

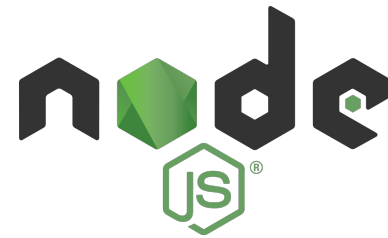
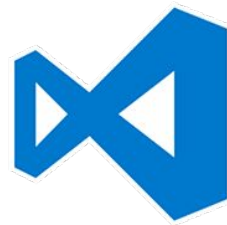


04. 프로젝트 준비

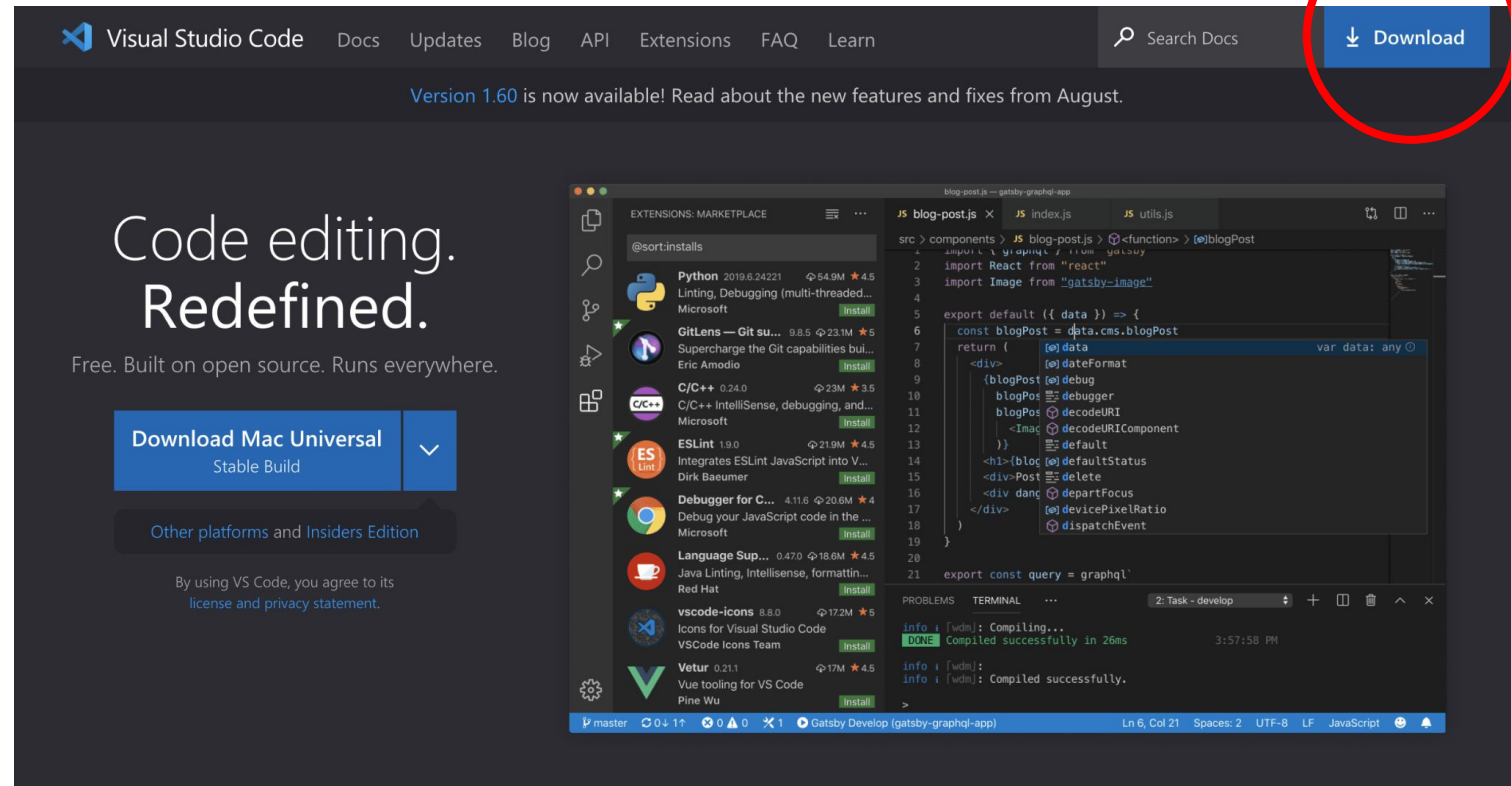


1. 에디터 설치 및 환경세팅
 - VSC, Node, Sass 설치
 - Live Server, Watch Sass 세팅
2. 프로젝트 생성하기
3. 개발범위 파악하기

1. 에디터 설치 및 환경세팅



VSC(Visual Studio Code) 설치

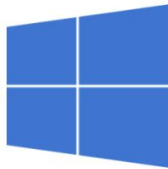


출처 : <https://code.visualstudio.com/>

사용하고 있는 운영체제에 맞게 다운로드

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



↓ Windows

Windows 7, 8, 10

User Installer	64 bit	32 bit	ARM
System Installer	64 bit	32 bit	ARM
.zip	64 bit	32 bit	ARM



↓ .deb

Debian, Ubuntu

↓ .rpm

Red Hat, Fedora, SUSE

.deb	64 bit	ARM	ARM 64
.rpm	64 bit	ARM	ARM 64
.tar.gz	64 bit	ARM	ARM 64

Snap Store



↓ Mac

macOS 10.11+

.zip Universal Intel Chip Apple Silicon

출처 : <https://code.visualstudio.com/Download>

Node 설치

다운로드

최신 LTS 버전: 14.17.6 (includes npm 6.14.15)

플랫폼에 맞게 미리 빌드된 Node.js 인스톨러나 소스코드를 다운받아서 바로 개발을 시작하세요.

