개발환경 구축

1. node 설치

```
https://nodejs.org
npx, npm 사용을 위해
LTS버전을 사용추천

node -v
node --version
npm --version
npm -v
npx --version
npx -v
```

2. vscode 설치

```
https://code.visualstudio.com/

Plugin
- Korean Language Pack for Visual Studio Code
- Material Icon Theme
- ESLint
- Auto Close Tag
- Auto Rename Tag
- Prettier
- ES7+ React/Redux/React-Native snippets
```

3. react 설치

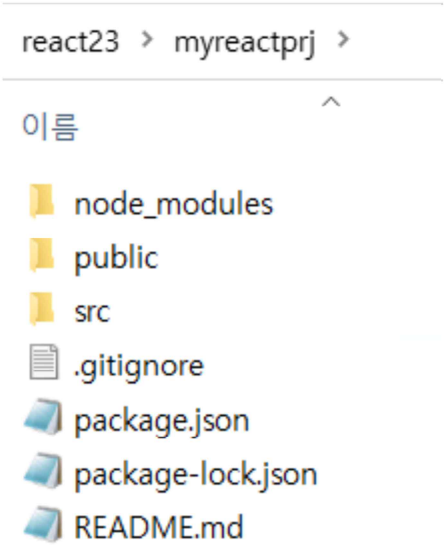
```
npx create-react-app 프로젝트명

cd 프로젝트명
시작: npm start
중지: Ctril+C
실행확인: http://localhost:3000/
```

프로젝트 생성

- npx create-react-app react23prj
- cd react23prj
- npm start
- https://localhost:3000/

