 닫히고 여기서 실행중인 명령도 강제 종료된다.

주요 소스 파일

1. index.html
2. index.jsx
3. App.jsx

Single Page App

리액트 앱을 포함하여 프론트엔드 앱들은 싱글 페이지 앱이다. (Single Page App)
하나의 웹 페이지 내에서만 실행된다는 뜻이다.
그 웹페이지를 벗어나면 프론트엔드 앱은 종료된다.
그 웹 페이지를 새로고침하면 프론트엔드 앱은 재시작 하게 된다.

html 웹페이지에 구현된 자바스크립트 코드가 그 웹페이지를 벗어나면 종료되기 때문이다.

진입점

`create-react-app`에서 React 앱의 진입점(entry point)은 `src/index.js` 로 설정되어 있습니다. 이것은 `create-react-app` 이 제공하는 기본 구조에서의 규칙 중 하나입니다.

일반적으로 `package.json` 파일 내부에서 `"main"` 필드로 진입점을 지정할 수 있습니다. 하지만 `create-react-app` 에서는 대부분의 설정이 내부적으로 이루어지기 때문에 개발자가 직접적으로 `"main"` 필드를 설정할 필요가 없습니다.

실행 순서

1. **public/index.html**
2. **src/index.jsx**
   a.
   App.jsx 컴포넌트 생성
3. **JSX ⇒ JavaScript 로 변환, DOM 에 렌더링**

public/index.html

1️⃣ 브라우저에서 `public/index.html` 파일이 열리면, 그 안에 있는 `<div id="root"></div>` 는 React 앱이 렌더링될 장소를 제공합니다.

src/index.jsx

2️⃣ 그 후, `index.jsx` 파일이 실행됩니다. `index.jsx` 에서는 `ReactDOM.createRoot()` 를 사용하여 `App` 컴포넌트를 `<div id="root"></div>` 에 렌더링합니다.

3️⃣ React는 이후에 App.js 파일에서 정의된 컴포넌트를 생성하고, JSX를 JavaScript로 변환하여 해당 DOM 요소에 렌더링합니다.

랜더링(rendering)

리액트 앱에서 생성한 화면은 Html 태그들로 구성되어 있다
그 html 태그들을 웹 페이지에 보여지도록 하는 작업을 렌더링이라고 부른다.

프로젝트 시작하기

HTML 파일 작성

public/index.html

이 프로젝트에 구현된 리액트 앱은, index.html 페이지 내에서 실행된다.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, init
ial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo19
2.png" />
    <title>React App</title>
  </head>
<body>
    <noscript>You need to enable JavaScript to run this ap
p.</noscript>
  <!-- React 앱을 표시할 컨테이너 - 리액트 앱의 화면은 이 div 태그
내부에 출력된다.-->
  <div id="root"></div>
```

```
</body>
</html>
```

메인 화면 구현

index.jsx

```
//React 라이브러리 패키지와 ReactDOM 라이브러리 패키지를 import 한다
import React from 'react';
import ReactDOM from 'react-dom/client';

//index.css 파일의 CSS 서식을 적용한다.
import './index.css';

//App.tsx 파일에 구현된 App 컴포넌트를 import 한다. 리액트 앱의 구성 요소는 컴포넌트이다.
import App from './App';

//reportWebVitals 라이브러리 패키지를 import 한다. 리액트 앱의 성능을 측정하기 위한 패키지이다.
import reportWebVitals from './reportWebVitals';

//아래 코드는 다음과 같다.
//<div id="root"></div> 태그에 App 컴포넌트를 렌더링 한다.
const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

reportWebVitals();
```

App 컴포넌트 작성

App.jsx

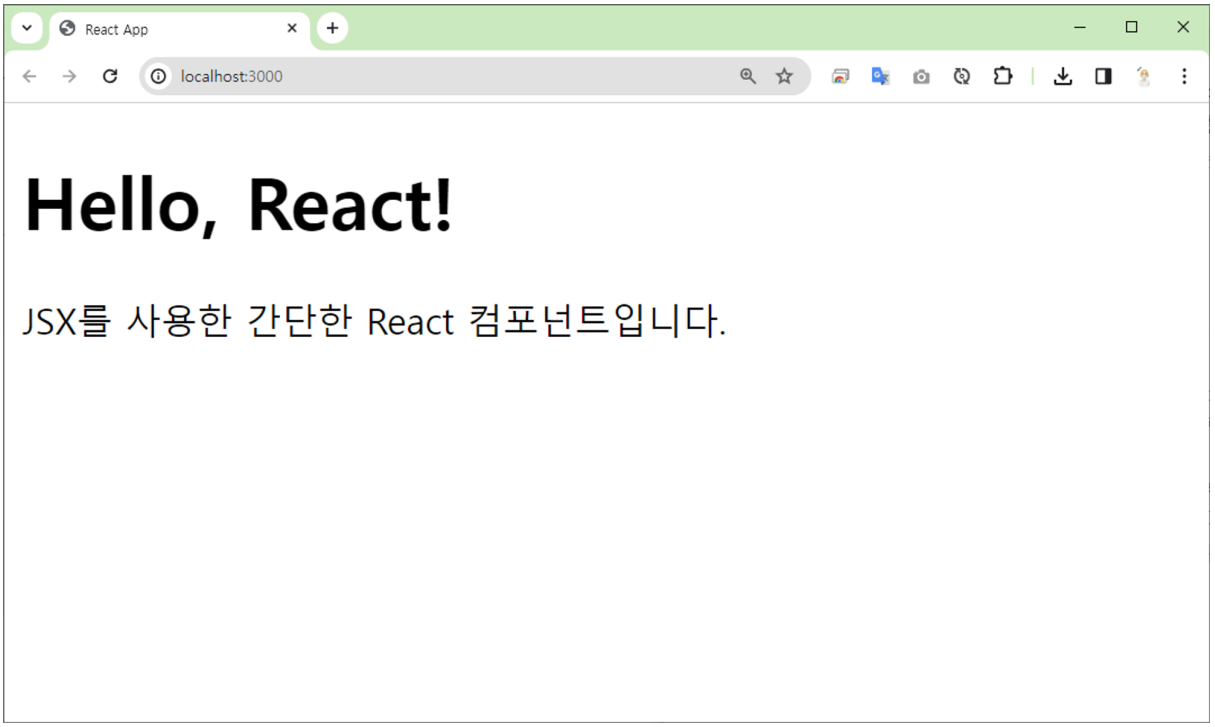
```
import React from 'react';
import './App.css';