LTS
대다수 사용자에게 추천

현재 버전
최신 기능


Windows Installer
node-v14.17.6-x86.msi


macOS Installer
node-v14.17.6.pkg


Source Code
node-v14.17.6.tar.gz

Windows Installer (.msi)

Windows Binary (.zip)

macOS Installer (.pkg)

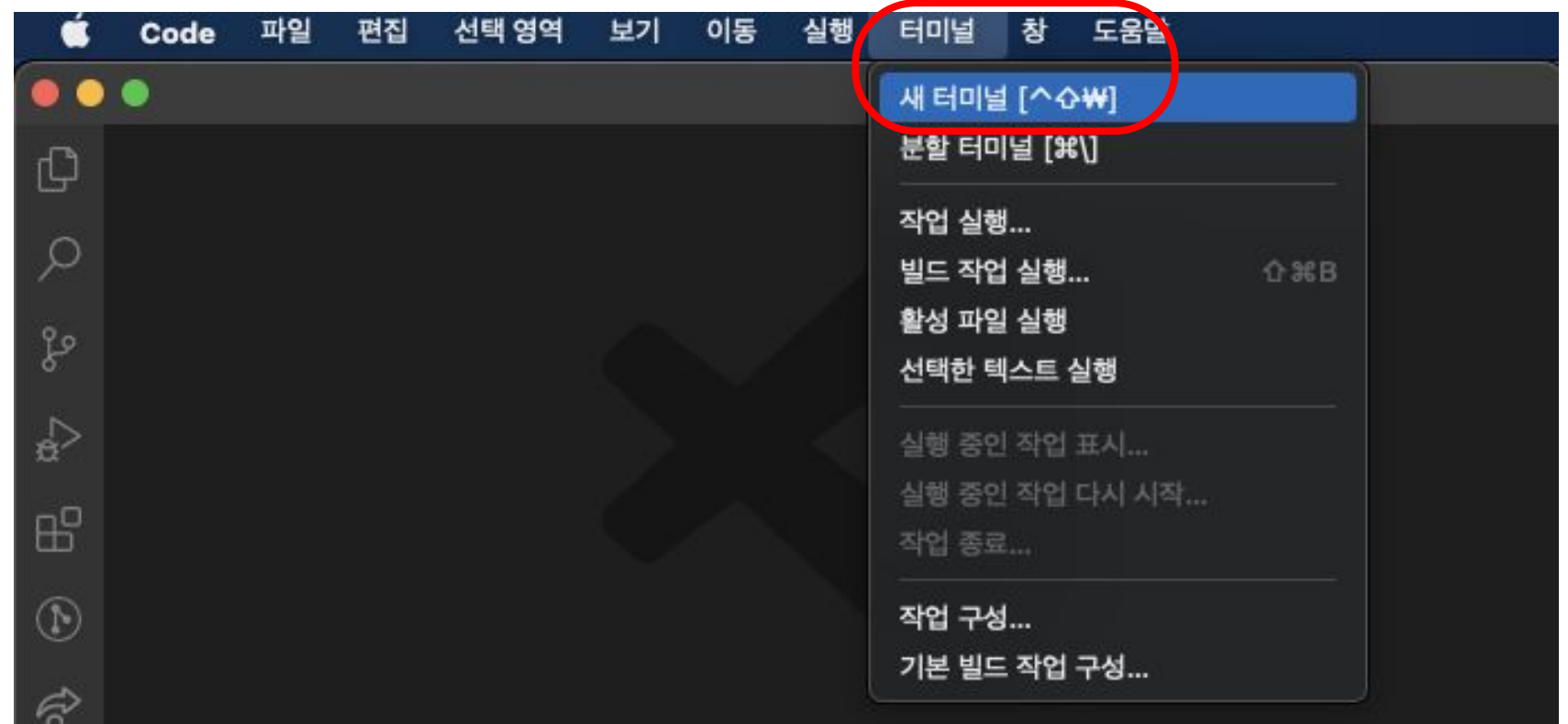
macOS Binary (.tar.gz)

32-bit	64-bit
32-bit	64-bit
64-bit	
64-bit	

출처 : <https://nodejs.org/ko/download>

VSC > Terminal 열기

터미널(Terminal) > 새 터미널(New Terminal) 열기



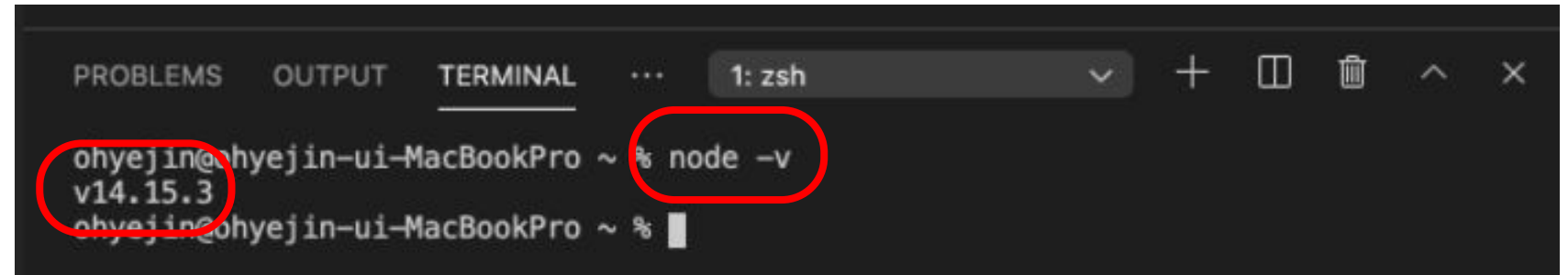
VSC > Terminal 열기

터미널에서 **node -v** 확인

안내. 터미널 입력 방법

\$ <- 달러 모양은 입력하지 않습니다.

