

유용한 VS Code 필수 확장 프로그램

1-1. JavaScript(ES6) Code Snippets (자바스크립트 코드 스니펫)

코딩할 때 가장 하기 싫은 부분은 바로 반복되는 코드의 입력일 것이다. 자바스크립트 코드 스니펫은 가장 인기있는 자바스크립트 코드 조각에 대한 스니펫을 제공하기도 하고 사용자별로 정의한 스니펫도 지원하니 이보다 더 좋을 순 없을 것이다.

지원되는 언어(파일 확장자)는 다음과 같다.

- JavaScript (.js)
- TypeScript (.ts)
- JavaScript React (.jsx)
- TypeScript React (.tsx)
- Html (.html)
- Vue (.vue)

[인스톨방법과 제공되는 스니펫이 궁금하다면 아래 링크](#)

[참고](https://marketplace.visualstudio.com/items?itemName=xabikos.JavaScriptSnippets) <https://marketplace.visualstudio.com/items?itemName=xabikos.JavaScriptSnippets>

User Snippets(사용자 코드조각) 사용법

1. File->Preferences->User Snippets
 2. 이미 작성되었거나 새롭게 만들 스니펫 파일명을 입력
 3. 다음 화면이 뜨면 어떻게 스니펫을 작성해야 하는지 설명이 기본적으로 들어있으니 참고한다.
-

```

1  // Place your javascript workspace snippets here. Each snippet is defined under a
2  // snippet name and has a scope, prefix, body and
3  // description. Add comma separated ids of the languages where the snippet is
4  // applicable in the scope field. If scope
5  // is left empty or omitted, the snippet gets applied to all languages. The prefix is
6  // what is
7  // used to trigger the snippet and the body will be expanded and inserted. Possible
8  // variables are:
9  // $1, $2 for tab stops, $0 for the final cursor position, and ${1:label}, ${2:another}
10 // for placeholders.
11 // Placeholders with the same ids are connected.
12 // Example:
13 // "Print to console": {
14 //   "scope": "javascript,typescript",
15 //   "prefix": "log",
16 //   "body": [
17 //     "console.log('$1');",
18 //     "$2"
19 //   ],
20 //   "description": "Log output to console"
21 // }

```

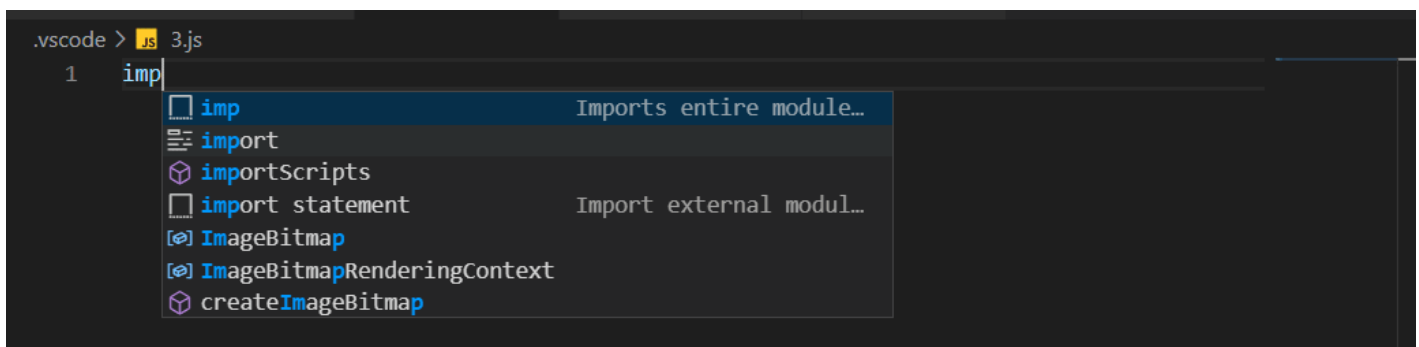
한글로 간단히 풀면,

```

19 "스니펫 파일명": {
20   "scope": "사용할 프로그래밍 언어",
21   "prefix": "스니펫을 사용할때 쓸 약어",
22   "body": [
23     "입력할 내용1",
24     "입력할 내용2"
25   ],
26   "description": "이 스니펫에 대한 설명"

```

4. 스니펫 적용-작성된 스니펫을 사용하려면, 아래와 같이 화면에서 지정해둔 약어를 타이핑한 후 사용하면 된다. (아래 예는 스니펫 약어를 'imp'로 설정한 경우)



Tip: 스니펫 형식으로 작성을 대신해주는 웹사이트 : <https://snippet-generator.app/>

1-2. ES7 React / Redux / GraphQL / React-Native snippets

ES7 upgrade! VS Code를 이용한 React 개발자라면 반드시 가지고 있어야 할 확장 중 하나일 것이다. 키워드 몇 가지를 가지고 간편하게 코드작성을 할 수 있다.