폴더 구조



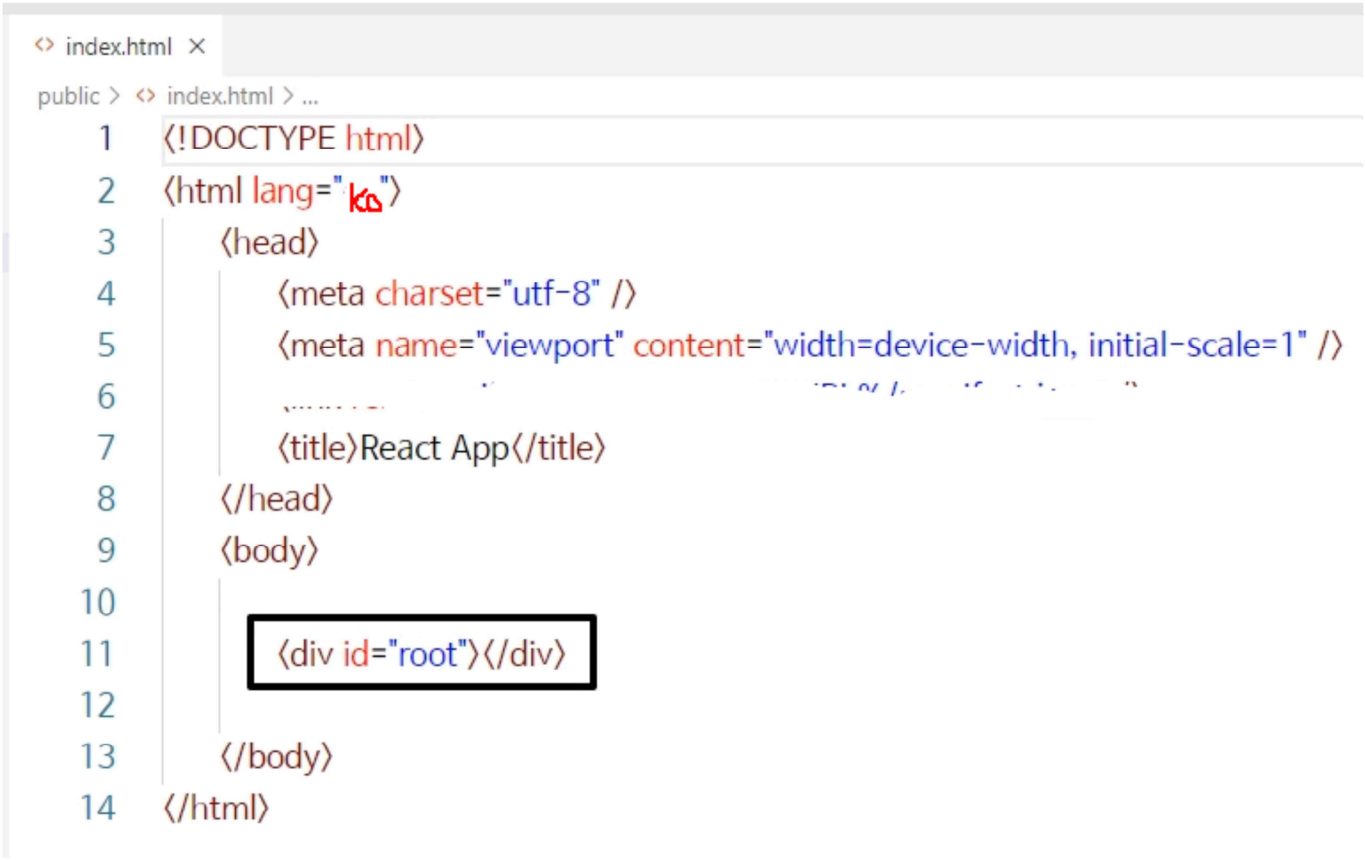
https://localhost:3000/index.html

- 표면적인 시작점
  - react23prj/public/index.html
  - react23prj/src/index.js
- 실질적인 시작점
  - react23prj/src/App.js

react23prj/public/index.html

- 주석문 삭제
- Ctrl+k,f : 자동 indent처리
- Ctrl+/ : 주석 토글
- Alt+shift+A: 주석 토글
- 제일중요한 부분

