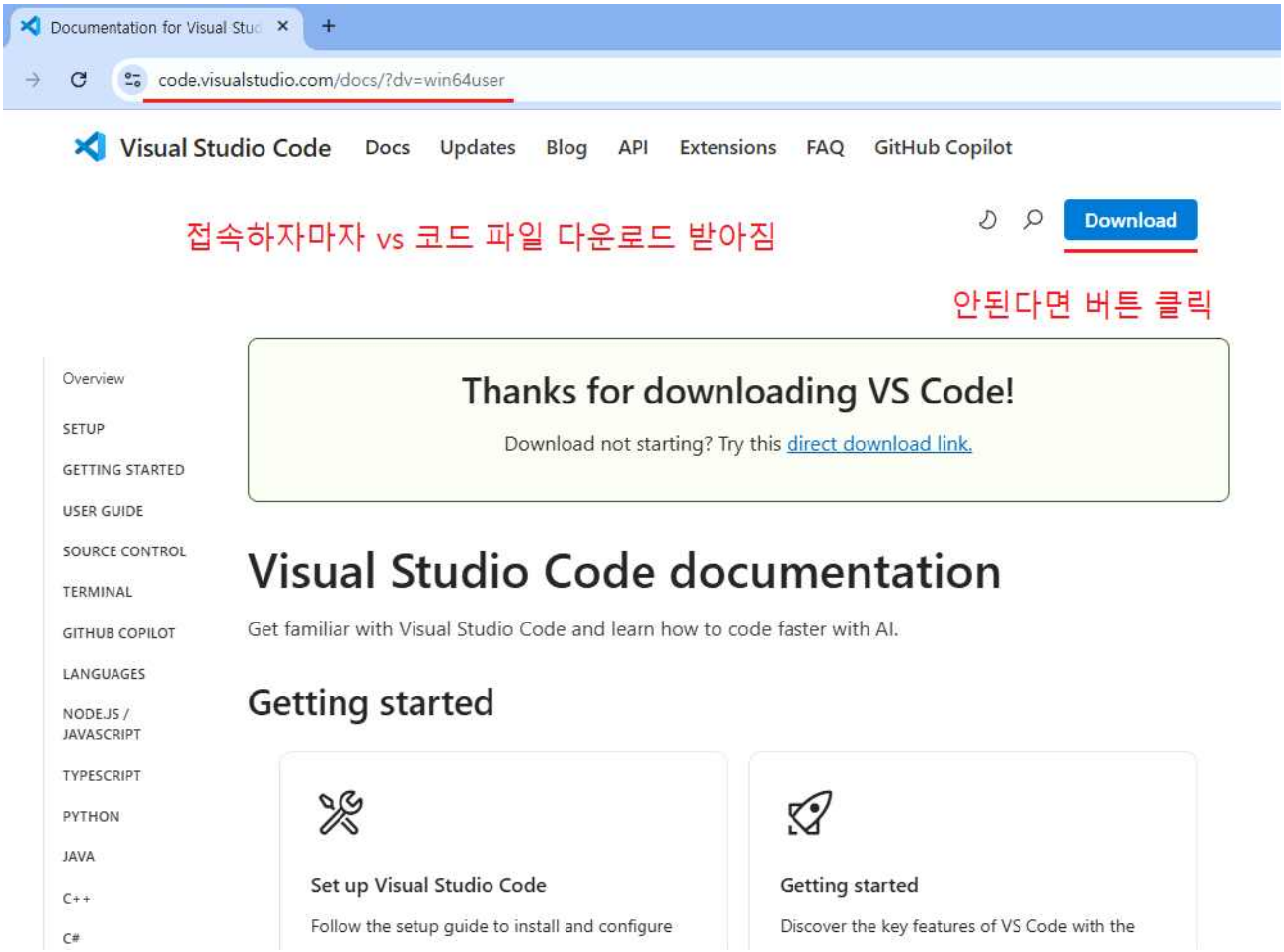
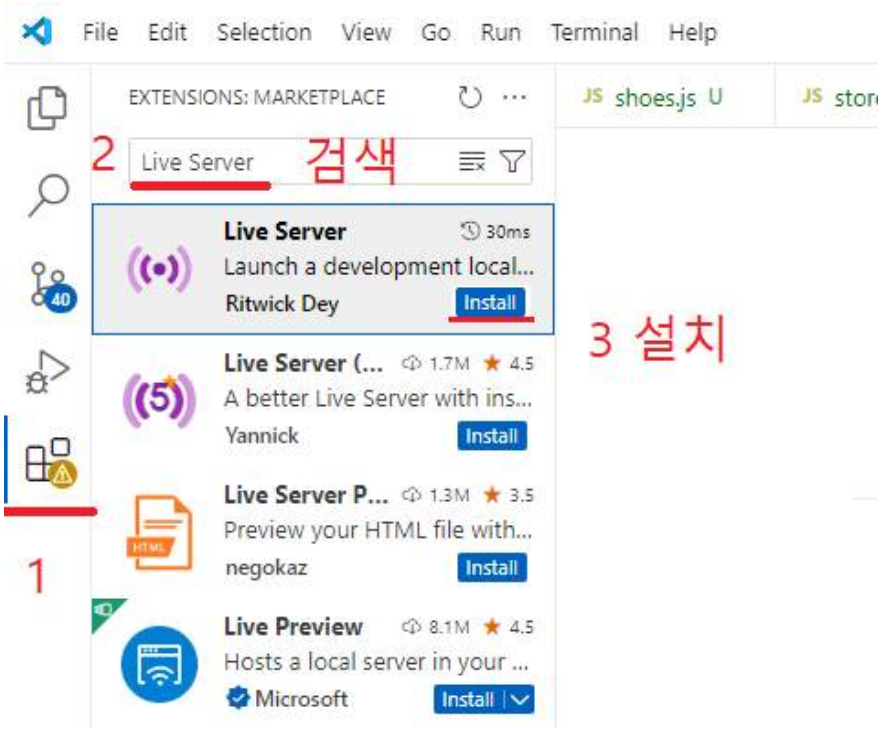
VS / node / react 설치

1 VS 코드 설치 (설치 전 자리에 있는지 확인)

https://code.visualstudio.com/docs/?dv=win64user

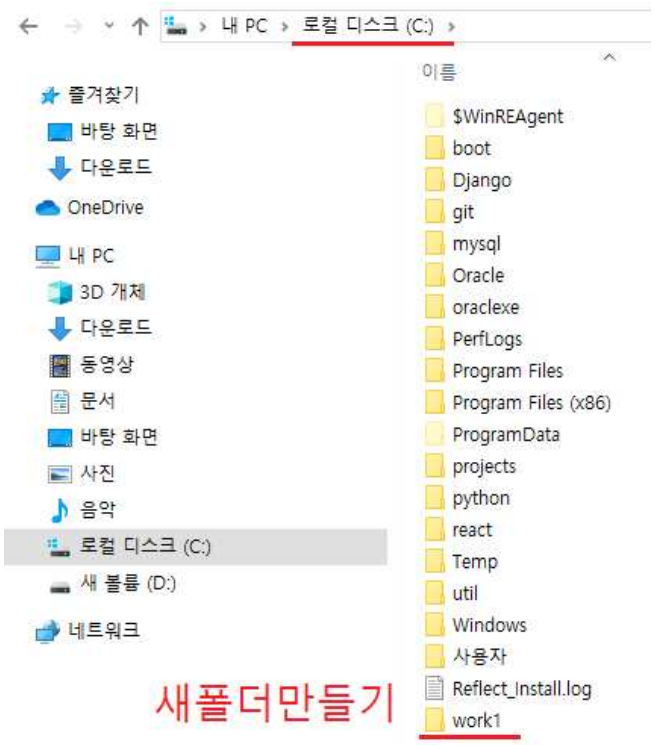


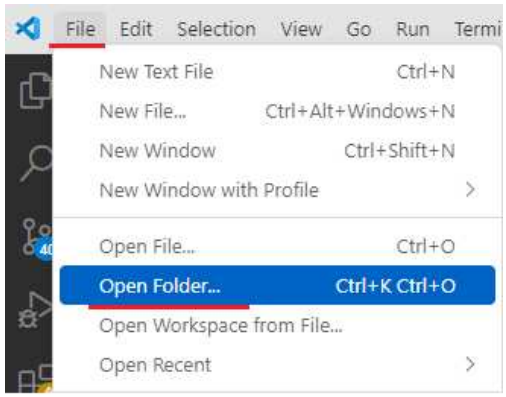
1 - 3 비주얼스튜디오 확장 프로그램 깔기

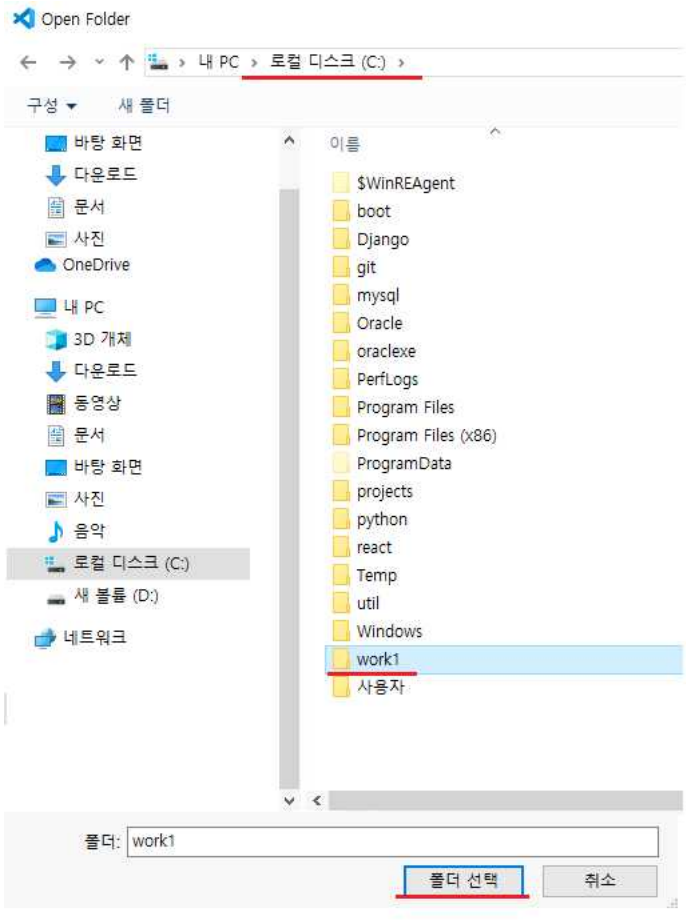
비주얼 스튜디오에서	
Open in browser	Alt + B
Live Server	Live Server → 실시간으로 브라우저에서 적용된 내용을 확인 (Alt + B)
auto rename tag	태그의 시작/종료를 동시에 수정
prettier	코드 자동 정렬 (Ctrl + Shift + B)
css Peek	HTML에서 class에 적용된 CSS 속성으로 바로 이동 (<Ctrl> + <마우스 클릭>)
Auto import	파일명을 입력하면 자동 import
Auto import -es6	
Indent Rainbow	들여쓰기를 색깔로 구분해서 가독성을 높임
Reactjs code snippets	rsc로 함수 만들어줌 (자동완성
ES7 + React/Redux/React-Native snippets	rfce - 함수만들어줌(자동완성)

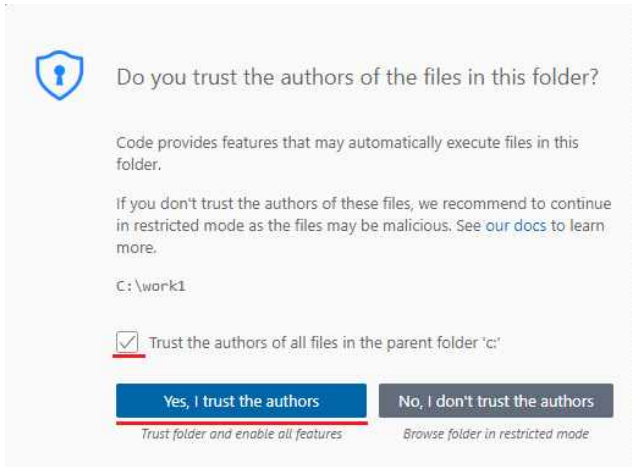
1 - 4 비주얼스튜디오 – 설정 (생략)