\$ <- 달러로 시작하는 코드는 터미널(Terminal)에 입력하라는 뜻입니다.



```
PROBLEMS OUTPUT TERMINAL ... 1: zsh
ohyejin@ohyejin-ui-MacBookPro ~ % node -v
v14.15.3
ohyejin@ohyejin-ui-MacBookPro ~ %
```

(자신이 설치한 버전에 따라 버전을 나타내는 숫자를 다를 수 있습니다.)

Sass 설치 후 버전 확인하기

```
$ sudo npm i -g sass
```

```
$ sass --version
```

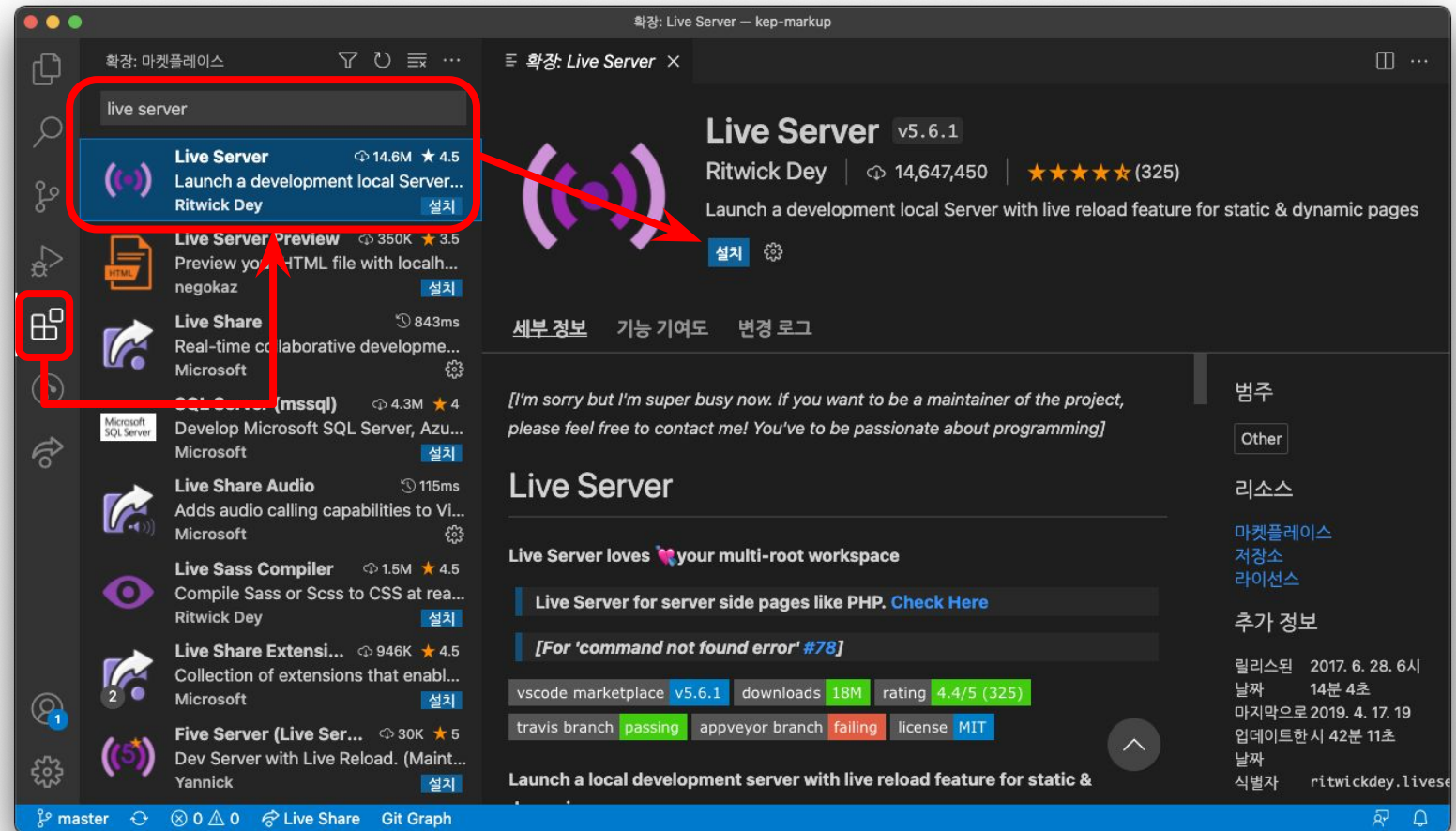
Password는 입력해도 터미널 창에 보이지 않습니다.

맥북에서 사용하고 있는 사용자 비밀번호를 입력 후 Enter를 치세요!