지원되는 스니펫 전체 목록이 궁금하다면 여기 링크 <https://marketplace.visualstudio.com/items?itemName=dsznajder.es7-react-js-snippets>

React	
Prefix	Method
imr→	import React from 'react'
imrd→	import ReactDOM from 'react-dom'
imrc→	import React, { Component } from 'react'
imrcp→	import React, { Component } from 'react' & import PropTypes from 'prop-types'
imrpc→	import React, { PureComponent } from 'react'
imrpcp→	import React, { PureComponent } from 'react' & import PropTypes from 'prop-types'
imrm→	import React, { memo } from 'react'
imrmp→	import React, { memo } from 'react' & import PropTypes from 'prop-types'
impt→	import PropTypes from 'prop-types'
imrr→	import { BrowserRouter as Router, Route, NavLink } from 'react-router-dom'
imbr→	import { BrowserRouter as Router } from 'react-router-dom'
imbrc→	import { Route, Switch, NavLink, Link } from 'react-router-dom'
imbr→	import { Route } from 'react-router-dom'
imbrs→	import { Switch } from 'react-router-dom'
imbrl→	import { Link } from 'react-router-dom'
imbrnl→	import { NavLink } from 'react-router-dom'
imrs→	import React, { useState } from 'react'
imrse→	import React, { useState, useEffect } from 'react'
redux→	import { connect } from 'react-redux'
rconst→	constructor(props) with this.state

2. ESLint

ESLint는 코드를 자동으로 형식화하고 오류가 날 경우 경고 메시지로 개발자를 지원한다. 만약 여러 개발자가 협업을 하는 경우라면 일관된 서식으로 마치 한사람이 코딩한 것과 같은 결과를 얻을 수도 있다.

ESLint를 설치하기 위해서는 Node.js가 필요하니 먼저 Node.js를 설치하고(NPM설치) 다음과 같이 단계별로 설치해보자.

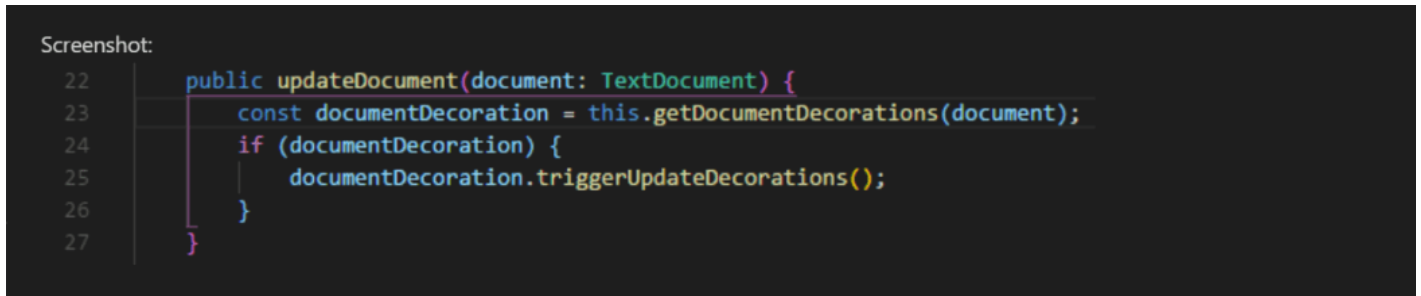
1. 먼저 VS Code에서 ESLint를 인스톨
2. 터미널을 열고 npm install eslint 실행(npm이 없는 경우 먼저 npm설치 후 실행)
3. 이어서 eslint -init 실행(자신이 자주 사용하는 것 위주로 설정값을 준다. 나중에 변경가능)
4. 폴더에 .eslintrc.js가 생성되었는지 확인
5. .eslintrc.js의 rules에 원하는 규칙을 입력 (eslint의 rules는 <https://eslint.org/docs/rules> 를 참조)

[좀 더 많은 설정값이 궁금하다면 다음을 클릭](https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint) [https://marketplace.visualstudio.com/items?](https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint)

itemName=dbaeumer.vscode-eslint

3. Bracket Pair Colorizer

코드에서 수천 수만개의 대괄호들은 개발자들의 골칫거리가 아닐 수 없다. 어떤 괄호가 서로 짝인지 알아보는 것은 거의 불가능에 가깝다. 이제 Bracket Pair Colorizer를 설치해보자. Bracket Pair Colorizer를 설치하는 순간, 우리의 대괄호들이 색깔을 갖게 된다. 서로 일치하는 색상을 사용하므로 코드를 훨씬 더 보기 쉽게 해준다.