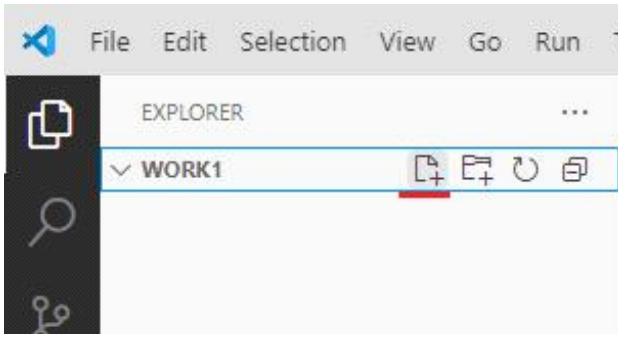
작업폴더만들고 index.html 파일만들기

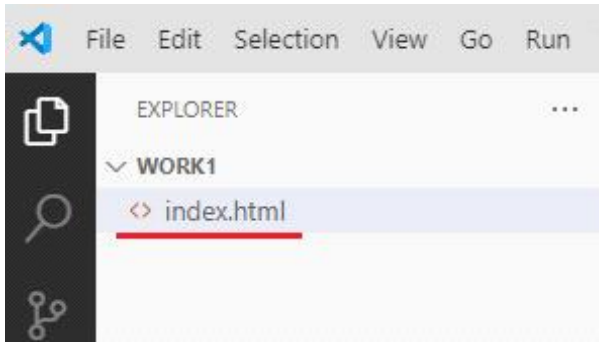


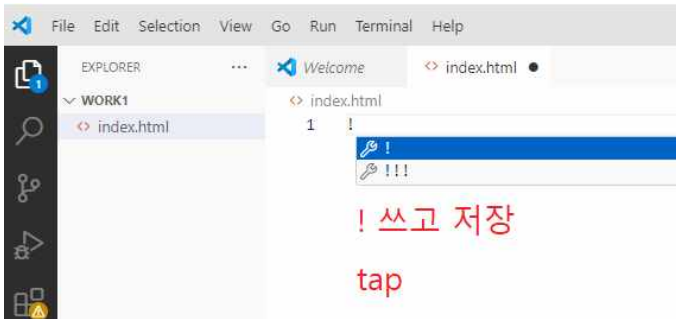




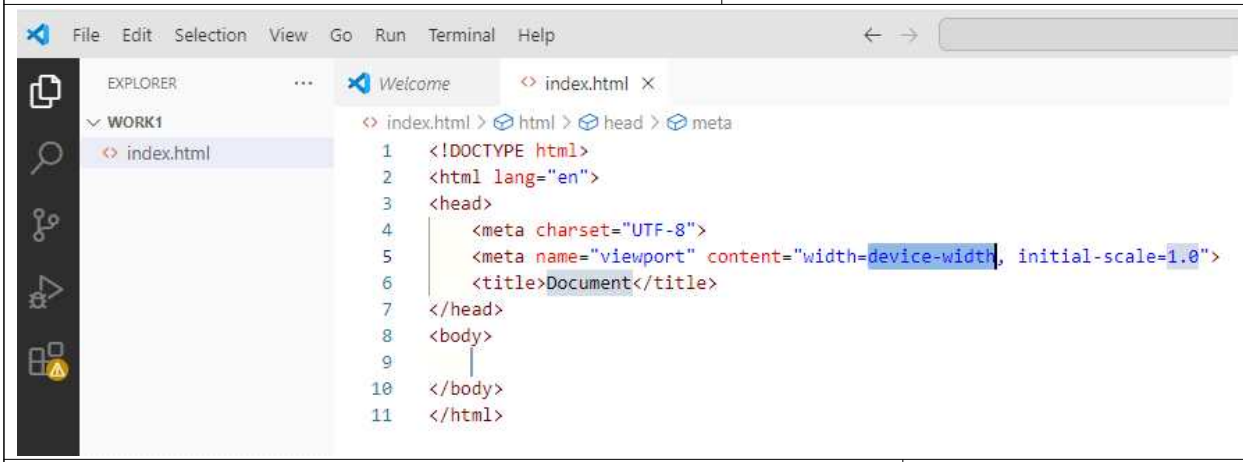


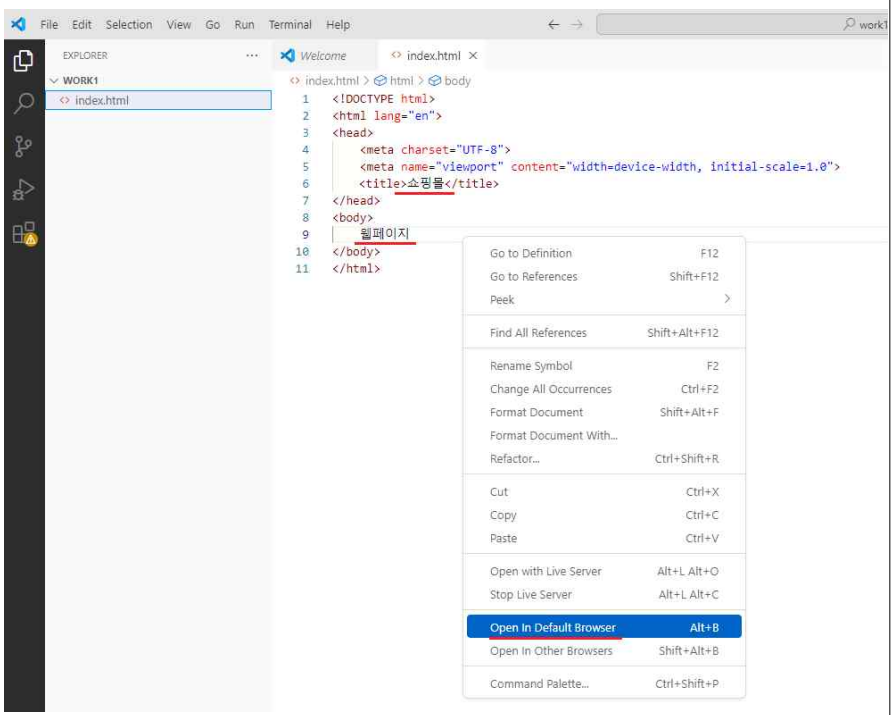






1. 파일저장하기 index.html
2. ! + Tab 키

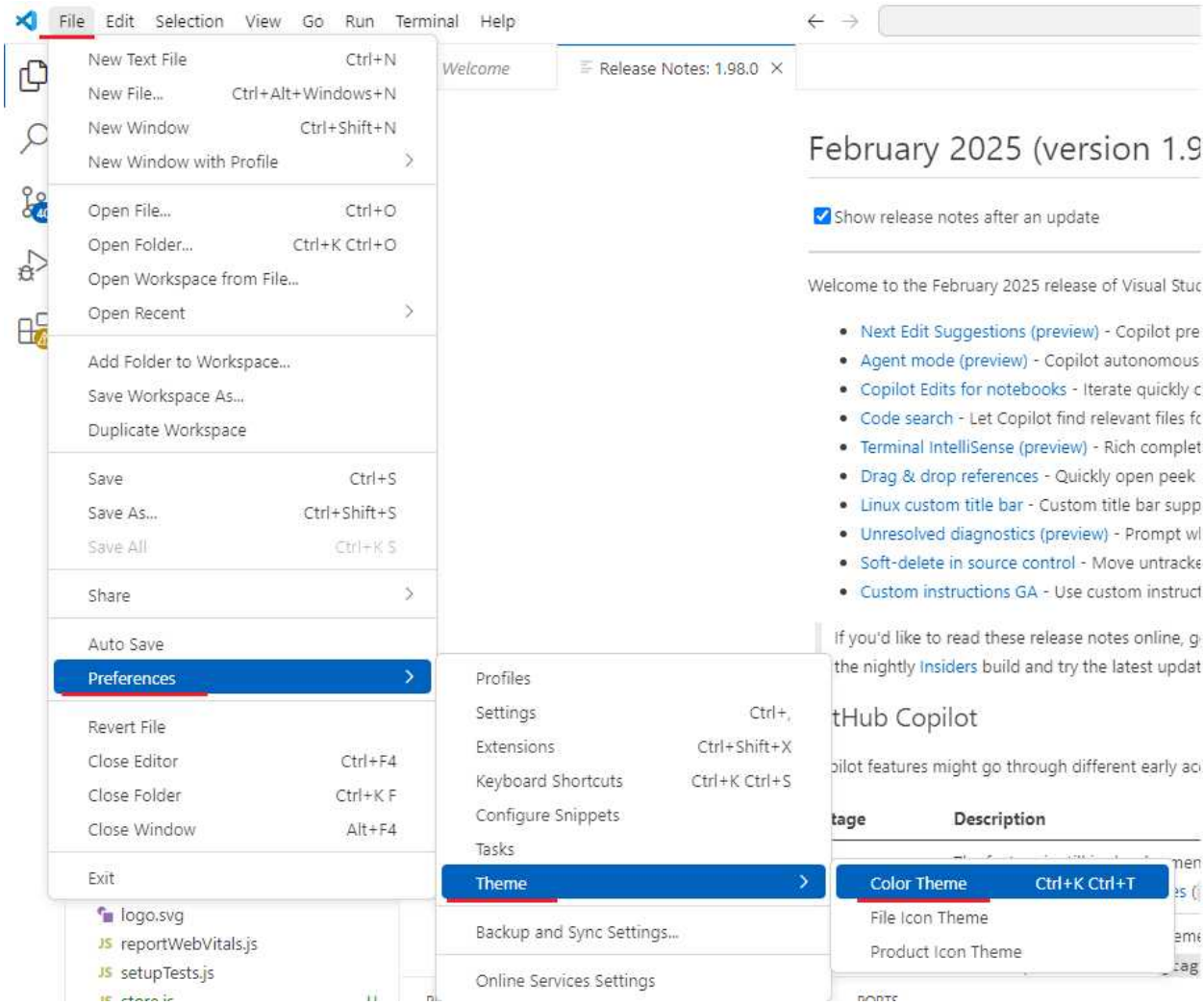


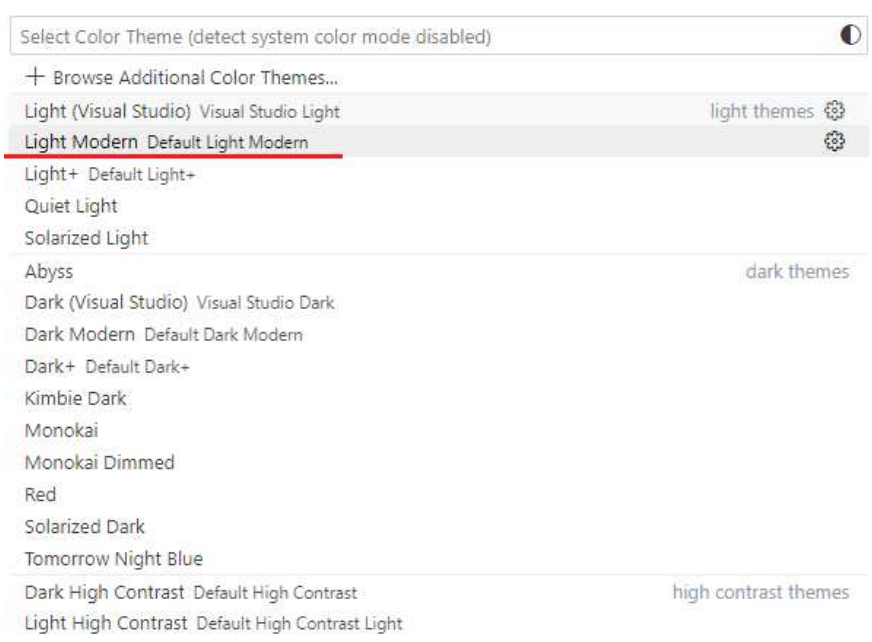


웹페이지

비주얼스튜디오 - 밝은 테마 사용하기

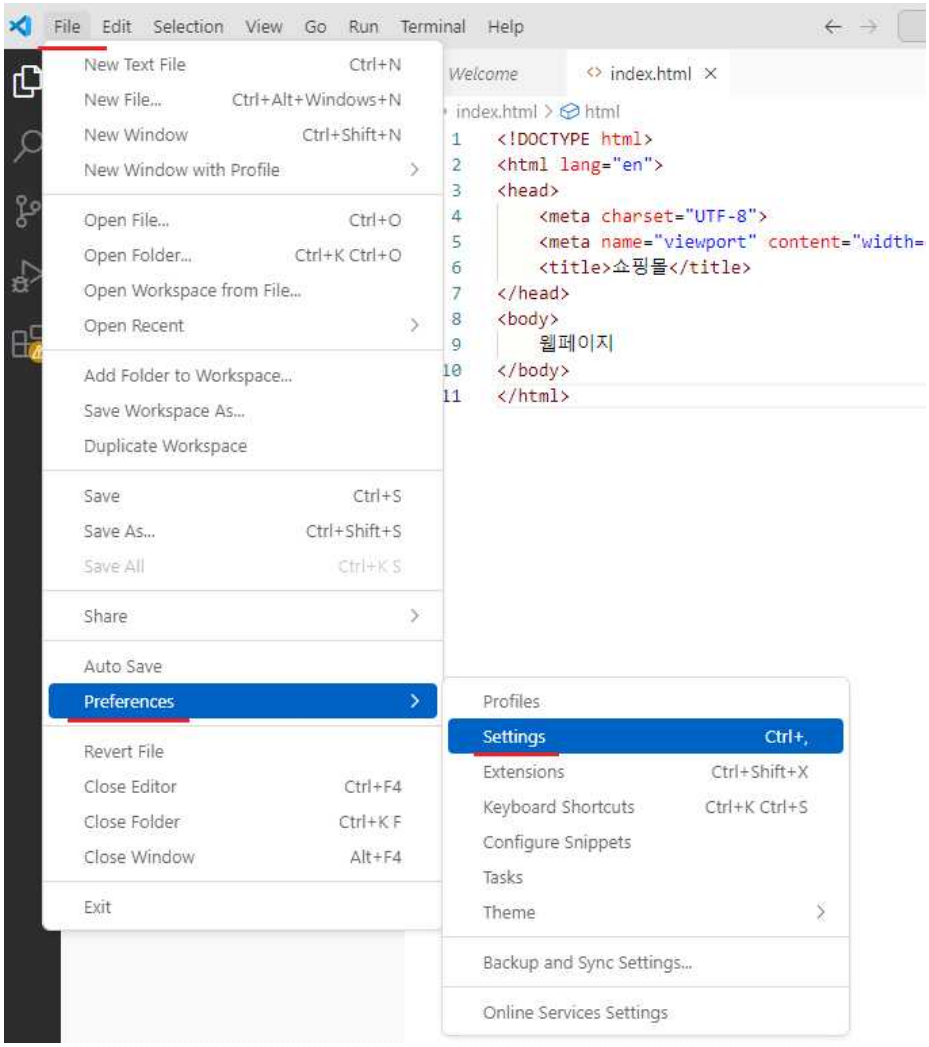
File > Preferences > Theme > Color Theme