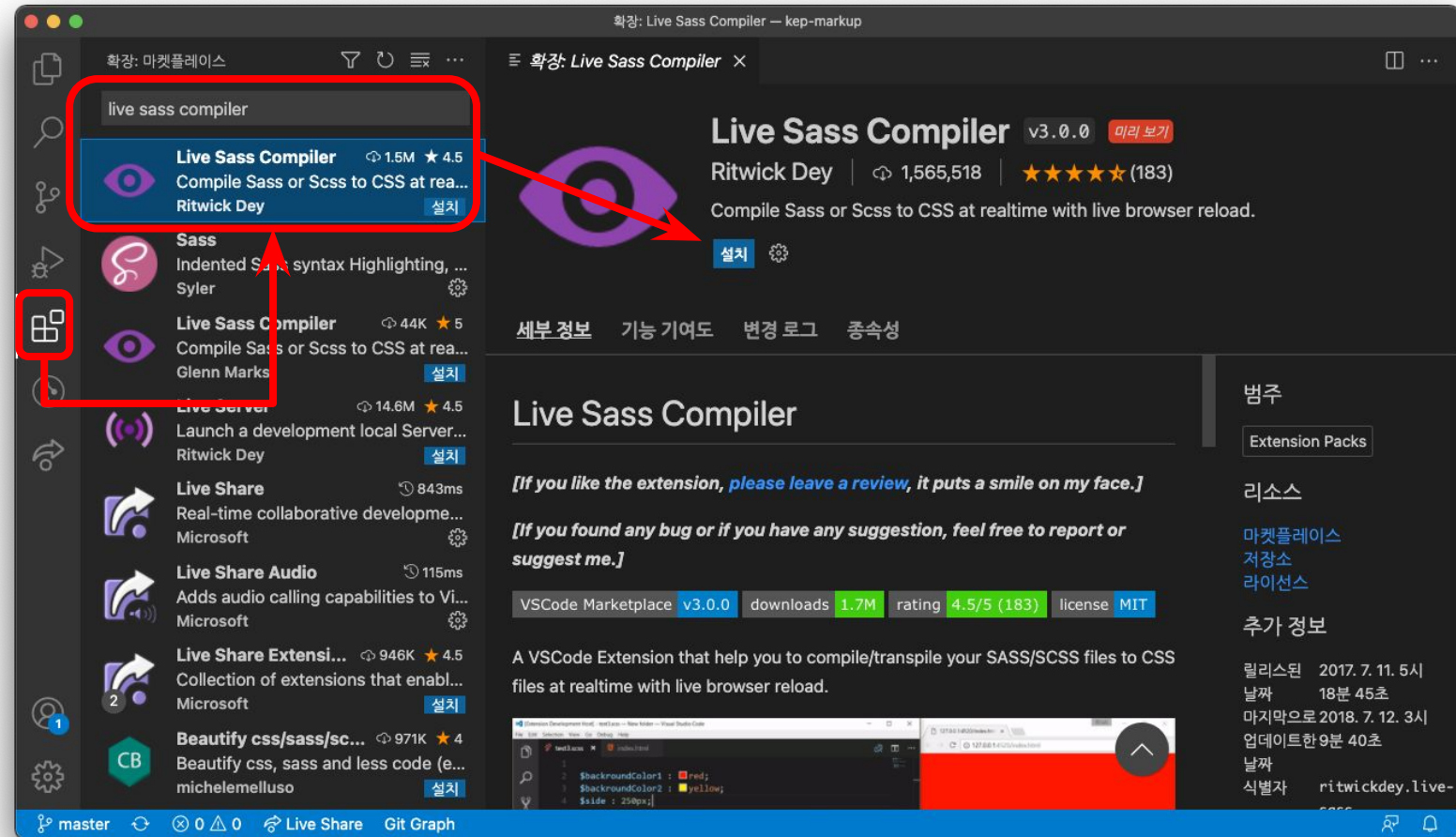
```
ohyejin@ohyejin-ui-MacBookPro ~ % sudo npm i -g sass
Password:
/usr/local/bin/sass -> /usr/local/lib/node_modules/sass/sass.js
+ sass@1.30.0
updated 3 packages in 0.558s
```

```
PROBLEMS  OUTPUT  TERMINAL  ...  1: zsh  ▾  +  [
ohyejin@ohyejin-ui-MacBookPro ~ % sass --version
1.30.0 compiled with dart2js 2.10.4
ohyejin@ohyejin-ui-MacBookPro ~ %
```