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



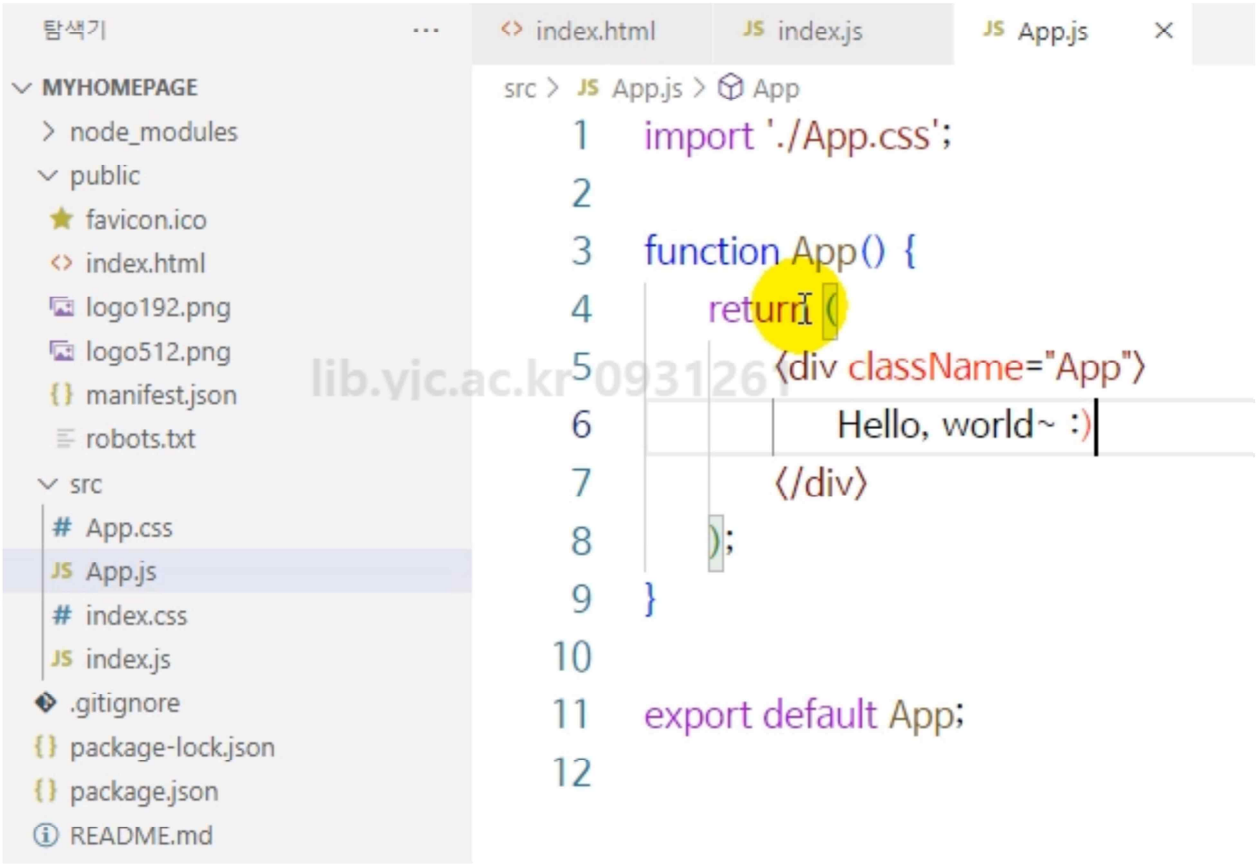
react23prj/src/index.js

- `<App />` 컴포넌트 부분: App 선택후 Ctrl 클릭

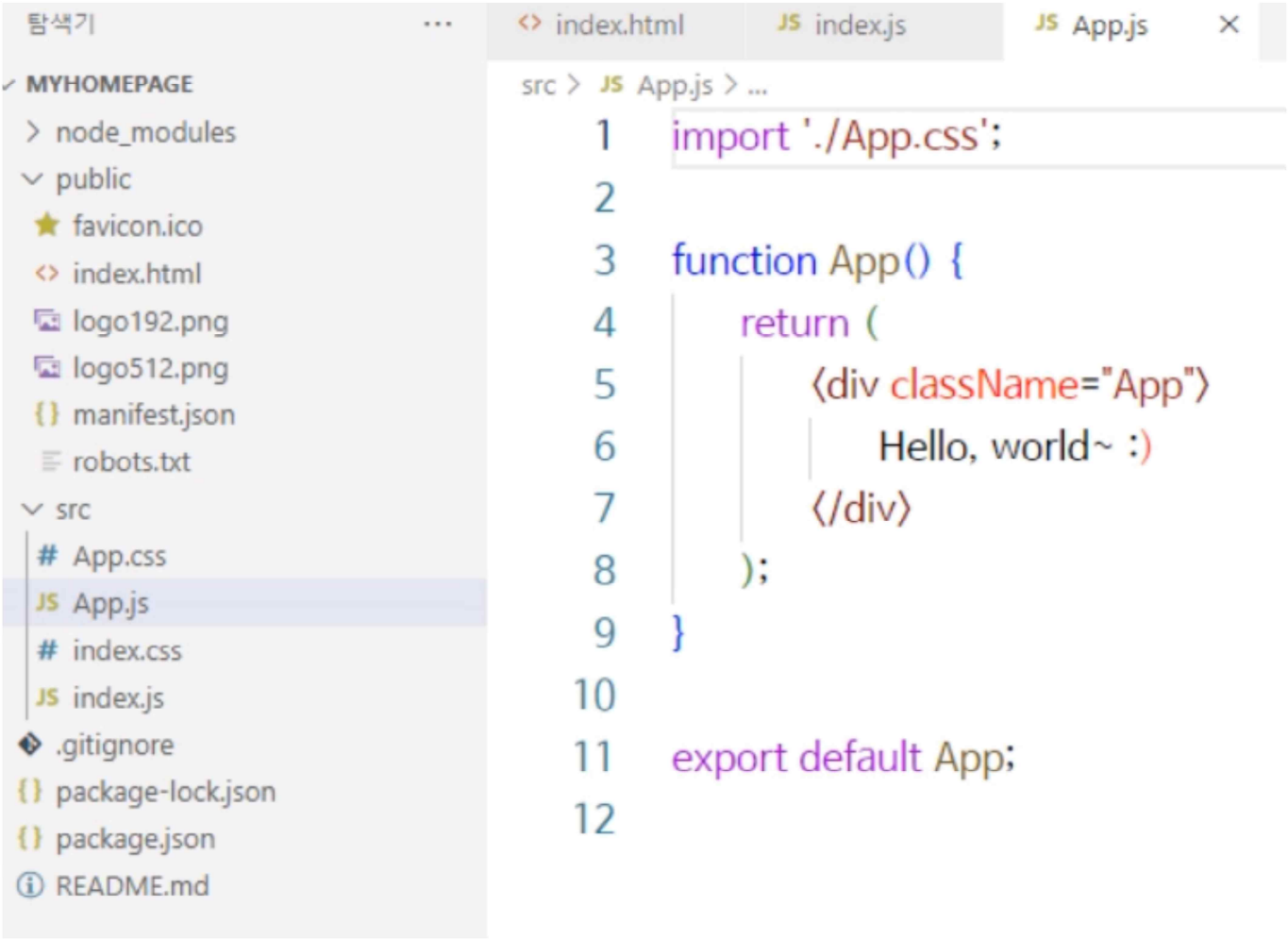
react23prj/src/App.js

```
function App() {
  return (
```

```
      <div className="App"></div>
    )
  }
```