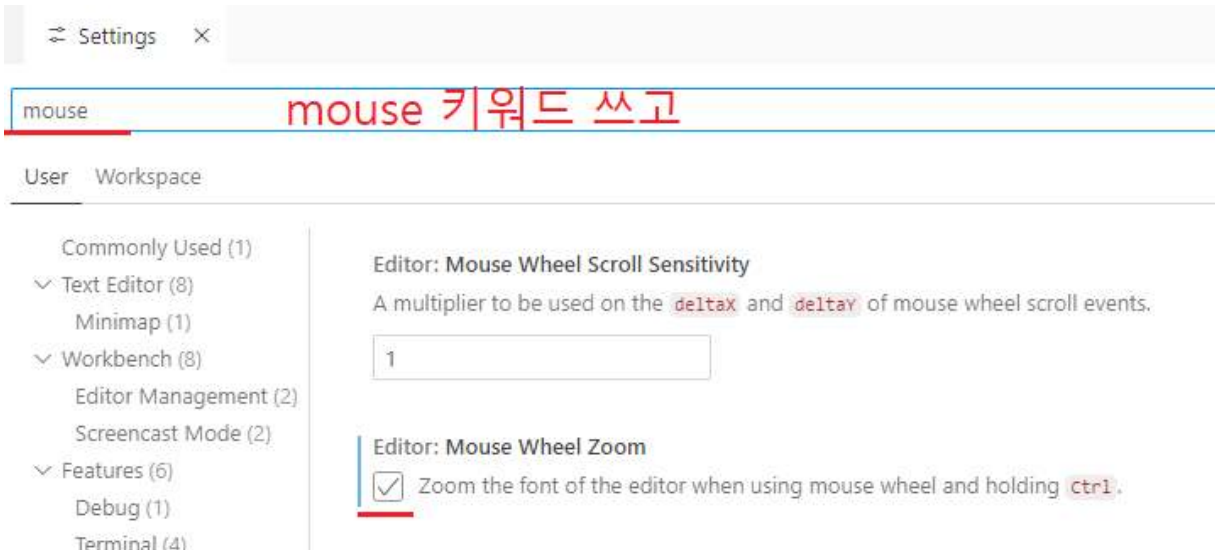


비주얼스튜디오 - 마우스휠로 폰트 키우기

File > Preferences > Settings

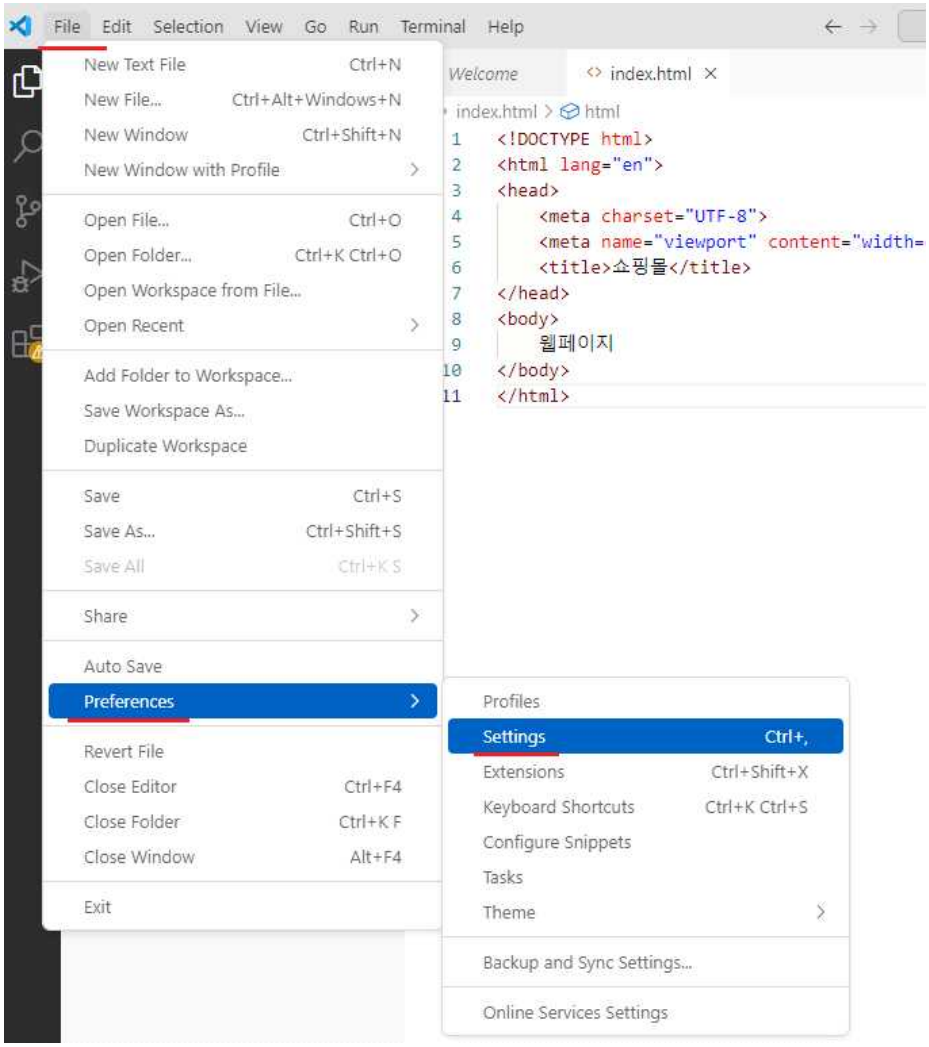


settings > mouse키워드 써주고 > Mouse Wheel Zoom 체크하기

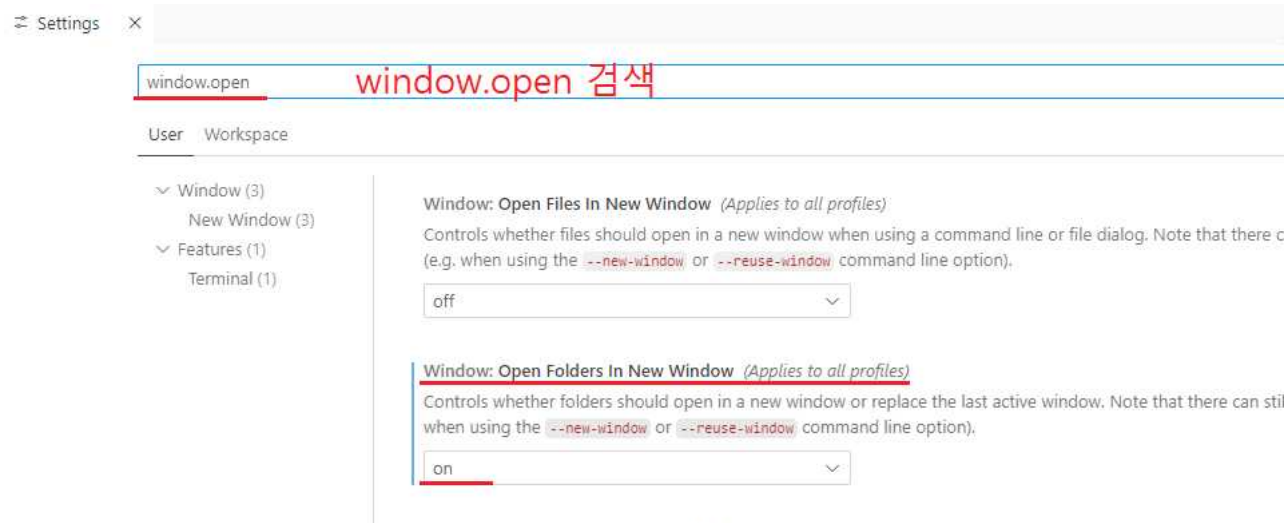


VSCode에서 파일을 항상 새 창에서 열기, 프로젝트 두 개이상 동시에 작업 설정

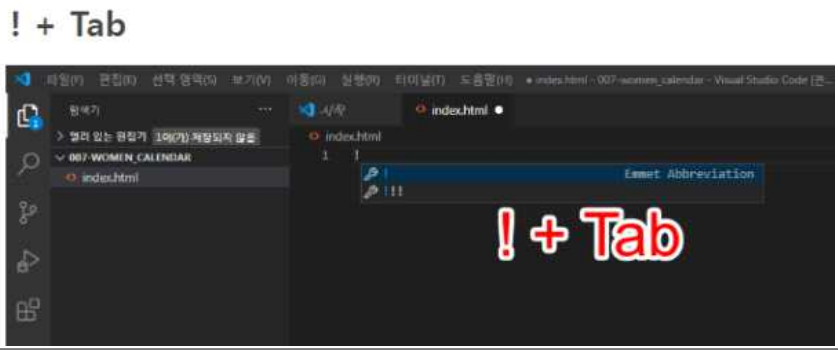
File > Preferences > Settings



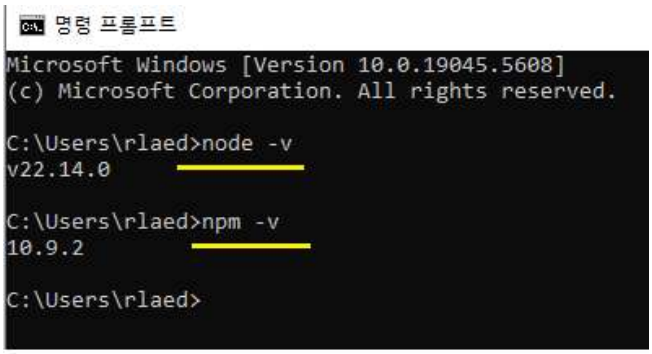
검색 : window : Open Files In New Window 에서 On으로 설정

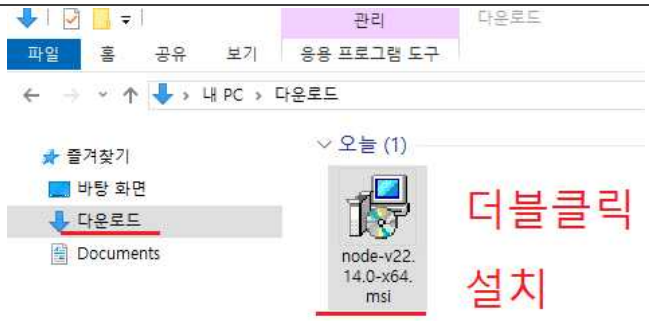
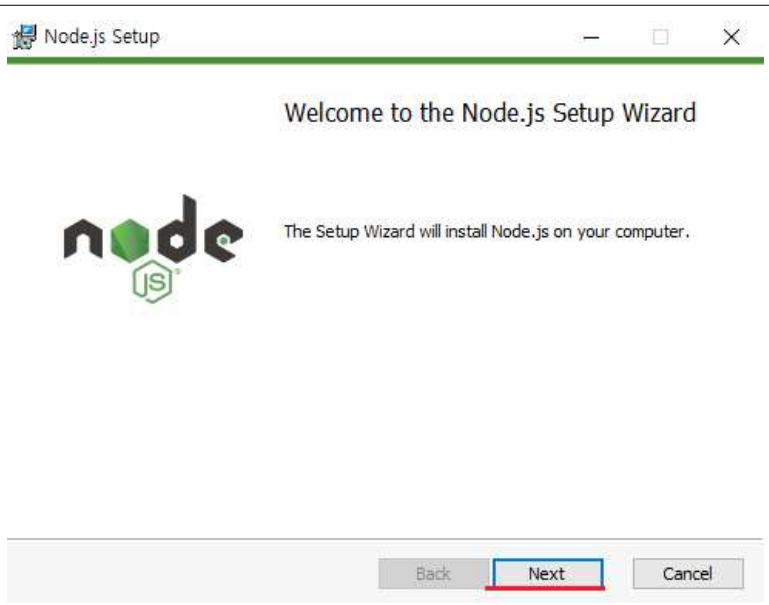