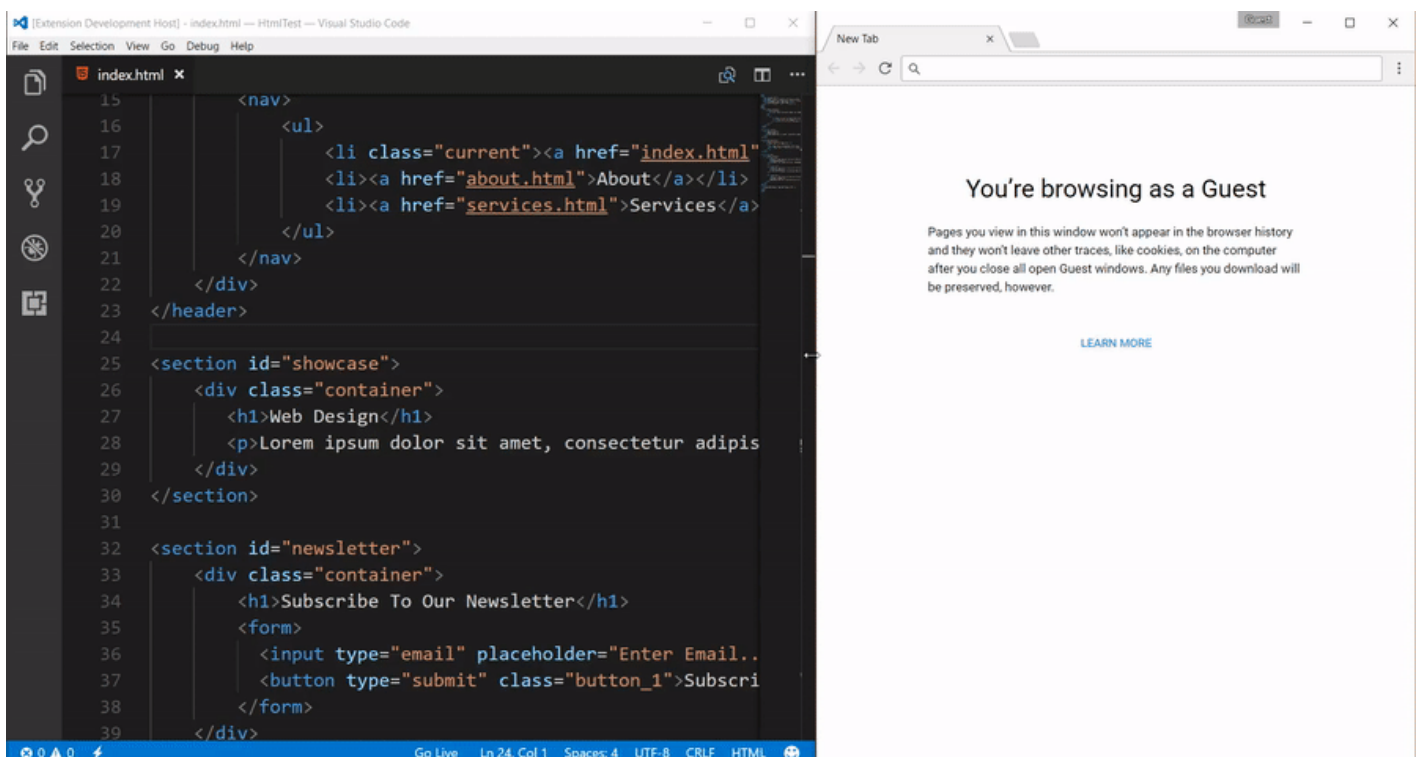


[좀 더 다양한 설정이 궁금하시다면 아래 링크 클릭](https://marketplace.visualstudio.com/items?itemName=CoenraadS.bracket-pair-colorizer) [https://marketplace.visualstudio.com/items?](https://marketplace.visualstudio.com/items?itemName=CoenraadS.bracket-pair-colorizer)

itemName=CoenraadS.bracket-pair-colorizer

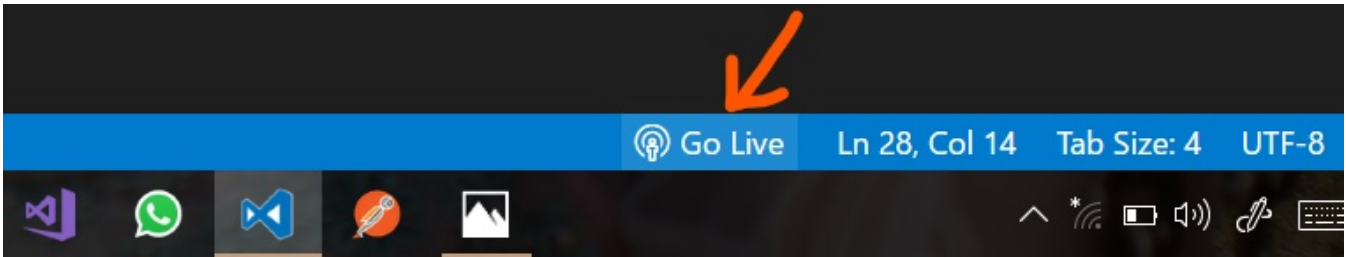
4. Live Server(라이브 서버)

코드를 변경하고 브라우저로 전환한 다음 새로 고침하여 변경 사항을 확인하는 것은 개발자의 끝없이 반복 작업이다. Live Server는 로컬 호스트 서버를 시작시키고 그곳에서 앱을 실행시키며 코드가 변경될 때마다 자동으로 새로 고침이 되는 기능을 제공한다.