불필요한 파일 제거후

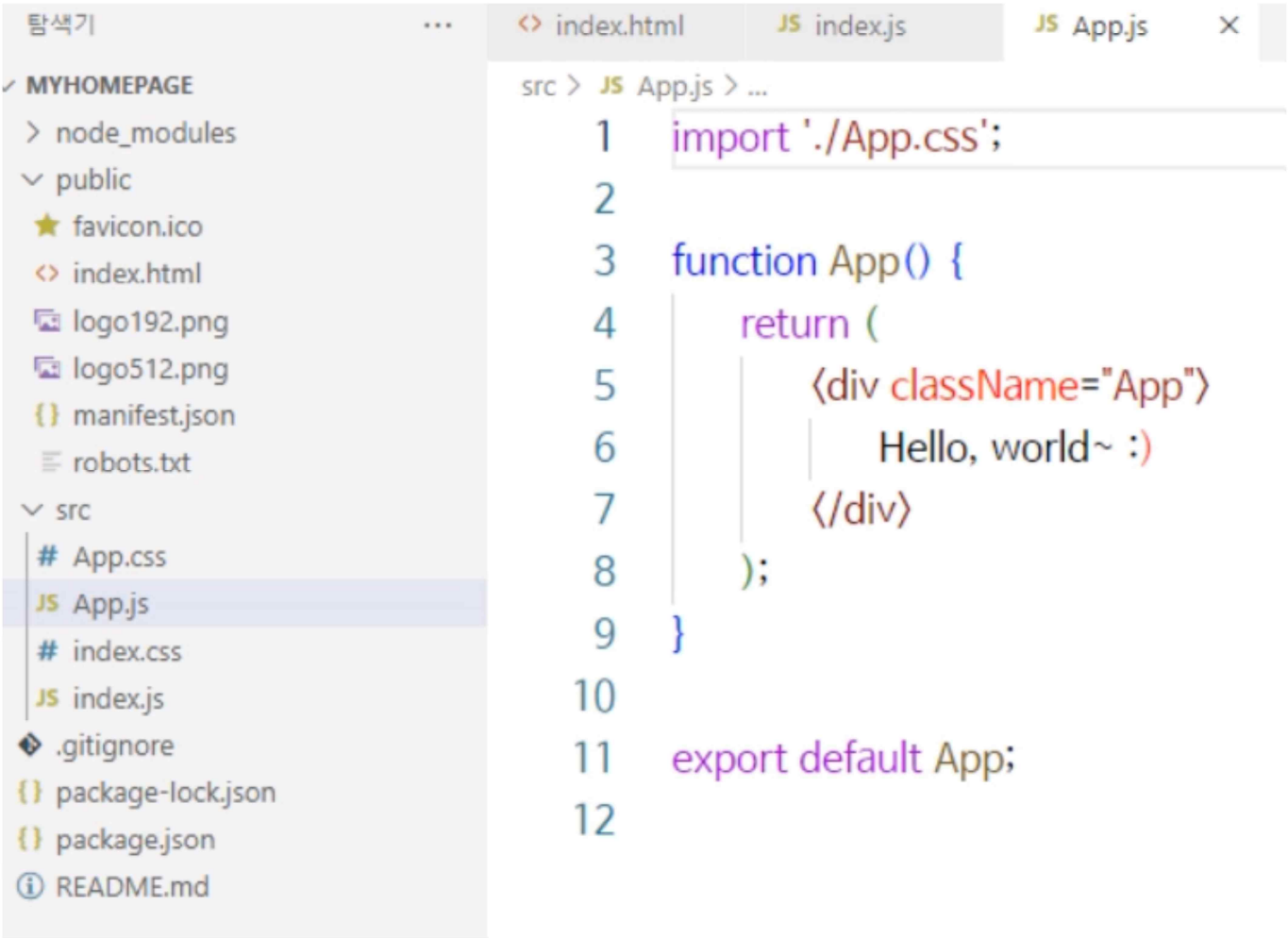

탐색기
index.html
index.js
App.js
MYHOMEPAGE
node_modules
public
favicon.ico
index.html
logo192.png
logo512.png
manifest.json
robots.txt
src
App.css
App.js
index.css
index.js
.gitignore
package-lock.json
package.json
README.md
src > App.js > ...
import './App.css';
function App() {
return (

Hello, world~ :)

);
}
export default App;

처음부터는 불필요한 파일과 코드 삭제

- `<div id="root"></div>`
- `<App />`
- `return (<div className="App"></div>)`



{: width="100" height="100"}

컴포넌트 기본 작성 규칙: JSX - JS+XML(html)

항상 대문자로 컴포너트명 지정

함수형

- 최신에는 함수형으로 작성
- 각 컴포넌트에는 무조건 하나의 최상위 태그을 포함해야, 최상위 태그의 자식으로 소스코드 작성되도록

```
function LikeButton() {
 return (
   <div className="App">   // JSX 문법
   </div>
 );
```

```
}

const domContainer = document.querySelector('#like_button_container');
//const domContainer = document.getElementById('like_button_container');
const root = ReactDOM.createRoot(domContainer);
root.render(<LikeButton></LikeButton>);
```

- className: 일반적인 html은 아님, 유사, js의 클래스 키워드와 동일 - class
- JSX의 기본 표기법: camelCase
    - backgroundColor
- html,css: kebab-case
    - background-color

클래스형

```
<like_button.js>
class LikeButton extends React.Component{
    //생성자
    constructor(props){
        super(props);
        this.state ={liked:false};
    }
    render(){
        if(this.state.liked){
            return 'You liked this.';
        }
        return React.createElement(