비주얼스튜디오 단축키

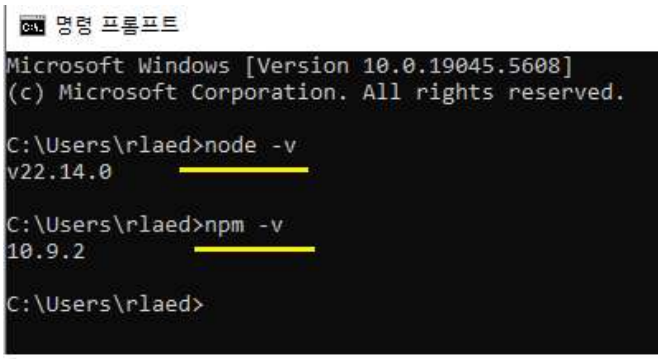
html 자동완성하기	1. 파일저장하기 : index.html 2. ! + Tab 키 
코드 정렬하기	1. Ctrl + A 키를 눌러 코드 전체를 모두 선택 2. Ctrl 키를 누른 상태에서 K 와 F 를 차례대로 클릭 (json 정렬도 같음) 3. prettier 코드 자동 정렬 (Ctrl + Shift + B)
브라우저 열기	Alt + B

2 node 설치 (설치 전 cmd에서 node 있는지 확인, 없으면 설치)

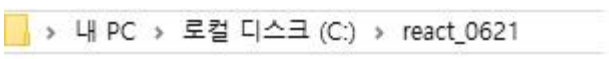
cmd에서 node가 있는지 확인	
https://nodejs.org/ko/	

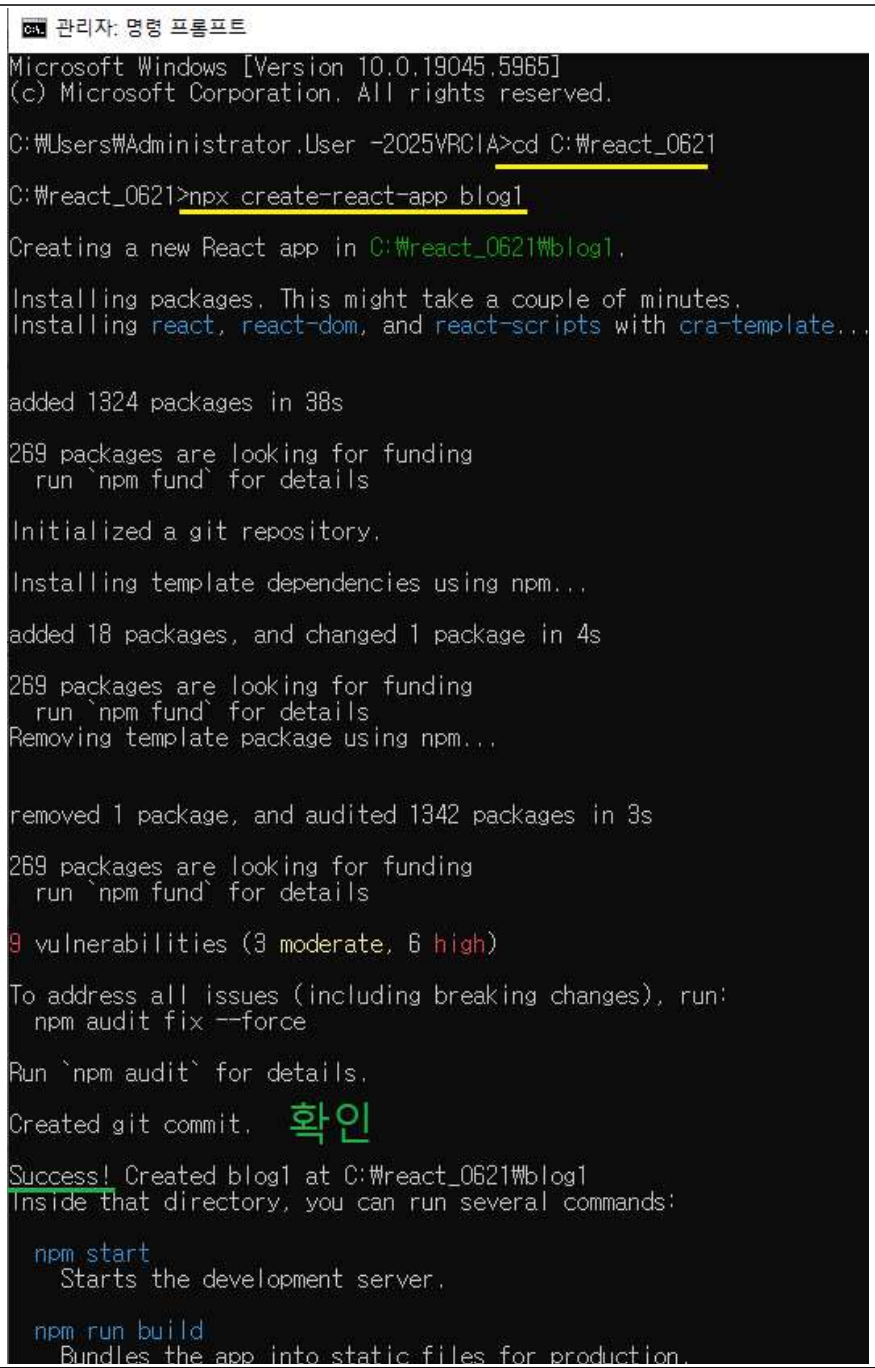
내컴퓨터에서 다운로드 폴더 확인 후 설치	
설치	

동의 체크 설치	

설치 끝나고 cmd에서 확인 node –v npm -v	

3 react 프로젝트 만들기

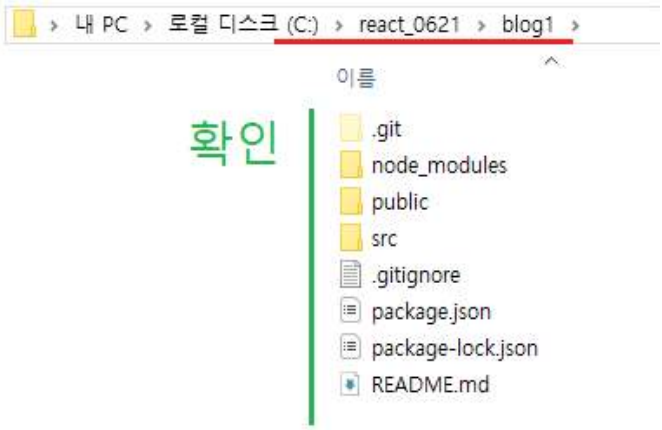
탐색기로 작업 폴더 만들기	

<table>
<tr><td rowspan="4">cmd 열기
cd C:₩react_0621
npx create-react-app blog1</td><td>


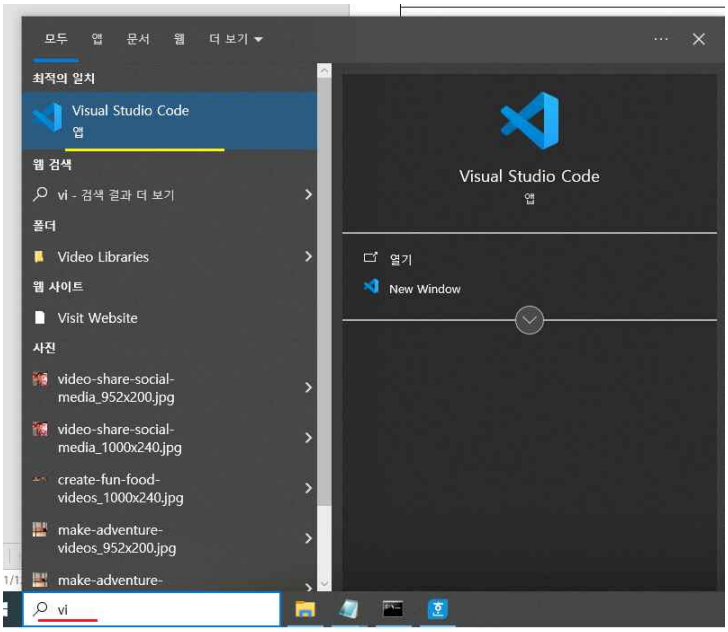
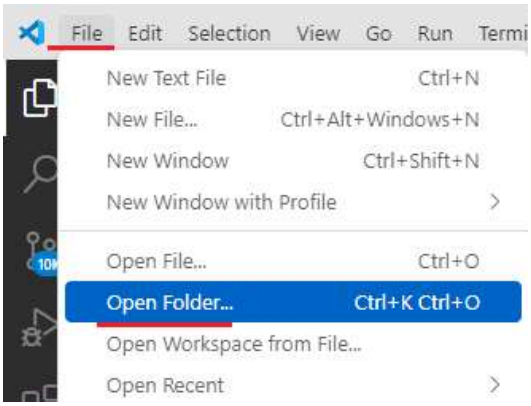
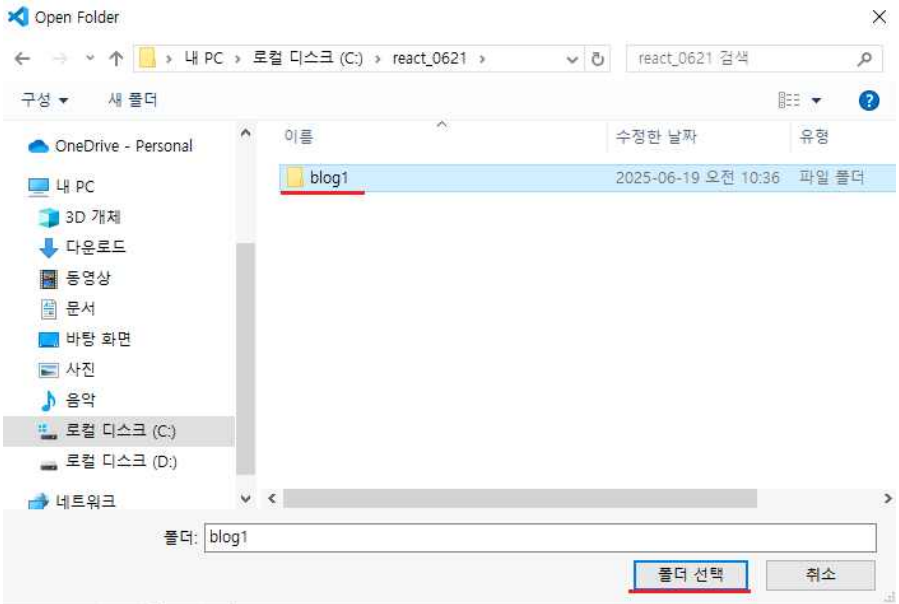
</td></tr>
<tr><td>리엑트 프로젝트가 생성되지 않으면 참고 url: https://blog.naver.com/ysboo2/223757816691</td></tr>
<tr><td>위의처럼 안뜨면

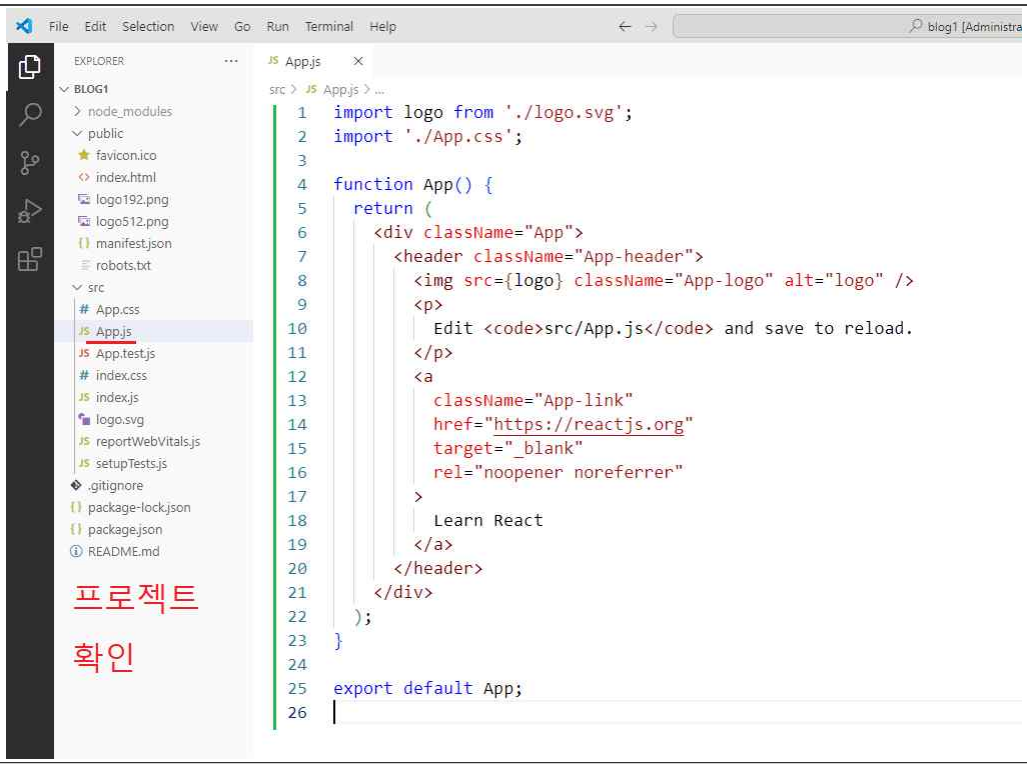
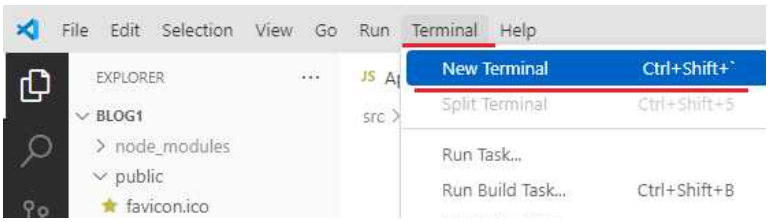
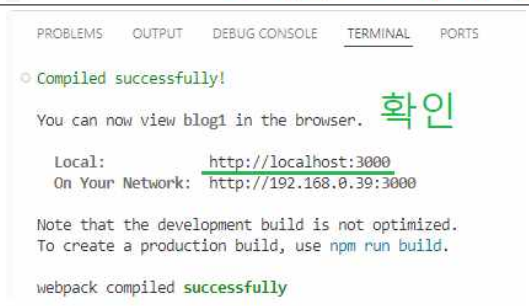
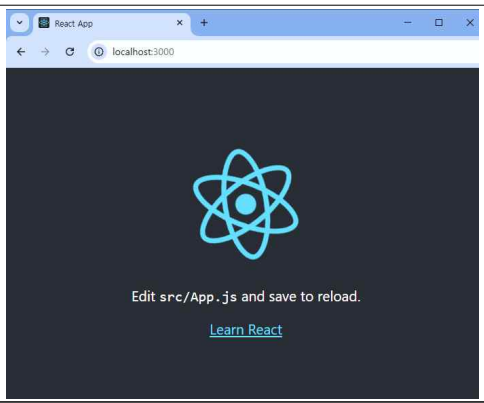
cmd에서 코드 복사
붙여넣기</td><td>cmd > 프로젝트 경로 > 아래코드 붙여넣고 실행 (아래코드 한줄, 한번에 복사)</td></tr>
<tr><td>npm install react@18 react-dom@18 @testing-library/jest-dom@5.17.0
@testing-library/react@13.4.0 @testing-library/user-event@13.5.0 web-vitals