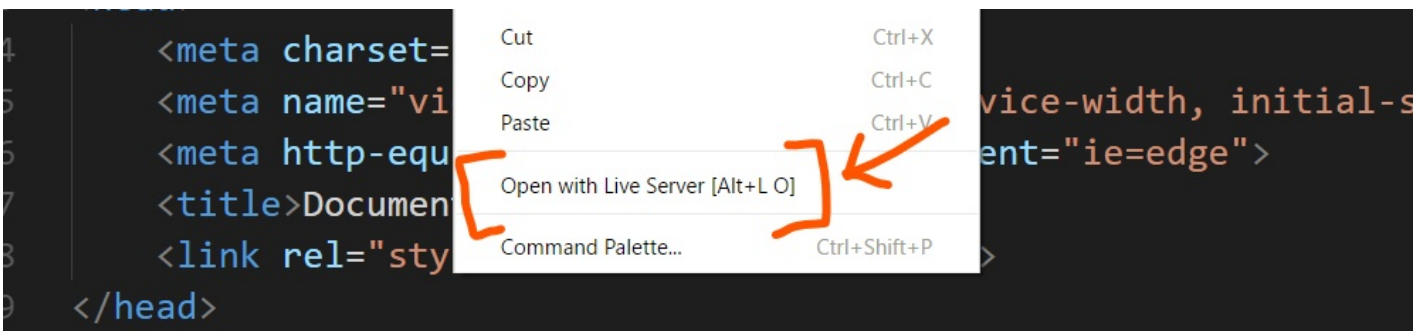


설정방법

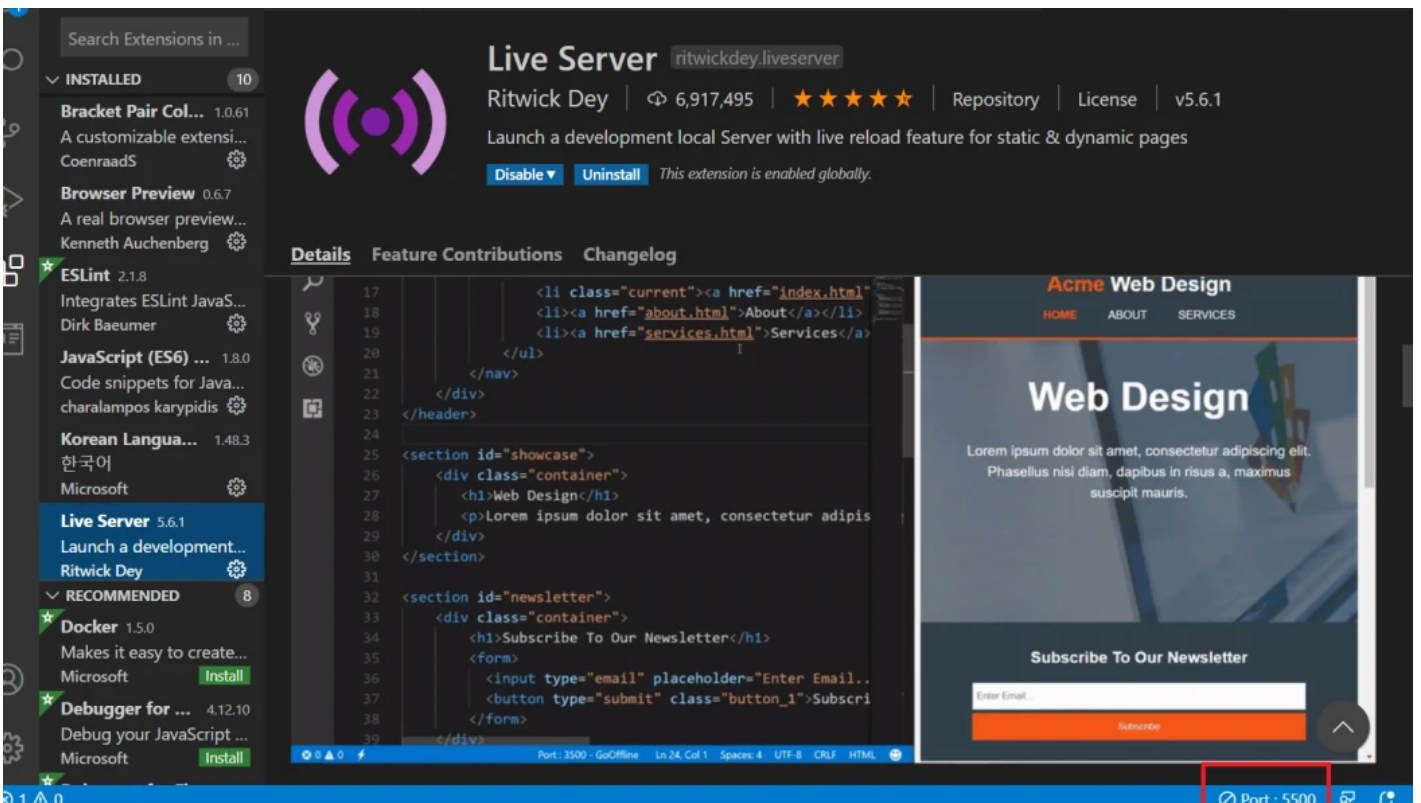
1. Live Server를 인스톨
2. 오른쪽 하단(Windows)에 "Go Live" 클릭