// JSX를 사용한 컴포넌트 정의
const App = () => {
    console.log('App() 함수 호출됨');
  return (
    <div>
      <h1>Hello, React!</h1>
      <p>JSX를 사용한 React 컴포넌트입니다.</p>
    </div>
  );
};

export default App;
```

개발 모드로 프로젝트를 실행하면, 수정된 소스코드가 즉시 실행에 반영된다.
이 기능을 핫 리로딩(Hot Reloading) 또는 핫 모듈 교체(Hot Module Replacement) 이라고 한다.

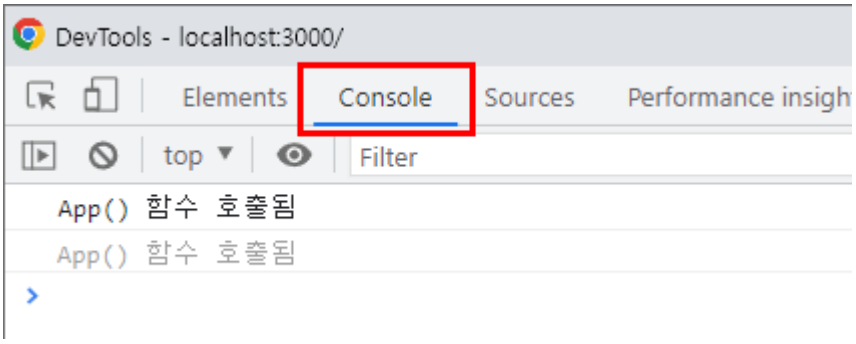
아주 가끔 핫 리로딩 기능이 작동하지 않을 수 있다고 한다.
그럴 때는 웹페이지를 새로고침 하면 된다고.

프로젝트 실행

```
npm start
```



console.log('App() 함수 호출됨');
이 메소드로 값을 출력하는 것은 웹브라우저 콘솔 창에 출력된다.



이렇게 콘솔 창에 어떤 값을 출력하는 것은 디버깅할 때 유용

렌더링

리액트 앱의 App 함수가 만들어서 리턴한 Virtual DOM 태그 객체들이
웹브라우저 창에 출력되는 과정을 렌더링 이라고 한다.

렌더링 절차는 다음과 같다.

1. 화면이 다시 그려져야 한다고 판단될 때, 리액트 엔진은 App 함수를 호출한다.

- 앱이 처음 시작 때
- 데이터가 변경되었을 때

2. App 함수는 화면을 구성하는 html 태그, Virtual DOM 객체를 생성하여 리턴 한다.
   이 리턴 값을 리액트 엔진이 받아서 처리합니다.
3. 처음 렌더링 하는 것이라면?
   App 함수가 리턴한 Virtual DOM 객체들과 동일한 구조의 DOM 객체들을 생성하여
   index.html 웹페이지의 태그 사이에 삽입한다.
   웹 브라우저는 새로 삽입된 DOM 객체들을 웹 브라우저 창에 그린다.
   렌더링 끝
4. 처음 렌더링 하는 것이 아니라면?

   리액트 엔진은,

   > 렌더링 직전 App 함수가 리턴한 Virtual DOM 객체들과
   > 다시 렌더링 하기 위해 방금 App 함수가 리턴한 Virtual DOM 객체들을 비교한다.

   위와 같이 비교하여 달라진 부분을 찾고,
   그 달라진 부분에 해당하는 DOM 객체들만 수정합니다.
   웹 브라우저는 수정된 DOM 객체들을 웹 브라우저 창에 다시 그린다.
   렌더링 끝.

패키지 설치 명령

실행에 필요한 패키지 설치 명령

실행에 필요한 패키지(라이브러리)를 설치하는 명령은 다음과 같다.

```
npm install 패키지명
```

위 명령을 아래처럼 간단히 입력해도 된다.

```
npm i 패키지명
```

실행에 필요한 패키지들은 package.json 파일의 depedencies 항목에 등록된다.

개발에만 필요한 패키지 설치 명령

실행할 때에는 필요 없고 개발할 때만 필요한 패키지를 설치하는 명령은 다음과 같다.

```
npm install --save-dev 패키지명
```

위 명령을 아래처럼 간단히 입력해도 된다.

```
npm install -D 패키지명
```

```
npm i -D 패키지명
```

개발할 때만 필요한 패키지들은 package.json 파일의 devDepedencies 항목에 등록된다.

node_modules 디렉터리

npm 명령에 의해서 다운로드 되고 설치된 패키지들은 node_modules 디렉터리 아래 저장된다.

다음 명령을 실행하면 package.json 파일에 등록된 패키지들이 모두 다시 다운로드 되므로,
프로젝트 폴더를 압축할 때 node_modules 디렉터리는 삭제하는 것이 좋다.

프로젝트에 필요한 패키지들 전체 설치 명령

```
npm install
```

위 명령을 실행하면 package.json 파일에 등록된 패키지들이 모두 다운로드 되어 프로젝트에 설치된다.