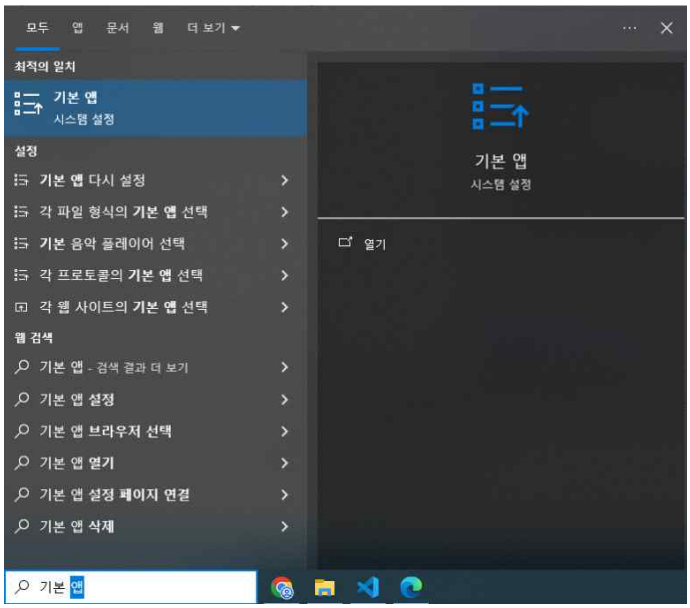
npx create-react-app blog1</td></tr>
</table>

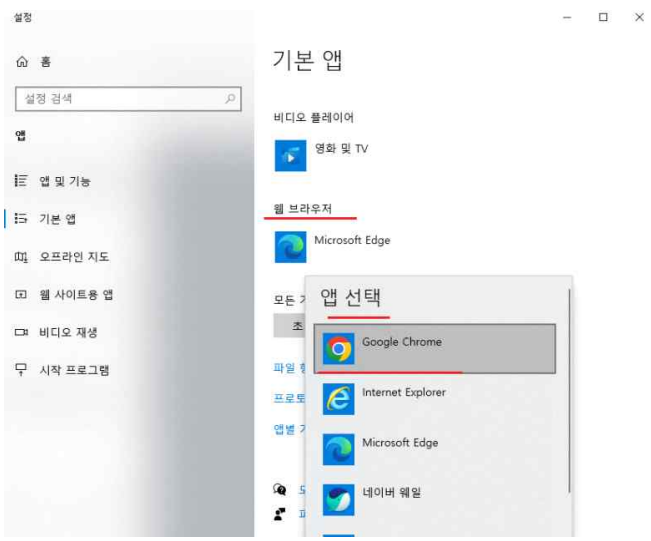
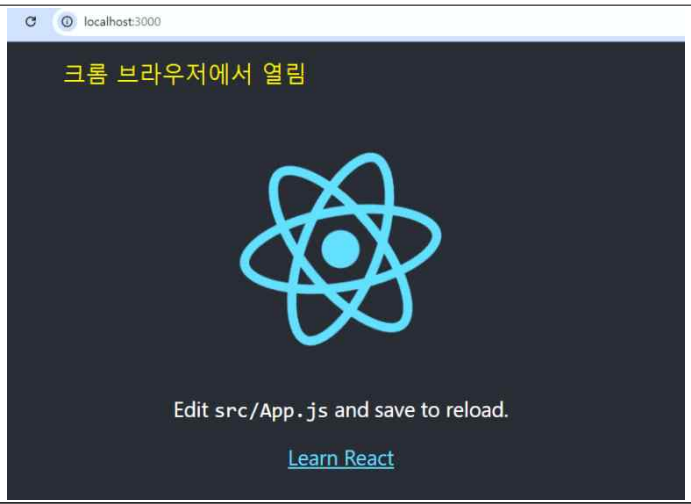
<table>
<tr><td rowspan="3">프로젝트 만들어졌는지
확인</td><td> 
</td></tr>
<tr><td> 
</td></tr>
<tr><td> 
</td></tr>
<tr><td colspan="2">탐색기에서 프로젝트가 만들어졌는지 확인하고 나서는 cmd창의 닫는다.</td></tr>
</table>

4 react 프로젝트 실행되는지 확인

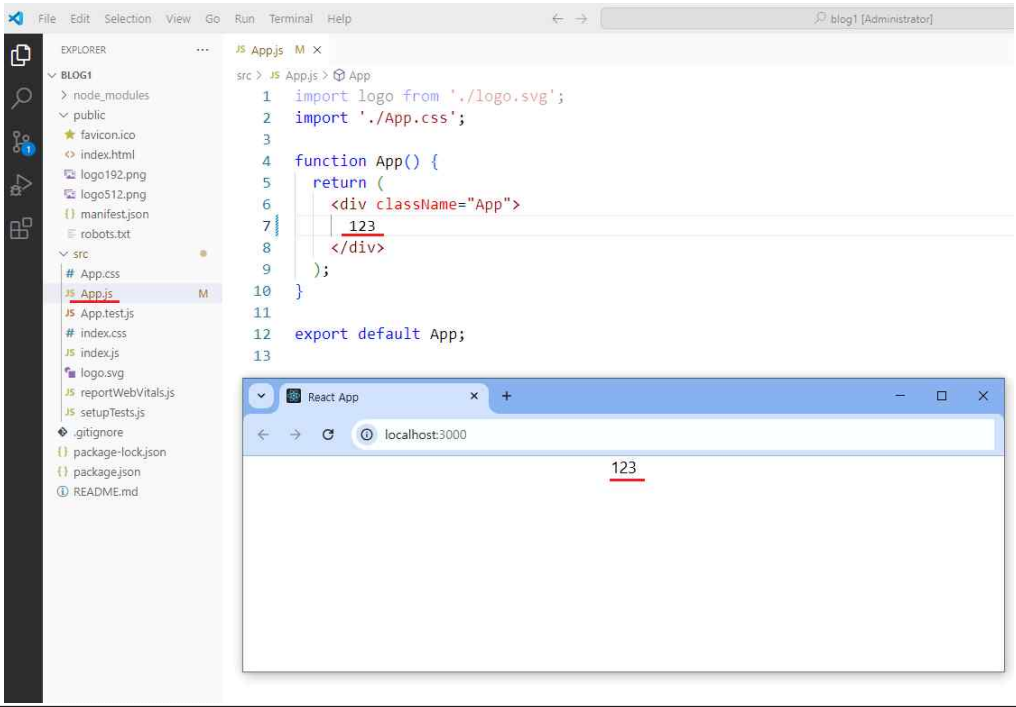
비주얼 스튜디오 열기	
vi 코드에서 Open Folder	
blog1 폴더 열기	

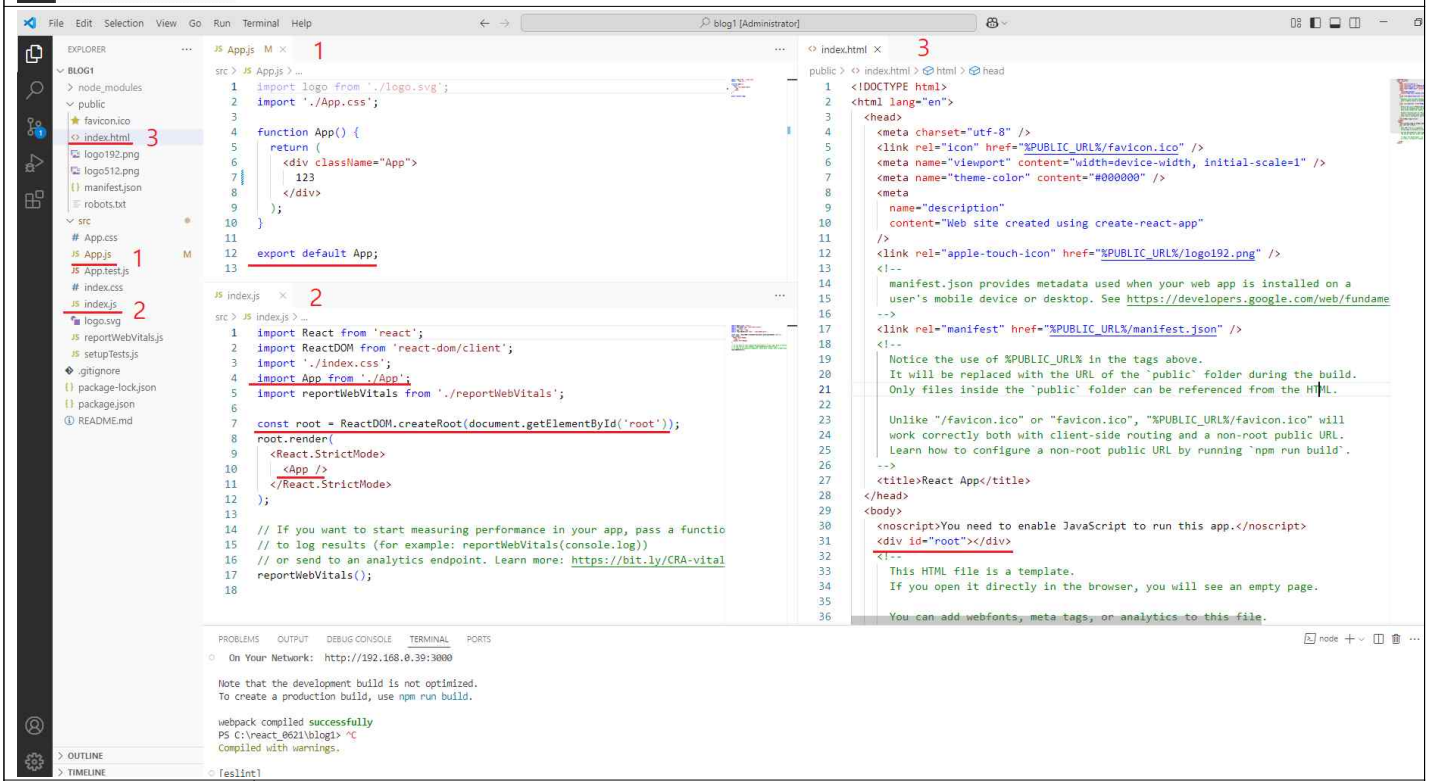
App.js 선택	
터미널 열기 단축키 Ctrl + Shift + ` (백틱)	
터미널에서 npm start	
React 앱은 http://localhost:3000 포트에서 실행	
브라우저에서 확인 http://localhost:3000/ 으로 자동으로 오픈	 옆의 이미지처럼 나와야 성공

<table>
<tr><th colspan="3">리엑트 프로젝트가 크롬 브라우저로 안 뜬다면</th></tr>
<tr><td>참고 url</td><td colspan="2">react 프로젝트 기본 브라우저가 크롬이 아닐경우 (크롬 브라우저)
https://blog.naver.com/ysboo2/223841170588</td></tr>
<tr><td colspan="3">기본 브라우저가 크롬이 아닐경우</td></tr>
<tr><td>윈도우 창에서
검색 기본앱</td><td colspan="2"> 
</td></tr>
<tr><td>기본앱 아래쪽에