3. 파일을 열때 우클릭한 후 Open with Live Server. 를 선택하여 연다..



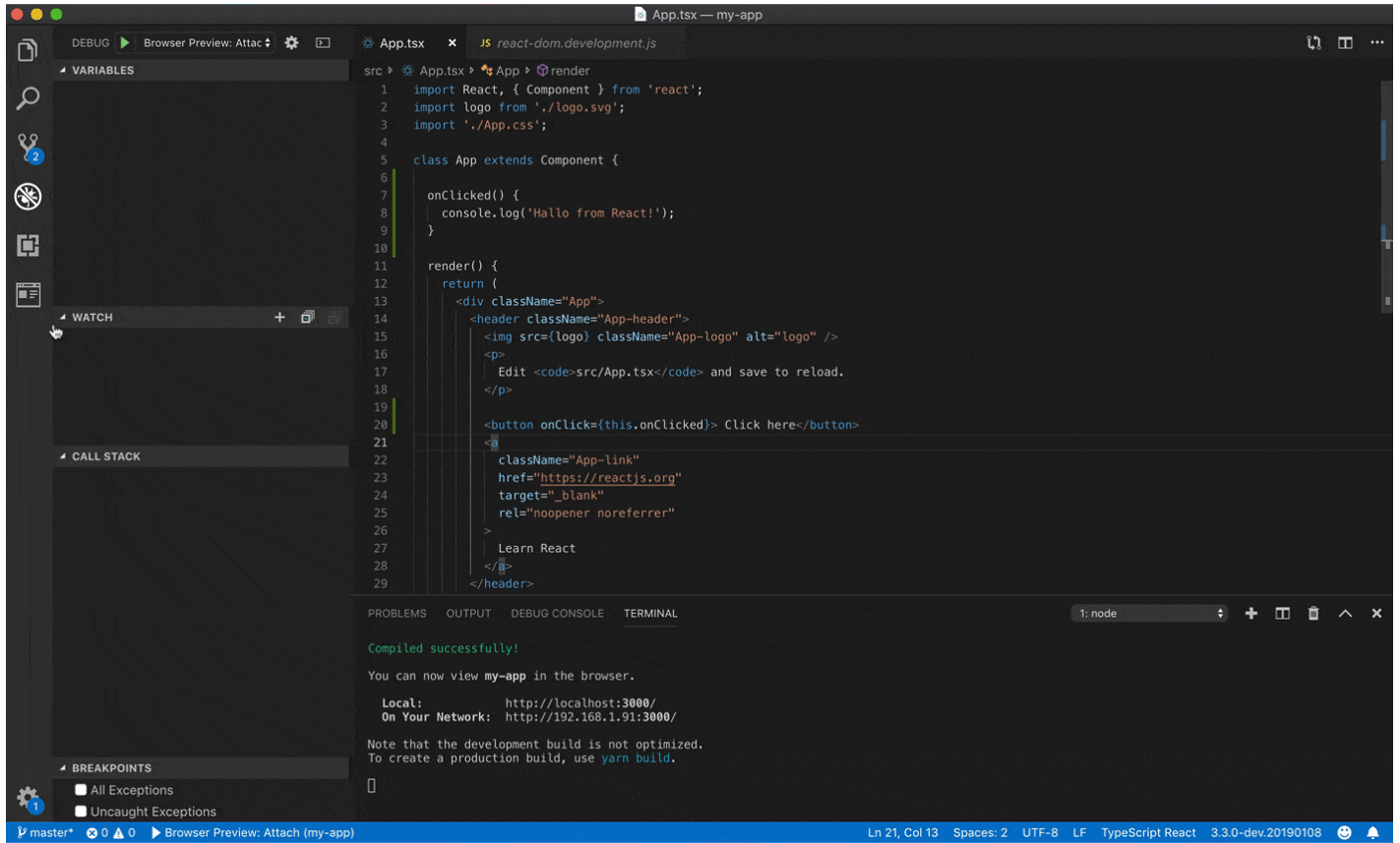
4. 서버를 중지할때는 아래 표시부분을 클릭



더 많은 기능을 확인하고 싶을땐 아래 링크 클릭 <https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer>