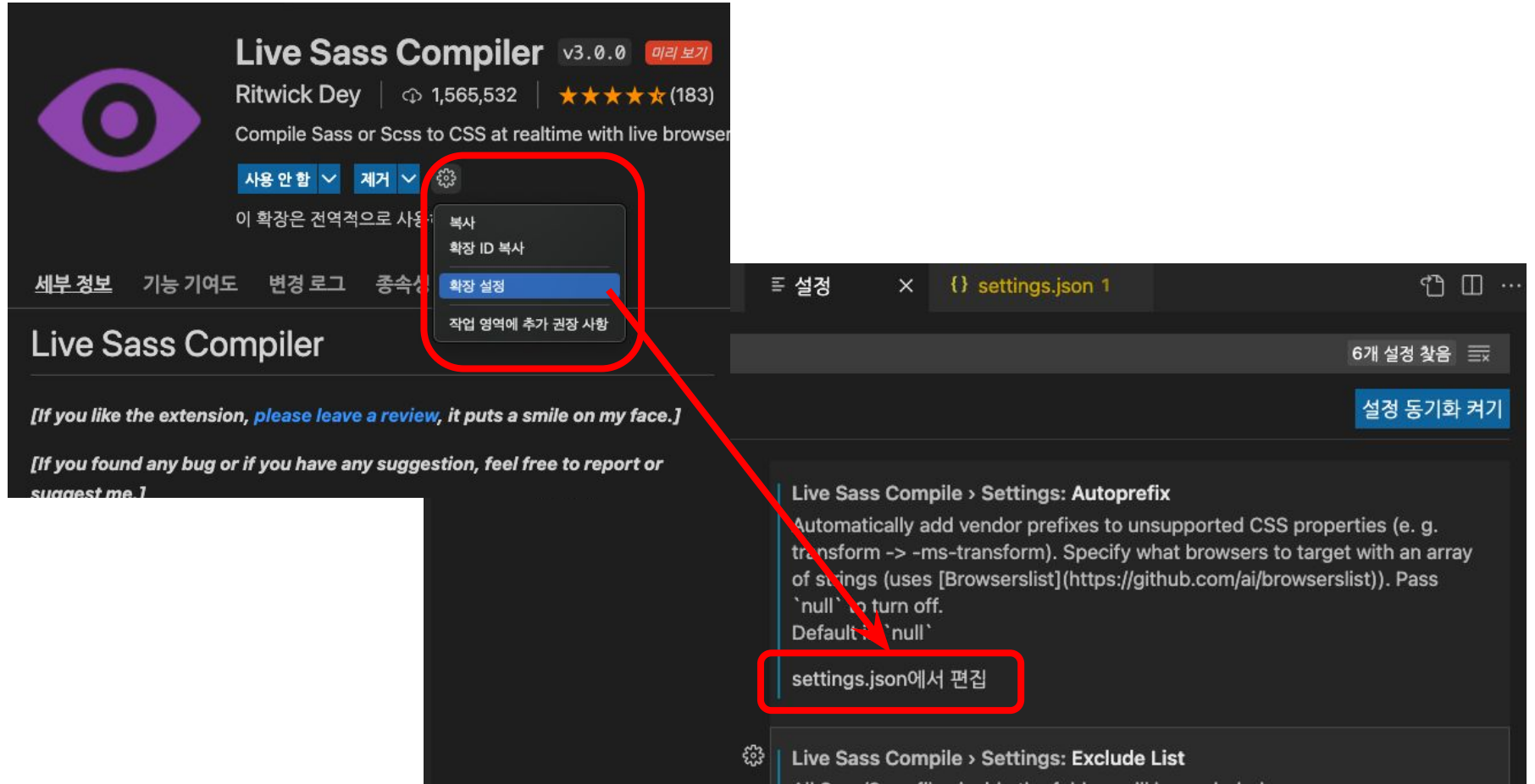

📌 마켓 플레이스 > Live Server 설치



📌 마켓 플레이스 > Live Sass Compiler 설치



Live Sass Compiler > 확장 설정 > setting.json에서 편집



The screenshot shows the Live Sass Compiler extension interface. On the left, the extension details for 'Live Sass Compiler' by Ritwick Dey are visible, including version v3.0.0 and 1,565,532 downloads. A red box highlights the '확장 설정' (Extension Settings) button. On the right, the 'Settings: Autoprefixer' configuration is shown, with a red box highlighting the 'settings.json에서 편집' (Edit in settings.json) link. A red arrow points from the '확장 설정' button to the 'settings.json에서 편집' link.

setting.json에서 편집

./ = 현재 파일의 위치

../ = 현재 파일의 부모 폴더

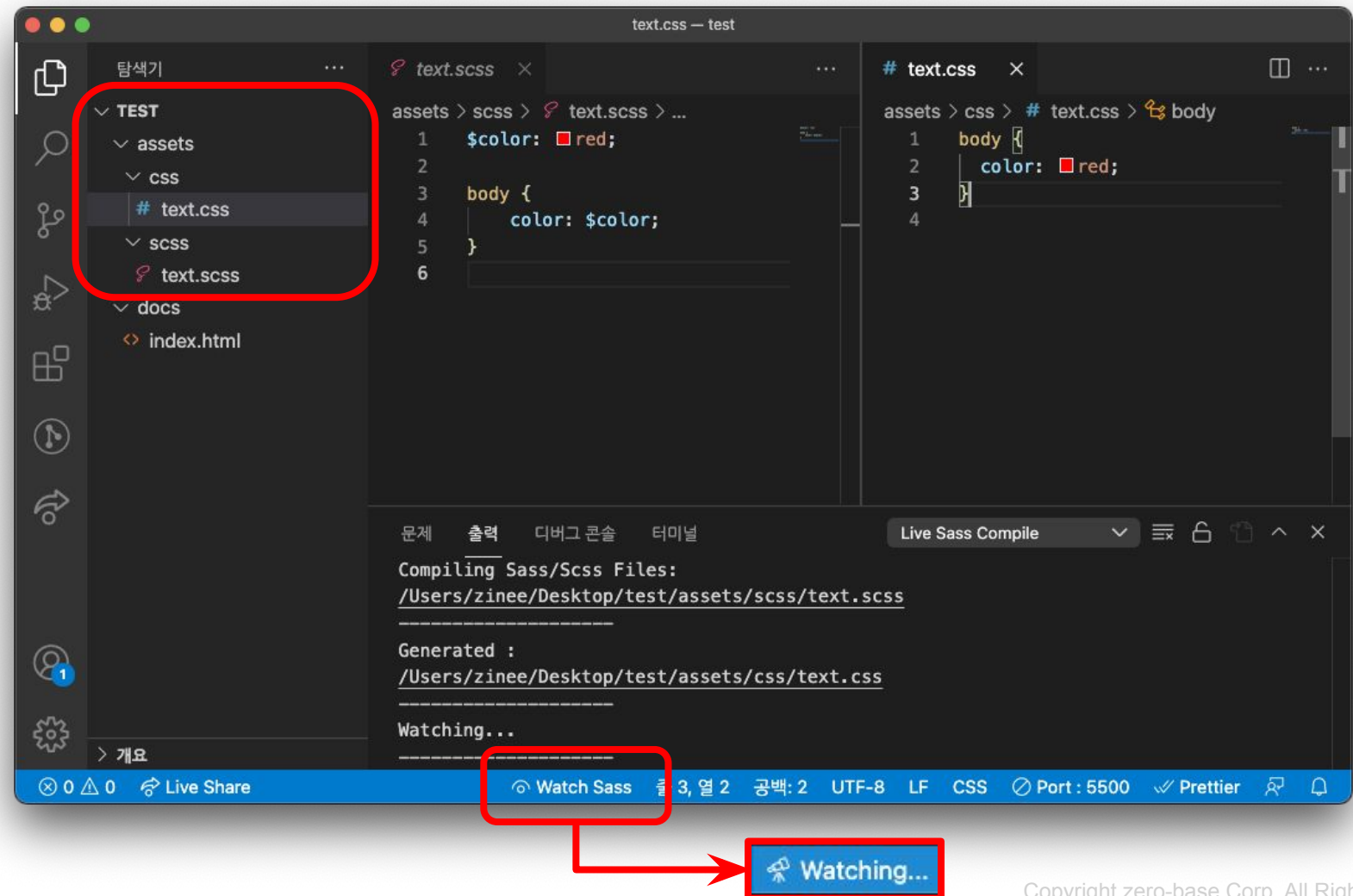


```
Users > zinee > Library > Application Support > Code > User > {} settings.json > ...
5  "liveSassCompile.settings.autoprefixer": [],
6  "liveSassCompile.settings.excludeList": [
7    "**/node_modules/**",
8    ".vscode/**",
9  ],
10 "liveSassCompile.settings.formats": [
11   {
12     "format": "expanded",
13     "extensionName": ".css",
14     "savePath": "./assets/css/"
15   }
16 ],
17 "liveSassCompile.settings.generateMap": false,
18 "git.autofetch": true,
```