웹브라우저
> 앱선택
> 구글 크롬 선택</td><td colspan="2"> 
</td></tr>
<tr><td>크롬으로 열리는지
확인</td><td> 
</td><td>프론트 서버 종료
터미널에서 Ctrl + C</td></tr>
</table>

App.js > 작업 header를 다 지우고 > 123만 작성 > 터미널에서 npm start > 브라우저에서 확인

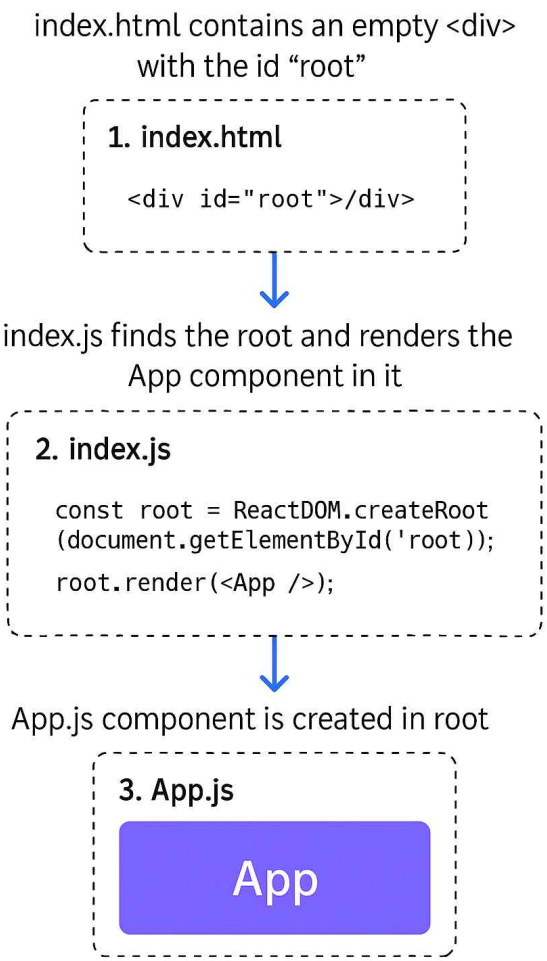




React 구조	React는 public/index.html이라는 HTML 껍데기 하나만 있고, 진짜 화면 내용은 src/App.js처럼 자바스크립트(컴포넌트)로 채워지는 구조	
동작순서	1 index.html	라는 빈 공간이 있음 (HTML 껍데기)

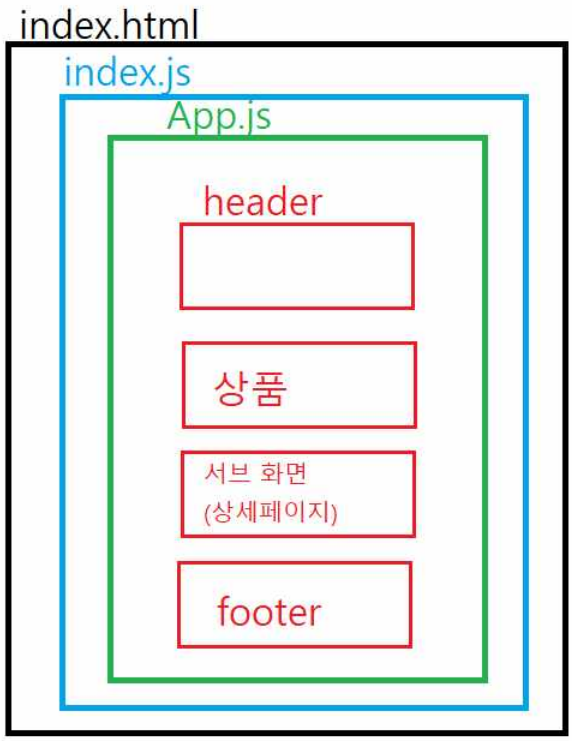
	2 index.js	root라는 그 빈 공간을 찾아서, App 컴포넌트를 거기다가 넣어줌
		const root = ReactDOM.createRoot(document.getElementById('root'));
		root.render(<App />);

	3 App.js	실제로 보이는 화면 내용을 여기서 만들고 구성함
index.html은 틀이고, 실제 내용은 App.js부터 컴포넌트로 구성		

그림으로 이해 1	

그림으로 이해 2

기존처럼 html을 여러개 만드는게 아니라 하나의 html안에 한페이지에 컴포넌트로 채움(SPA)



App.js 에서
모든 컴포넌트를 다 넣고 SPA

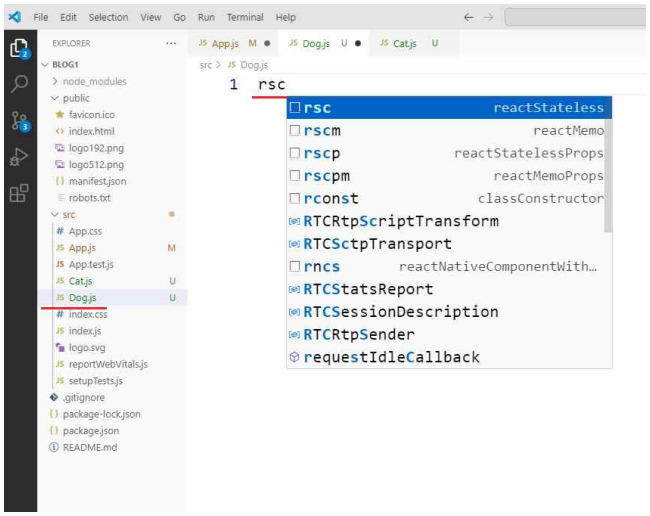
조건이 되면
컴포넌트가 보였다 안보였다 제어한다.

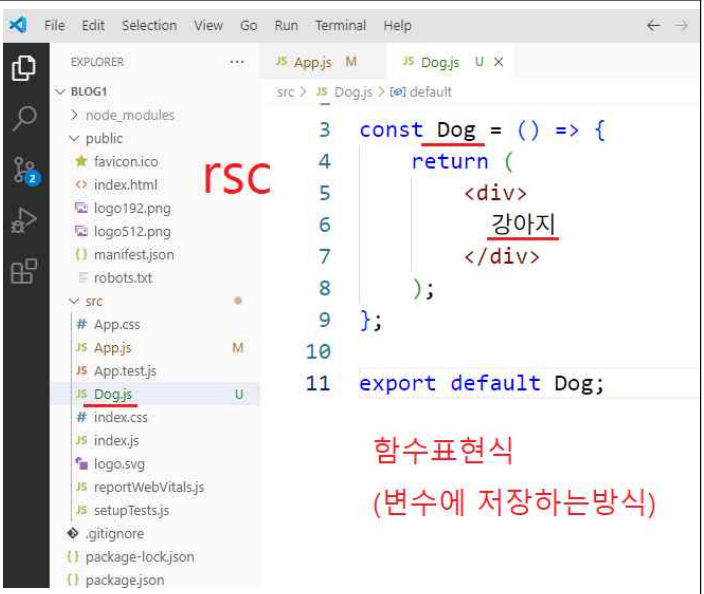
App.js는 모든 컴포넌트의 중심이고,
조건(if, useState, 라우터)에 따라 화면을 바꿔가며 보여주는 SPA 구조

SPA의 단점	문제는 서버에서 렌더링하면 검색이 잘되는데 리엑트로 만든 SPA는 검색에 노출이 잘 안되서 Next.js를 많이 사용한다. Next.js는 TypeScirpt가 기본이다
	일반적인 React(SPA)는 모든 화면을 브라우저에서(JavaScript로) 그리는 방식이다. 그래서 검색 엔진(구글, 네이버)이 페이지 내용을 제대로 읽지 못할 수 있다. 반면에 Next.js는 페이지를 서버에서 미리 만들어서(SSR) 브라우저에 보내준다. 그래서 검색 엔진에도 잘 노출되고, SEO(검색 최적화)에 유리하다. 또한, Next.js는 기본 설정에 TypeScript도 쉽게 쓸 수 있게 되어 있어서, 대규모 프로젝트나 팀 개발에 많이 사용되고 있어요.
SPA	Single Page Application 하나의 페이지에서 모든 기능이 동작하는 앱 처음에 HTML 하나만 로딩하고, 이후에는 페이지를 다시 새로고침하지 않고 필요한 부분만 JavaScript로 바꿔서 보여주는 방식 한 페이지 안에서 필요한 화면만 바꿔 보여주는 웹앱
SSR	Server Side Rendering (서버 사이드 렌더링) 사용자가 요청하면 서버가 HTML을 만들어 보내줌 검색엔진이 쉽게 읽을 수 있어 SEO에 유리함
SEO	Search Engine Optimization 검색 엔진에 잘 노출되게 만드는 기술, 검색엔진 최적화 Search → 검색 Engine → 검색엔진 (구글, 네이버 등) Optimization → 잘 보이도록 최적화

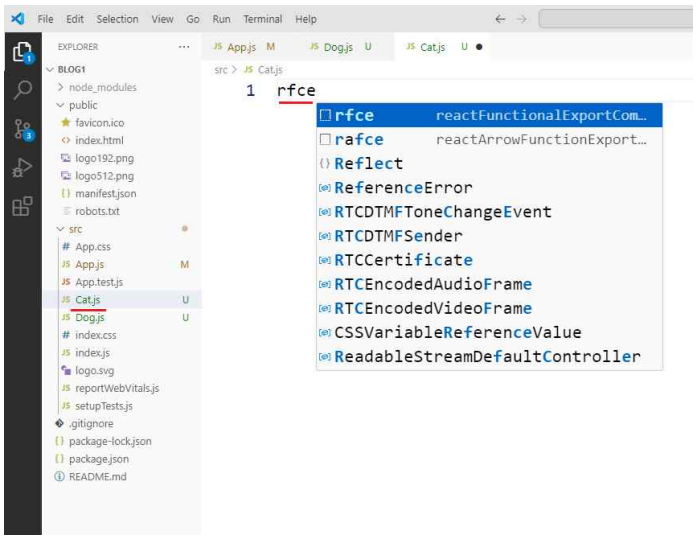
리엑트에서 함수 만드는 방법 – 대문자로 컴포넌트를 만들고 src폴더 아래 > Dog.js 만들기

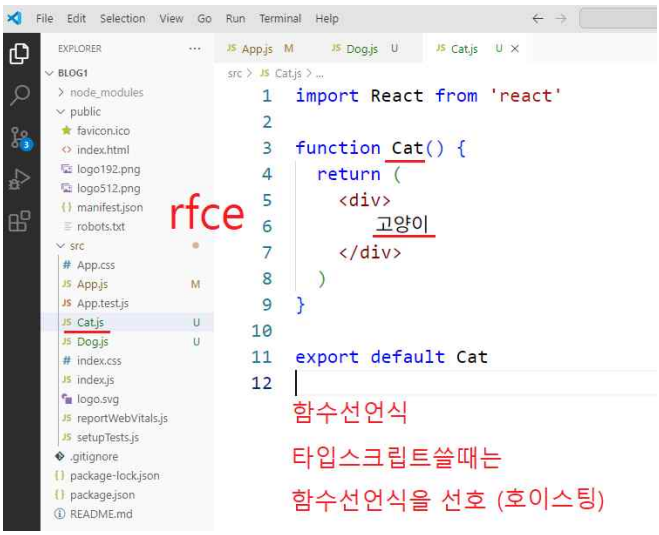
함수표현식 1 (변수에 담아쓰는 방법)



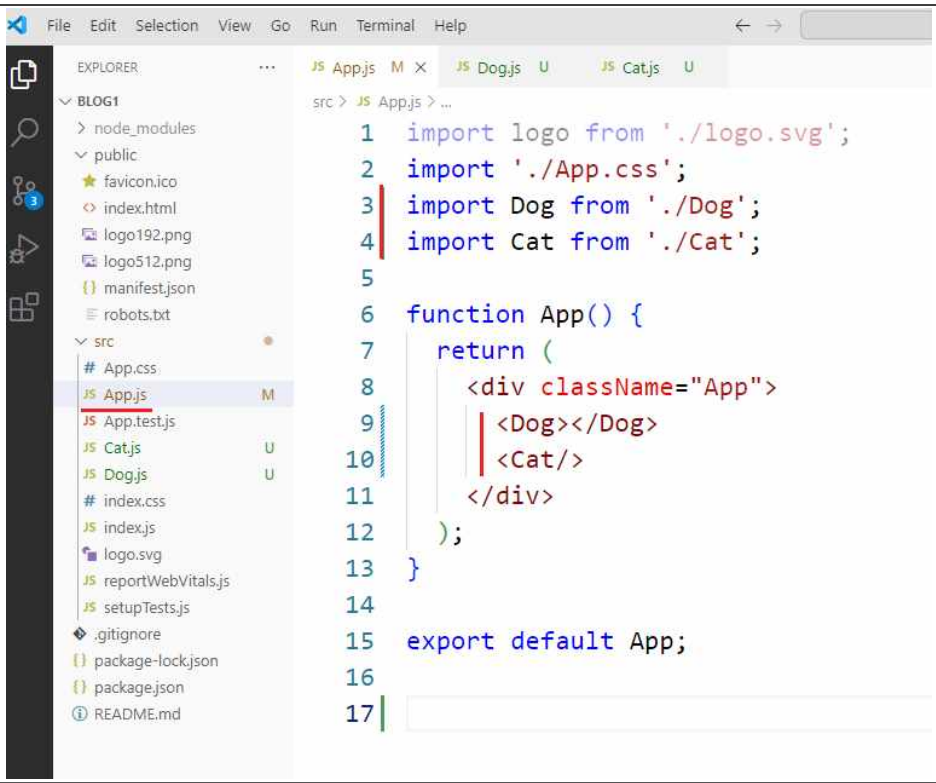


함수선언식2 – 타입스크립트에서는 함수선언식을 선호 (호이스팅)