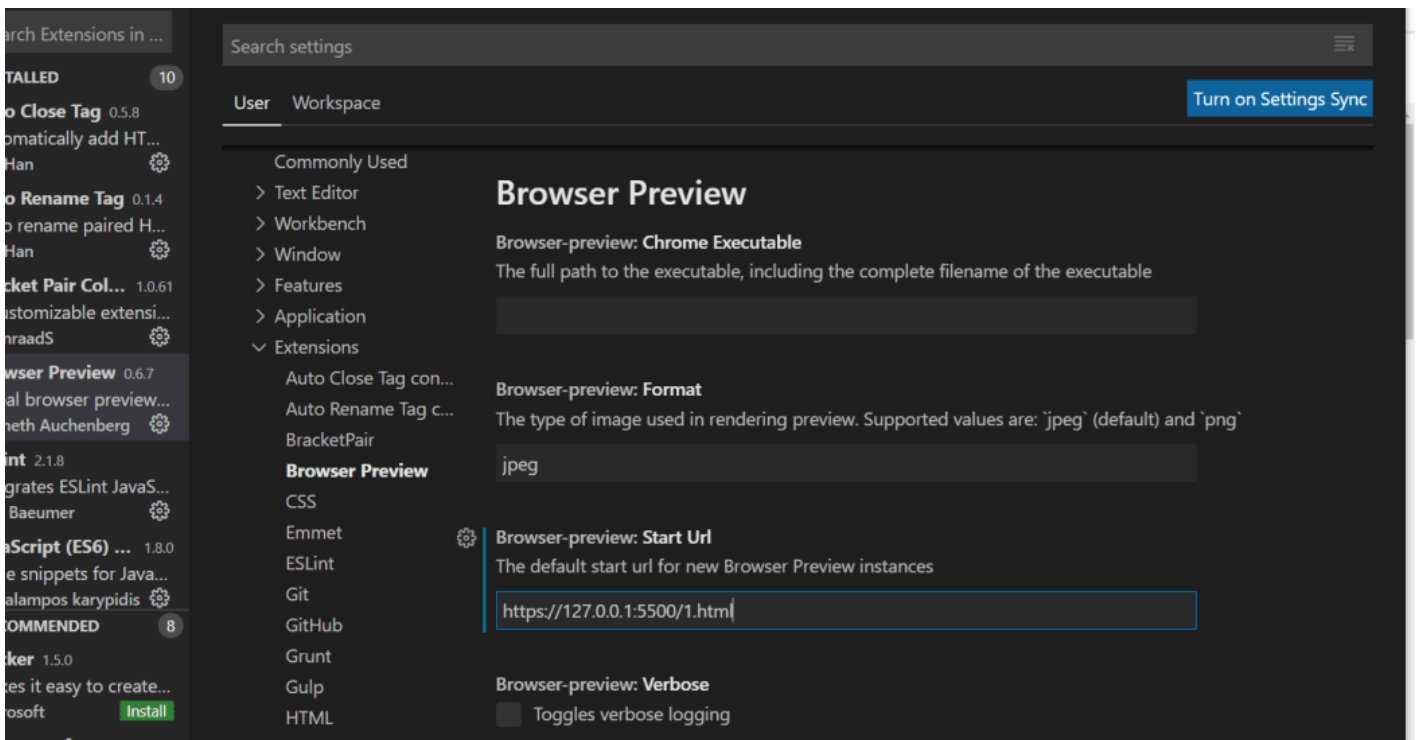
5. Browser Preview (VS Code내에서 브라우저 미리보기)

Live Server 확장의 기능에 한단계 더 나아가, Browser Preview에서는 비주얼 스튜디오 코드 안에서 바로 리로딩 미리보기를 할 수 있는 실제 브라우저를 제공한다. 더이상 브라우저로 이동할 필요가 없는 것이다.