            'button',
            {onClick: ()=> this.setState({liked:true})},
            'Like'
        );
    }
}

const domContainer = document.querySelector('#like_button_container');
const root = ReactDOM.createRoot(domContainer);
root.render(<LikeButton></LikeButton>);
```

인라인방식 CSS적용

- JSX 규칙에 따라 `<div style="border: 1px solid">` --> 에러

최상위에 하나의 엘리먼트가 작성되어 있어야

- 헬로우월드! ---> 에러
  - `<h1>헬로우월드!</h1>`
  - `<Fragment>헬로우월드!</Fragment>`
  - `import { Fragment } from "react";`
  - `<>헬로우월드!</>`

종료태크

- `<App></App>`
- 축약: `<App />`

import

- 수입하다. 불러오다.
- import 컴포넌트명 from '컴포넌트명.js'
- import 컴포넌트명 from '컴포넌트명'

export

- 수출하다. 내보내다.
- export default function App(){...} 도가능
- export default App; 대신

import MyHomeP from './App.js'

- `<MyHomeP />` 으로 render()에서 사용

터미널: ctrl+` (백틱)

프로젝트 폴더의 터미널에서 npm start로 실행

프로젝트 폴더의 터미널에서 Ctrl+C로 종료

리액트에서 사용 표기법 (관례) : Coding Conventions

- camelCase
  - className, backgroundColor
  - Hungarian(Windows API): strName, bBusy, szName, 접두사 - 데이터 자료형
- kebab-case
  - background-color
- snake_case
  - class_name_abc
- PascalCase
  - ClassName

index.js파일 호출

- index.html에서 명확히 호출하는 코드가 없음
  - `<script src='../src/index.js'>`코드가 없음
  - 추가하면 에러
- 명시적으로 호출하지 않고 서버에서 처리
  - webpack
  - 서버 구동시, index.html에 bundle.js파일 임포트 시킴
    - `<script src='/static/js/bundle.js'> </script>` 코드가 삽입됨
    - 실행브라우저의 소스보기로 확인
- 첫번째 `<style>`내: index.css
- 두번째 `<style>`내: App.css