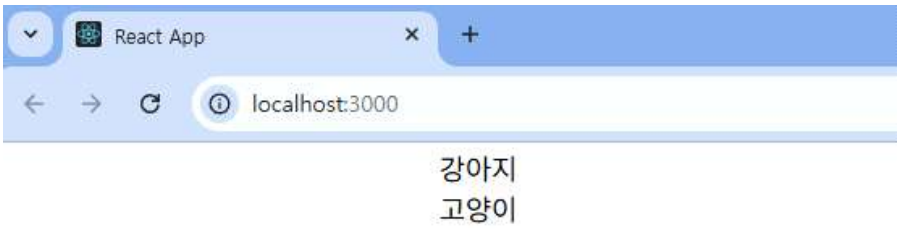


App.js 에 컴포넌트 가져오기

<table>
<tr><td>App.js</td><td> 
</td></tr>
<tr><td>App.js</td><td>import './App.css';
import Dog from './Dog';
import Cat from './Cat';

function App() {
return (

<Dog></Dog>
<Cat/>

);
}
export default App;</td></tr>
<tr><td>터미널</td><td>npm start</td></tr>
<tr><td>브라우저 확인</td><td> 
</td></tr>
<tr><td>터미널
프론트 서버
종료</td><td>Ctrl + C</td></tr>
</table>

html 구조

header	헤더를 의미, 머리말
nav	네비게이션을 의미 / 메뉴,로그인 서브메뉴
section	여러 중심 내용을 감싸는 공간을 의미, 웹컨텐츠들을 그룹으로 묶어주는 역할을 담당
article	글자가 많이 들어가는 부분을 의미, 본문내용 웹페이지 상에서의 실제 내용을 의미
figure	이미지사진 등
aisde	사이드에 위치하는 공간을 의미 (퀵메뉴, 서브메뉴)
footer	푸터를 의미, 웹사이트의 저작권 정보나 저작권표기

시맨틱 – 웹페이지

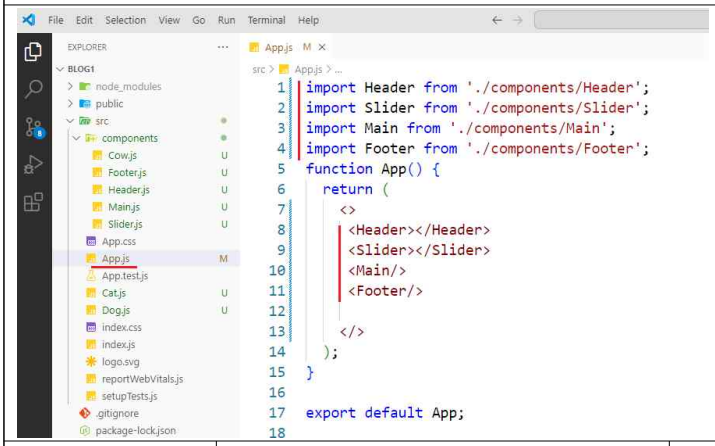
Header

Sliser

Main

Footer

컴포넌트를 이용한 웹 페이지 – App.js



폴더	파일
components	Header.js
	Slider.js
	Main.js
	Footer.js

Header.js

단축키
rsc

```
import React from 'react';

const Header = () => {
    return (
        <div>
            헤더
        </div>
    );
};

export default Header;
```

Slider.js

단축키
rsc

```
import React from 'react';

const Slider = () => {
    return (
        <div>
            슬라이더
        </div>
    );
};

export default Slider;
```

Main.js

단축키
rfce

```
import React from 'react'

function Main() {
  return (
    <div>
      메인
    </div>
  )
}

export default Main
```

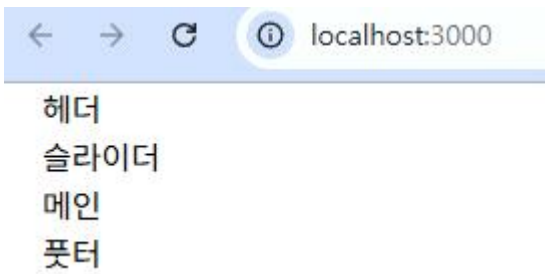
Footer.js

단축키
rfce

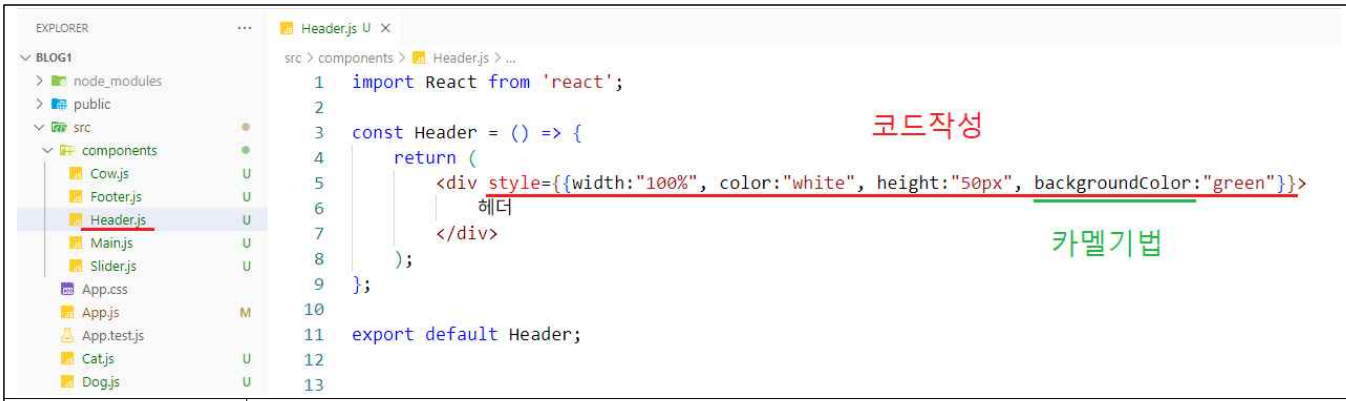
```
import React from 'react'

function Footer() {
  return (
    <div>
      풋터
    </div>
  )
}

export default Footer
```

App.js

```
import Header from './components/Header';
import Slider from './components/Slider';
import Main from './components/Main';
import Footer from './components/Footer';

function App() {
  return (
    <>
      <Header></Header>
      <Slider></Slider>
      <Main/>
      <Footer/>

    </>
  );
}

export default App;
```



Header.js 컴포넌트에 Style 입히기 – 1 인라인 스타일 style={{}} 머스테치(Mustache)



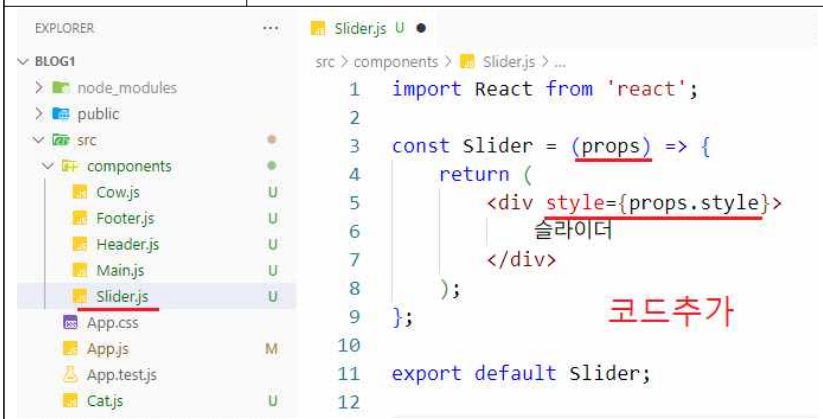
components > Header.js	import React from 'react';
	
	const Header = () => {
	return (
	<div style={{width:"100%", color:"white", height:"50px", backgroundColor:"green"}}>
	헤더
	
	);
	};
	
	export default Header;

터미널 확인 npm start	 슬라이더 메인 풋터
style={{ }} 머스테치(Mustache)	JavaScript 객체를 표현하는 JSX 문법 바깥 { ... } → JSX 안에서 JavaScript 표현식을 넣기 위한 괄호 안쪽 { ... } → JavaScript의 객체 리터럴 (스타일 정의용) JSX 안에 객체를 넣는다.

slider.js 컴포넌트에 Style 입히기 – 2 props로 스타일 적용하기

components > App.js

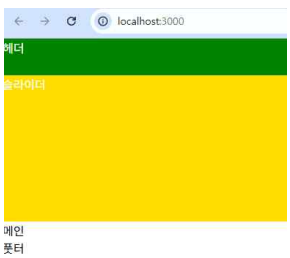
```
import Header from './components/Header';
import Slider from './components/Slider';
import Main from './components/Main';
import Footer from './components/Footer';
function App() {
  return (
    <>
      <Header/>
      <Slider style={{width:"100%", color:"white", height:"200px", backgroundColor:"gold"}}></Slider>
      <Main/>
      <Footer/>
    </>
  );
}
export default App;
```



Slider.js

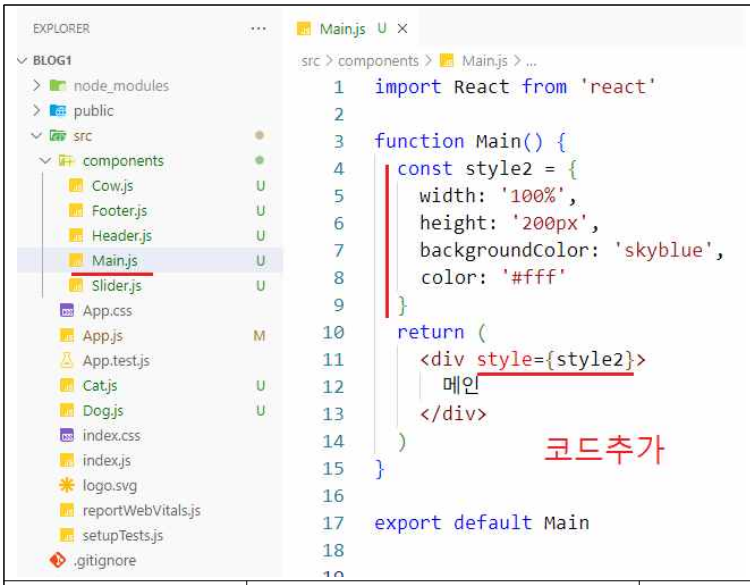
```
import React from 'react';

const Slider = (props) => {
    return (
        <div style={props.style}>
            슬라이더
        </div>
    );
};

export default Slider;
```

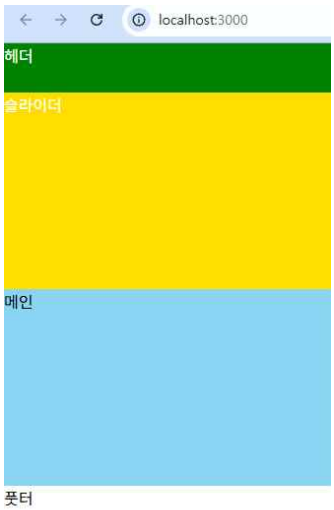
터미널 확인
npm start



style={{ ... }}	부모 컴포넌트(App)에서 스타일을 자식(Slider)에게 전달함
props	자식 컴포넌트가 전달받은 속성을 담는 객체
rops.style	부모가 보낸 스타일 객체를 꺼내어 에 적용함
props란	props는 properties(속성)의 줄임말로, 부모 컴포넌트가 자식 컴포넌트에게 값을 전달할 때 사용하는 객체 props는 자식 컴포넌트가 외부 데이터를 전달받는 방법