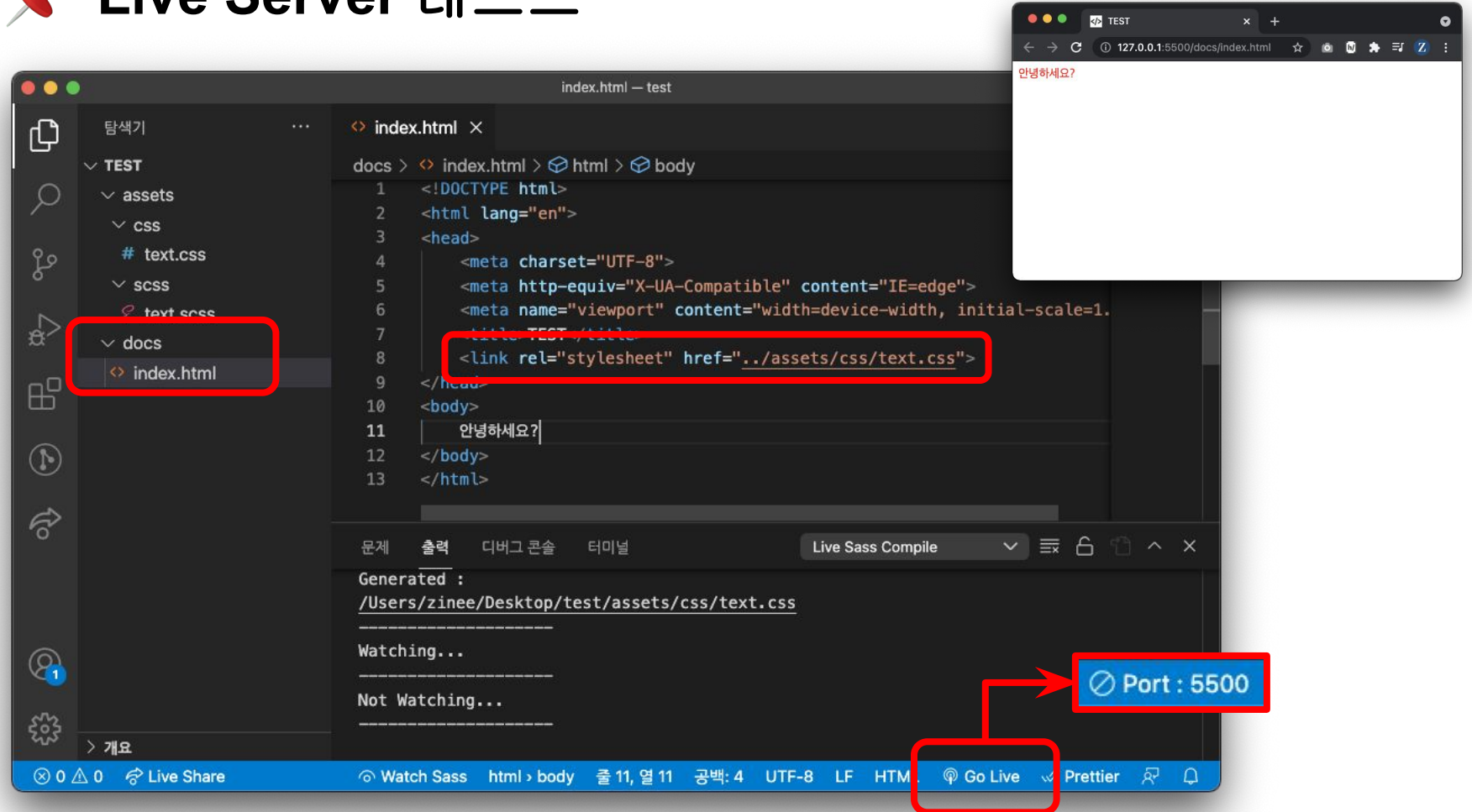
TEST

- assets
 - css
 - # text.css
 - scss
 - 🔗 text.scss
- docs
 - <> index.html

Watch Sass 테스트



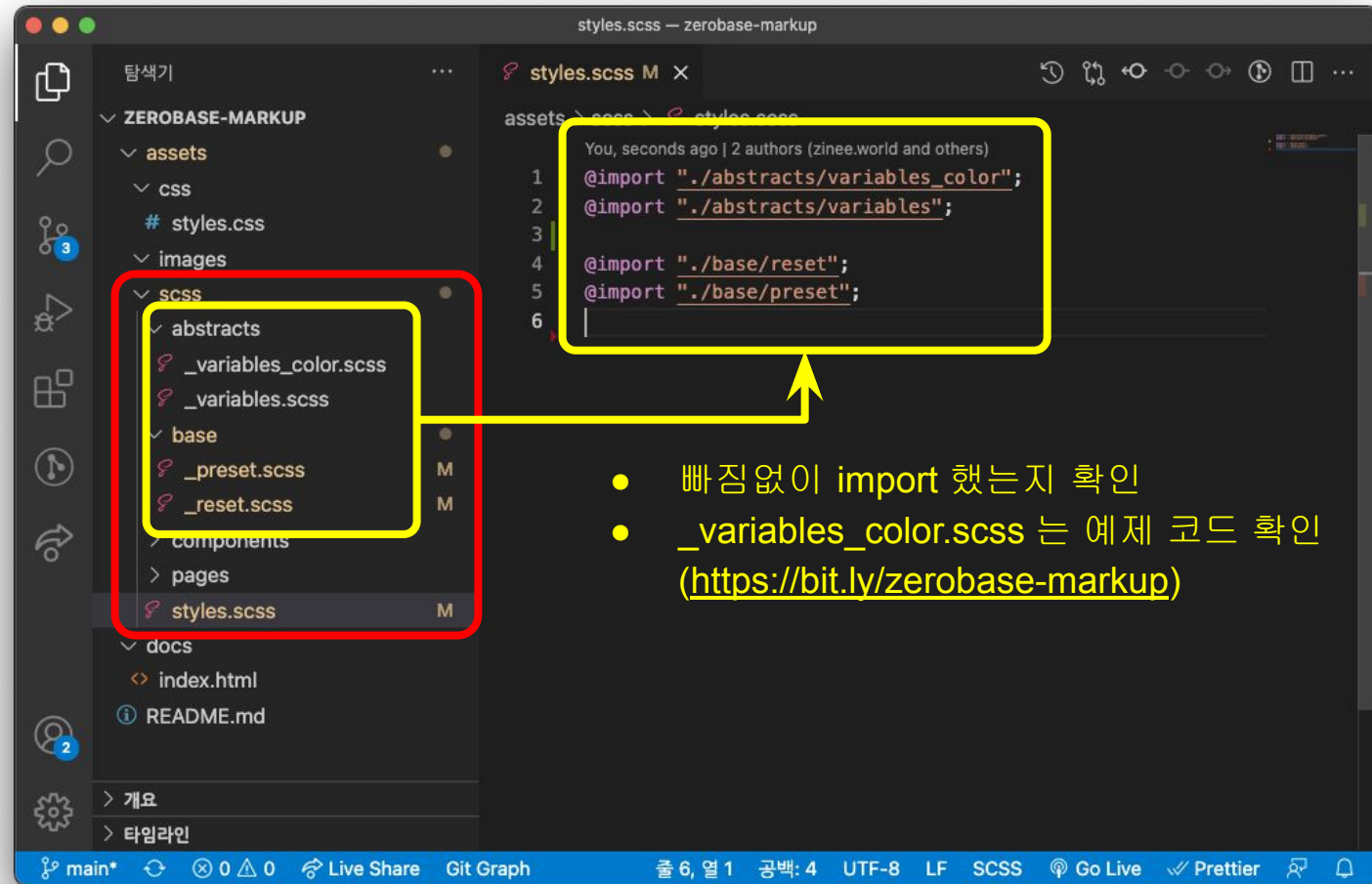
Live Server 테스트



2. 프로젝트 생성하기

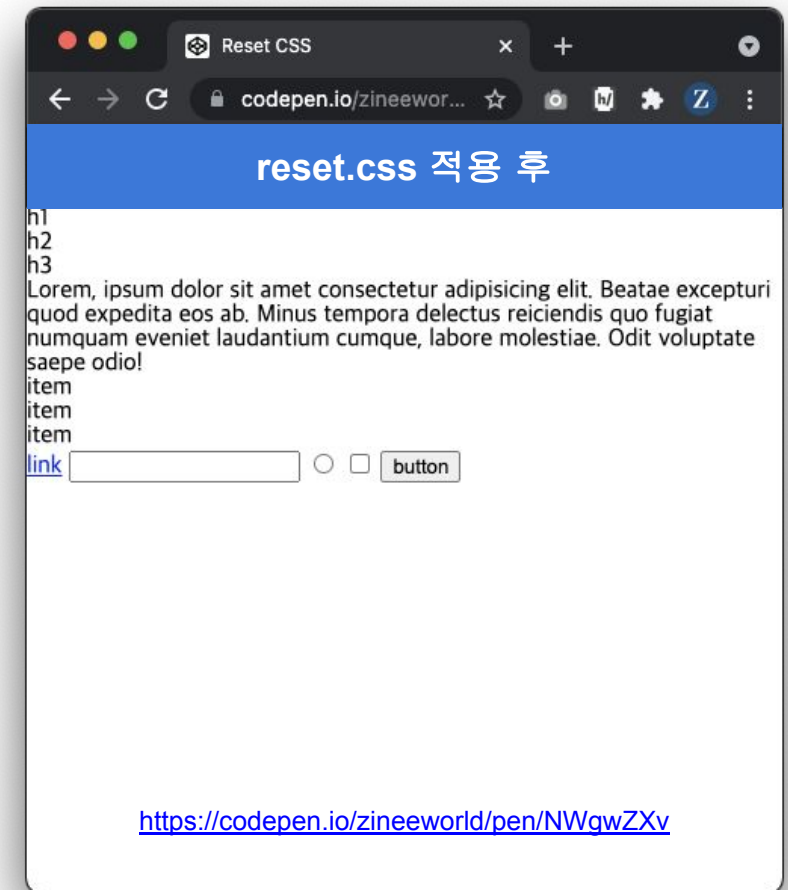
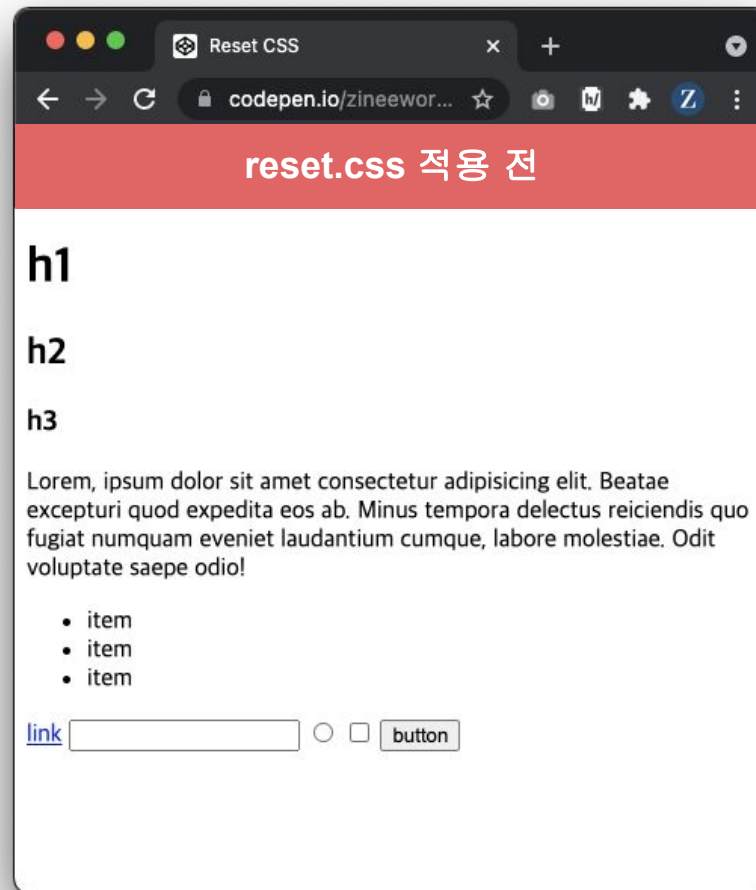
<https://bit.ly/zerobase-markup>

폴더 구조





Reset CSS



Reset CSS



```
ul {  
  display: block;  
  list-style-type: disc;  
  margin-block-start: 1em;  