설정방법

1. Browser Preview를 찾아 인스톨
2. 라이브로 리로딩되는 서버를 시작한다. 만약 서버가 없다면 걱정할 건 없다. Live Server를 설치한 후 “Go Live”하면 된다.
3. 명령 팔레트(ctrl(Mac에서는 cmd)+shift+p)에서 “Browser Preview:Open Preview”를 클릭
4. 잠시 기다리면 오른쪽 화면에 라이브 로컬 서버화면이 뜨는 것을 확인
5. 처음 시작하는 화면설정을 바꾸고 싶다면, File -> Preferences -> Extensions -> Browser Preview에서 “Browser-preview:Start Url”을 찾아 값을 변경한다.



[Browser Preview에 대해 더 많은 설정이 궁금하다면 아래 링크
클릭](https://marketplace.visualstudio.com/items?itemName=auchenberg.vscode-browser-preview) <https://marketplace.visualstudio.com/items?itemName=auchenberg.vscode-browser-preview>

6. Prettier

Prettier는 ESLint와 같이 오류내용을 잡아주지는 않지만 코드를 자동으로 형식화 하는데 있어서 매우 용이한 확장프로그램이다.

Prettier 역시 File->Preferences->Extension->Prettier에서 기본적으로 옵션값을 설정해주면 되는데 이것 역시 Settings.json에서 일괄적으로 설정해주어도 된다.

코드를 자동으로 형식화하는 확장 프로그램은 많기 때문에 Prettier를 기본적인 포맷터로 설정하려면 다음과 같이 전체 설정을 하거나 프로그래밍언어별 설정을 해준다.

```
{
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "[javascript]": {
    "editor.defaultFormatter": "esbenp.prettier-vscode"
  }
}
```

[더 많은 기능을 원하신다면 다음 링크를 클릭](https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode) <https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode>

7. Auto Close Tag 과 Auto Rename Tag (자동 닫기 태그 그리고 자동 이름바꾸기 태그)

이 Auto Close Tag는 PHP, Vue, Javascript, TypeScript, JSX, TSX 등등 그 전에는 자동 태그가 되지 않은 언어들을 지원한다. 또한 이 기능을 사용할지 말지, 어떤 언어에 이 기능을 사용할지 등에 대한 세부적인 설정이 가능하다.

세부적인 설정이 궁금하다면 아래 링크 클릭 <https://marketplace.visualstudio.com/items?itemName=formulahendry.auto-close-tag>

Auto Rename도 역시 VS Code가 기본적으로 제공하는 부분 이외의 언어들도 지원한다. 기능을 활성화시키는 옵션도 개별적인 설정에서 가능하다. 아래 프로그래밍 언어 이름 부분에는 기본값이 ["*"]으로 설정되어있는데 이것은 모든 언어에 적용시킨다는 의미이다.

```
{
  "auto-rename-tag.activationOnLanguage": ["html", "xml", "php", "javascript"]
}
```

8. Code Spell Checker (코드 내 철자 검사)

코드를 작성할 때 오타가 나거나 가끔은 철자가 정확히 떠오르지 않은 경우도 있을 것이다. Code Spell Checker는 철자가 틀렸을 경우 밑줄로 알려주고 가능한 단어들의 옵션을 제시한다. 기본적으로 영어를 지원해주고 그밖에 다른 나라의 언어는 추가적으로 설치한다면 가능하다.

가능한 언어를 확인하고 싶으면 다음을 클릭 <https://marketplace.visualstudio.com/items?itemName=streetsidesoftware.code-spell-checker>