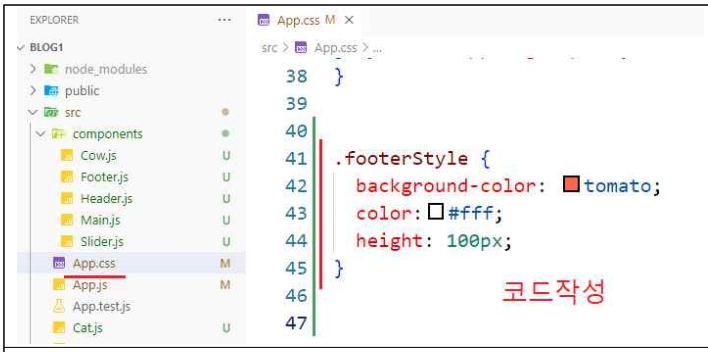
Main.js 컴포넌트에 Style 입히기 – 3 스타일 변수

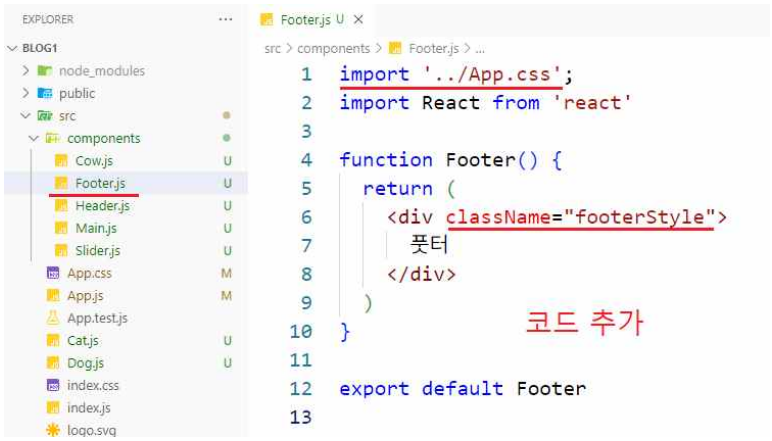


components > Main.js	import React from 'react'
	
	function Main() {
	//라인 스타일을 객체 형태로 정의
	const style2 = {
	width: '100%',
	height: '200px',
	backgroundColor: 'skyblue',
	color: '#fff'
	}
	return (
	
	메인
	
	)
	}
	
	export default Main
		터미널 확인
		npm start
			

style2	스타일 속성을 가진 객체		
style={style2}	style={객체} 구조는 JSX에서 인라인 스타일을 적용하는 기본 문법 (JSX에서 style에 객체를 넣는 방식)		

Footer.js 컴포넌트 - 4 외부 CSS 클래스 적용하기

<table>
<tr>
<td> 
</td>
<td>App.css – 맨아래 추가

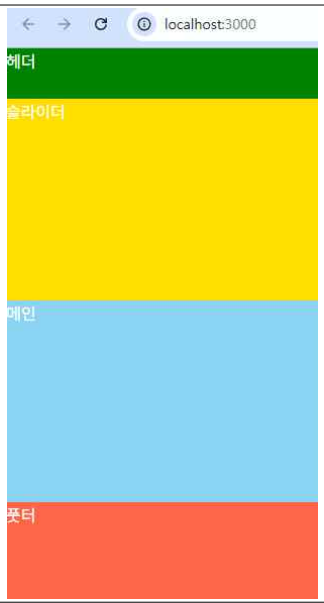
.footerStyle {
background-color: tomato;
color:#fff;
height: 100px;
}</td>
</tr>
<tr>
<td> 
</td>
<td>Footer.js

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

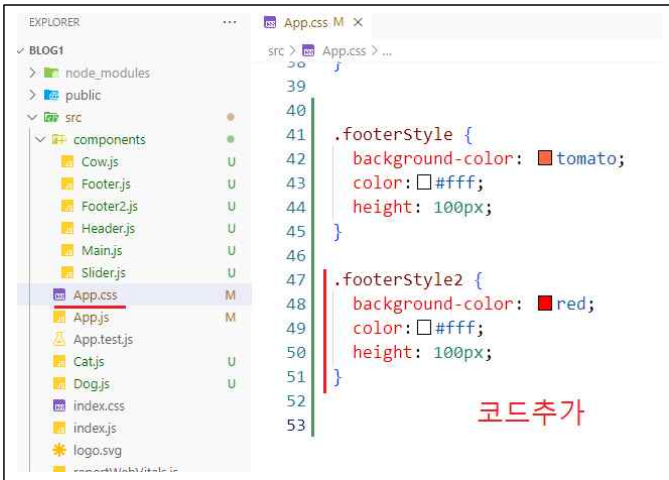
function Footer() {
return (

풋터

)
}

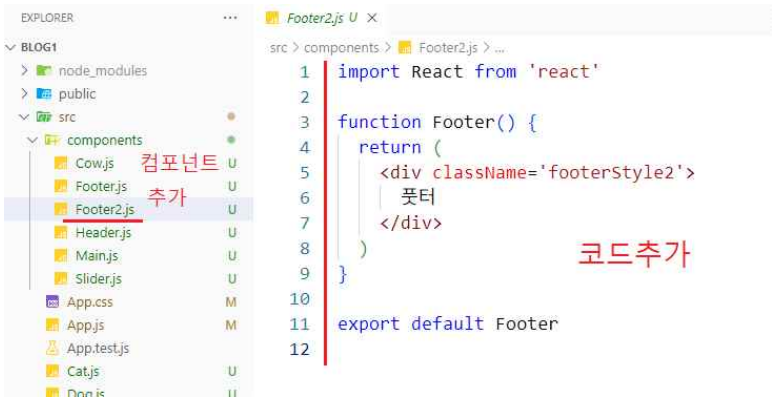
export default Footer</td>
</tr>
<tr>
<td>터미널 확인
npm start</td>
<td> 
</td>
</tr>
<tr>
<td>className 속성</td>
<td>Footer 컴포넌트에서 외부 CSS 파일(App.css)의 클래스를 불러와서,
className 속성을 통해 JSX 요소에 스타일을 적용하는 방식</td>
</tr>
</table>

Footer.js 컴포넌트2 - 4 외부 CSS 클래스 적용하기2



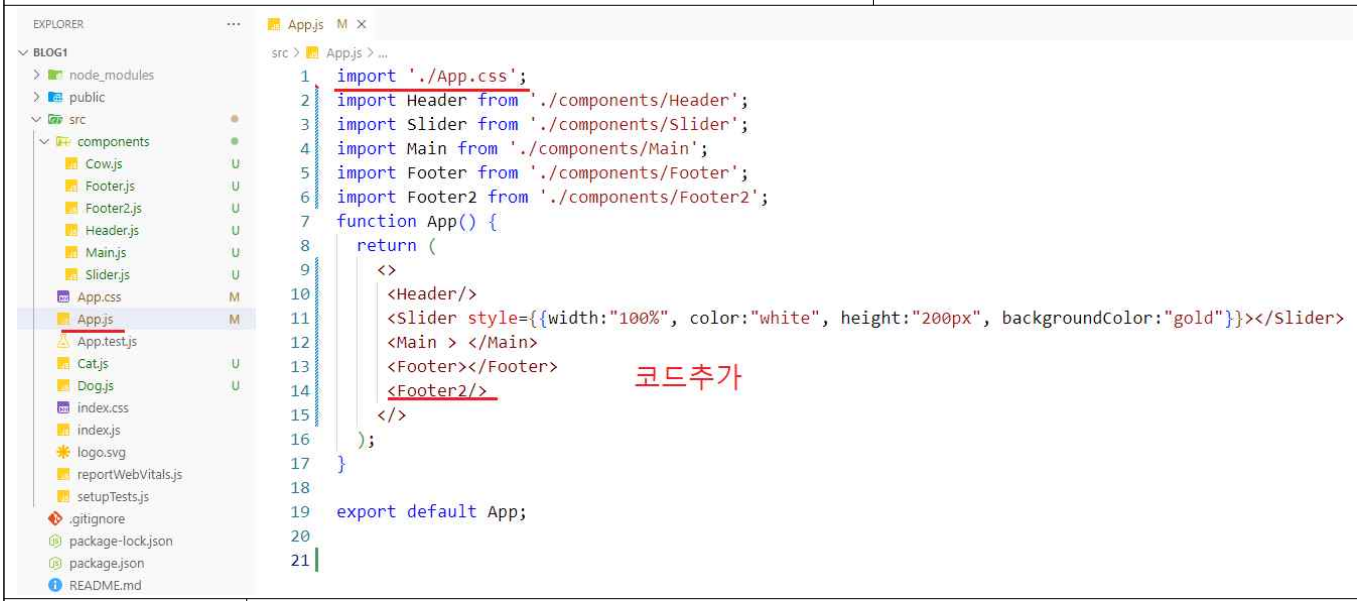
App.css – 맨아래 추가

```
.footerStyle {
  background-color: tomato;
  color:#fff;
  height: 100px;
}

.footerStyle2 {
  background-color: red;
  color:#fff;
  height: 100px;
}
```



Footer2.js

```
import React from 'react'

function Footer() {
  return (
    <div className='footerStyle2'>
      푸터
    </div>
  )
}

export default Footer
```

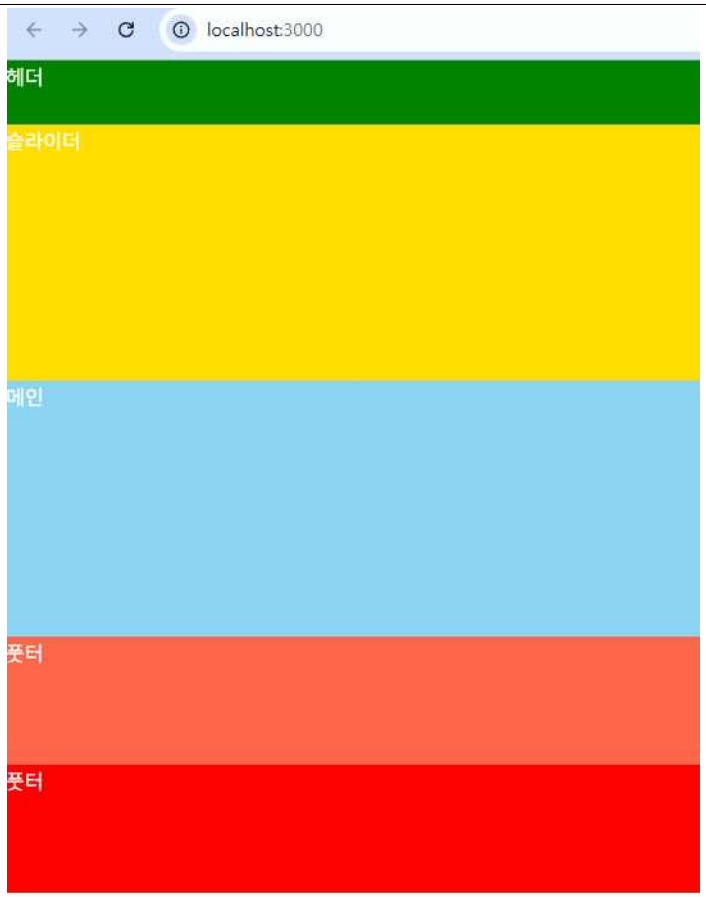


App.js

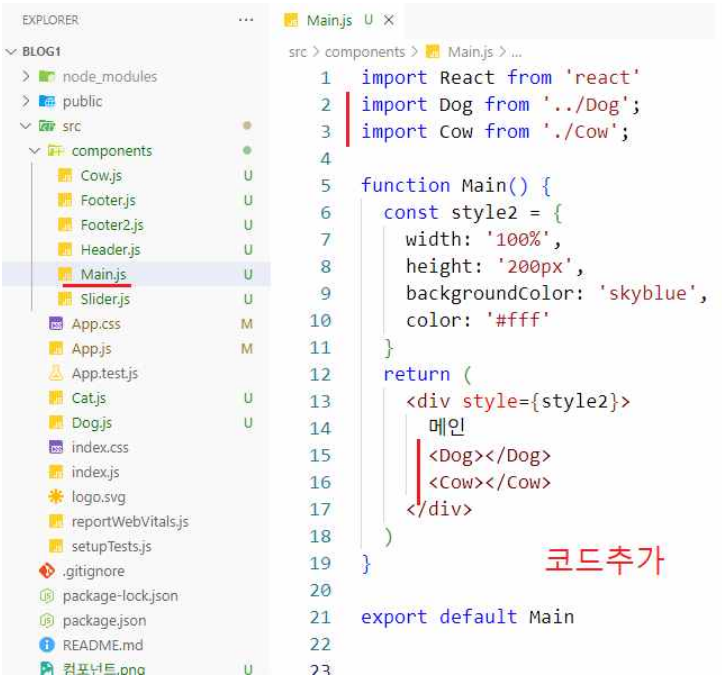
코드추가

전역 범위(global)

```
import './App.css';
import Header from './components/Header';
import Slider from './components/Slider';
import Main from './components/Main';
import Footer from './components/Footer';
import Footer2 from './components/Footer2';
function App() {
  return (
    <>
      <Header/>
      <Slider style={{width:"100%", color:"white", height:"200px", backgroundColor:"gold"}}></Slider>
      <Main > </Main>
      <Footer></Footer>
      <Footer2/>
    </>
  );
```

	} export default App;
터미널 확인 npm start	
className 속성	Footer 컴포넌트에서 외부 CSS 파일(App.css)의 클래스를 불러와서, className 속성을 통해 JSX 요소에 스타일을 적용하는 방식

Main 컴포넌트에 자식 컴포넌트(Dog, Cow) 포함하기



Main.js

```
import React from 'react'
import Dog from '../Dog';
import Cow from './Cow';

function Main() {
  const style2 = {
    width: '100%',
    height: '200px',
    backgroundColor: 'skyblue',
    color: '#fff'
  }
  return (
    <div style={style2}>
      메인
      <Dog></Dog>
      <Cow></Cow>
    </div>
  )
}

export default Main
```

터미널 확인
npm start