  margin-block-end: 1em;  
  margin-inline-start: 0px;  
  margin-inline-end: 0px;  
  padding-inline-start: 40px;  
}
```

user agent stylesheet

reset.css는 모든 프로젝트에서 기본적으로 필요한 스타일입니다.
normalize.css, reset.css로 검색해보면 많은 보일러 플레이트 코드*가 나옵니다.
무작정 복붙해서 사용하기보다는 프로젝트에 맞게 수정해서 나만의 reset.css를
만드는 것이 바람직합니다.

*보일러 플레이트 코드 : 보일러 플레이트는 변경 없이 계속하여 재사용할 수 있는 저작품을 말한다. 확대 해석하면, 이
아이디어는 때로 "보일러 플레이트 코드"라고 부르는, 재사용 가능한 프로그램을 가리키는데 사용되기도 한다.

3. 개발범위 파악하기

개발범위 파악하기



마크업을 하는 과정


- 전체 디자인을 훑어본다
 - 대략적인 큰 레이아웃을 구상한다
 - 메인 컴포넌트를 분석한다
 - 반복되는 컴포넌트를 구분한다
- 가변적인 요소가 들어가는 항목을 확인한다
 - 이미지가 가변일 때, **no image** 처리는 어떻게 되는가?
 - 이미지 비율이 다를때 어떤 기준으로 크롭하는가?
 - 텍스트가 가변일 때 몇 줄까지 노출하는가? (말줄임)
 - 값이 없을때 0 으로 노출하는지, 항목 자체가 사라지는지? (ex. 별점)
- 기획/디자인 측에서 요청한 인터랙션이 있는지?

figma



마크업에는 정답이 없음

- 같은 화면이더라도 마크업 하는 방식은 개발자마다 모두 다름
- 모작을 하는 과정 혹은 다뤄보지 않은 레이아웃에 대한 고민이 있을 땐, 벤치마킹할 사이트의 코드를 훑어본다
- 주로 국내 서비스라면 네이버, 글로벌 서비스라면 구글이 좋은 예제

 다음 강의 : **05. 진입 페이지**