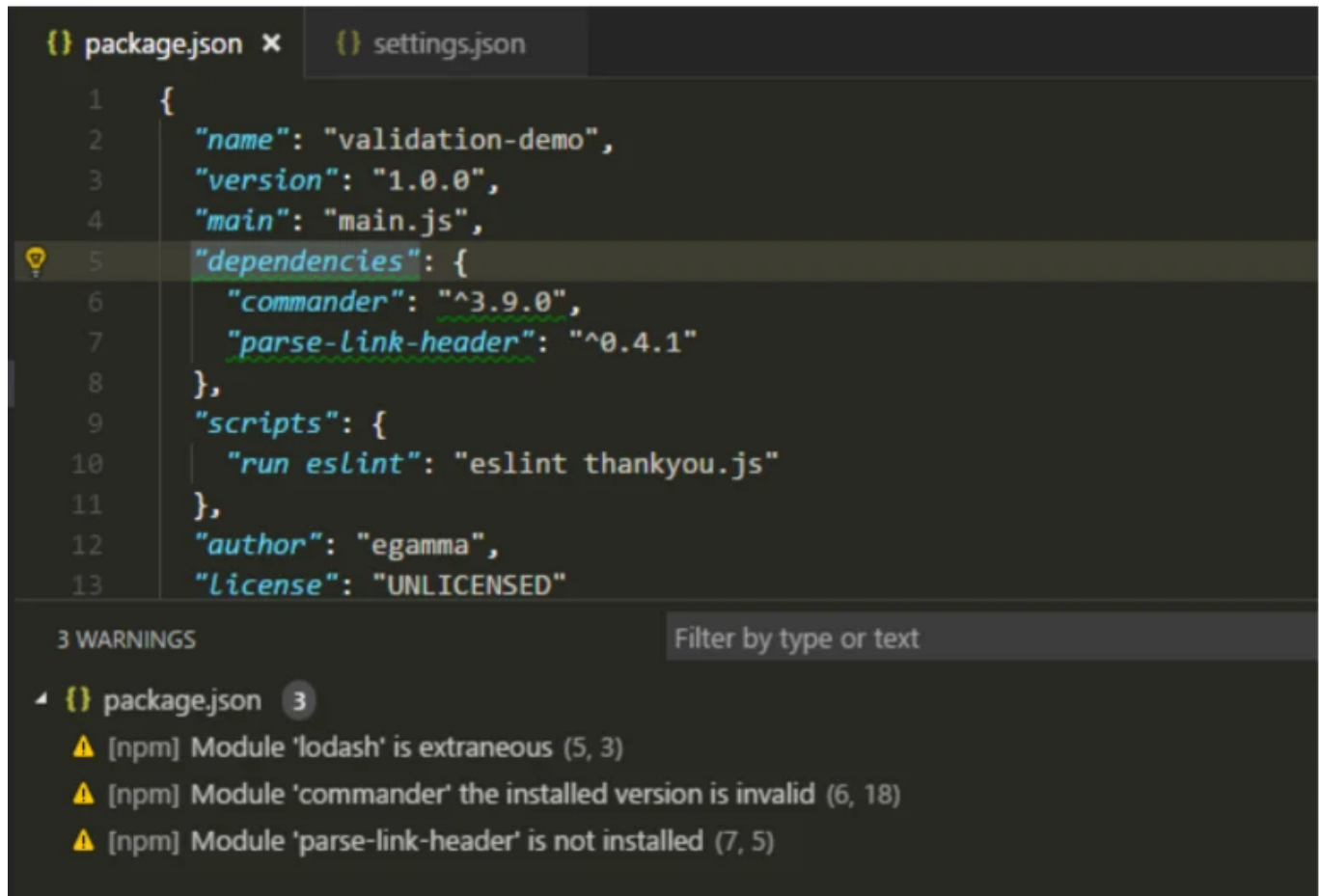
9. NPM(Node Package Manager)

node.js로 만들어진 package를 관리해주는 도구이다. 개발자는 npm을 통해 node.js로 만들어진 module을 웹에서 받아서 설치하고 package.json을 통해 관리할 수 있다.

node.js로 만들어진 외부 모듈을 쓰는 경우가 많아지면 모듈마다 버전관리도 힘들어지고 새 프로젝트에서 동일한 모듈을 인스톨해야할 경우 반복작업을 해야 하는 불편함을 해소하기 위해 package.json이라는 파일을 만들어 모듈별 관리를 한다. package.json은 생성시 다음과 같은 경우 경고메세지를 주기도 하며,

- package.json에는 있으나 인스톨되지 않은 경우
- 인스톨 되었으나 package.json에 정의되어있지 않은 경우
- 인스톨 되었으나 package.json에 정의되어있는 버전과 맞지 않을 경우

npm을 실행시 빠른 수정을 위해 아래와 같은(스크린하단) 경고메세지도 보여준다.



The screenshot shows a code editor with two tabs: 'package.json' and 'settings.json'. The 'package.json' tab is active, displaying a JSON configuration for a project named 'validation-demo'. The configuration includes fields for 'name', 'version', 'main', 'dependencies', 'scripts', 'author', and 'license'. The 'dependencies' section lists 'commander' and 'parse-link-header'. A yellow lightbulb icon indicates a warning on line 5. Below the code editor, a 'WARNINGS' section is visible, listing three warnings related to the 'package.json' file. The warnings are: 1. 'Module 'lodash' is extraneous (5, 3)', 2. 'Module 'commander' the installed version is invalid (6, 18)', and 3. 'Module 'parse-link-header' is not installed (7, 5)'. Each warning is preceded by a yellow triangle icon.

```
{
  "name": "validation-demo",
  "version": "1.0.0",
  "main": "main.js",
  "dependencies": {
    "commander": "^3.9.0",
    "parse-link-header": "^0.4.1"
  },
  "scripts": {
    "run eslint": "eslint thankyou.js"
  },
  "author": "egamma",
  "license": "UNLICENSED"
}
```

3 WARNINGS Filter by type or text

package.json 3

- ⚠ [npm] Module 'lodash' is extraneous (5, 3)
- ⚠ [npm] Module 'commander' the installed version is invalid (6, 18)
- ⚠ [npm] Module 'parse-link-header' is not installed (7, 5)