- V8
  - Chrome V8 Javascript 엔진으로 빌드된 Javascript 런타임
- Event Loop
  - 콜스택 -> 콜백큐
- Non-Blocking I/O
  - = 비동기식 I/O
- 싱글 스레드
  - 하나의 마스터 프로세스 -> CPU 개수만큼 워커 프로세스를 생성

개발환경 구성

- Visual Studio Code 설치
- Node.js 설치
  - 18.12.1 LTS 다운로드
  - 설치 후 확인

```
$ node -v                    ==> v18.12.1
$ npm -v                     ==> 8.18.0
```

- VS Code 터미널 모드 변경

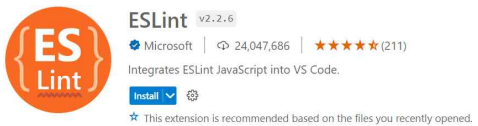
F1

View ->
Commend Palette

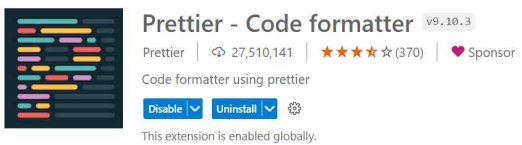
개발환경 구성

- VS code Extension 설치
  - JavaScript (ES6) snippets
    - 자바스크립트 코드 자동완성

  - ESLint
    - 자바스크립트 문법 오류 체크

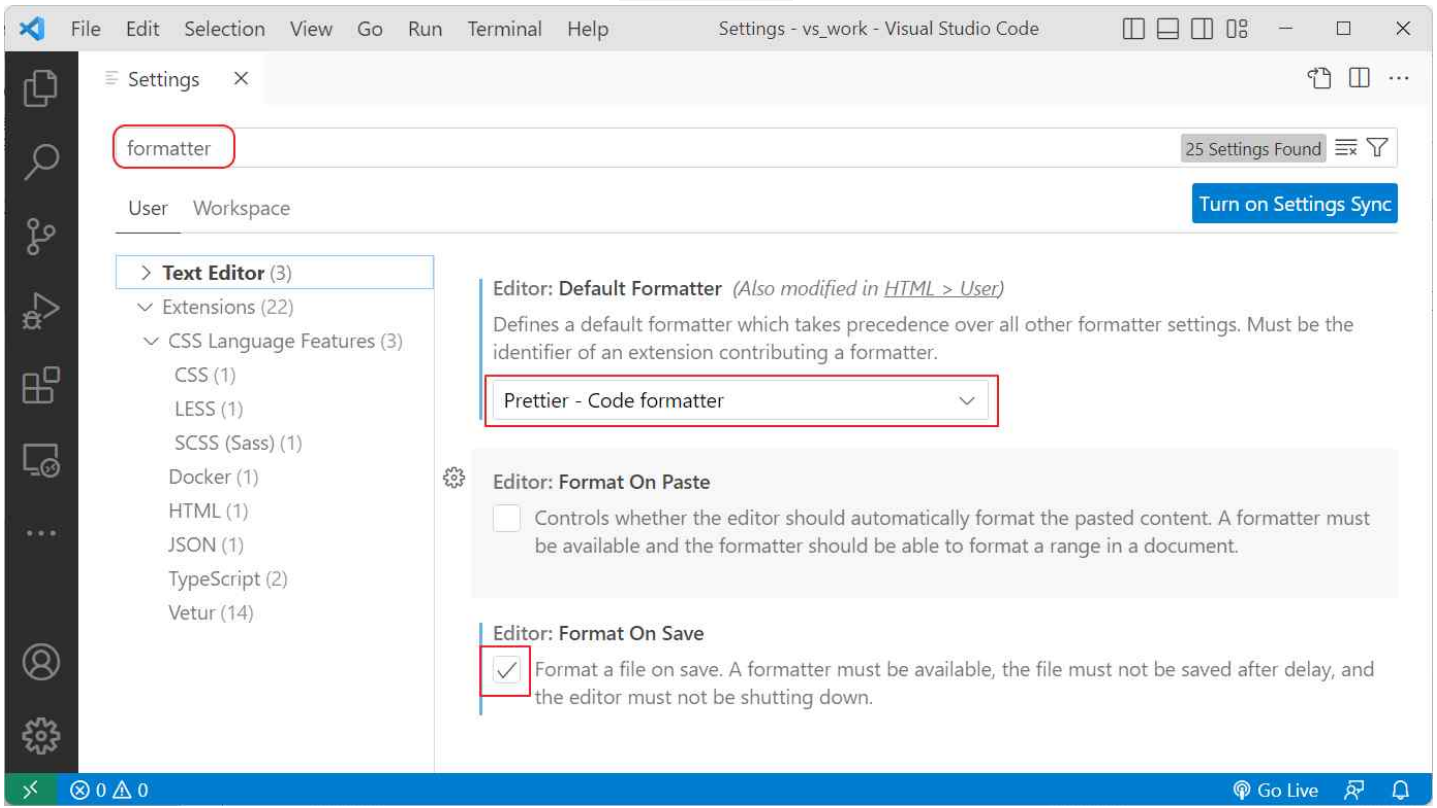


  - Pretter - Code Formatter
    - 미리 지정된 코드포맷 스타일로 자동변경

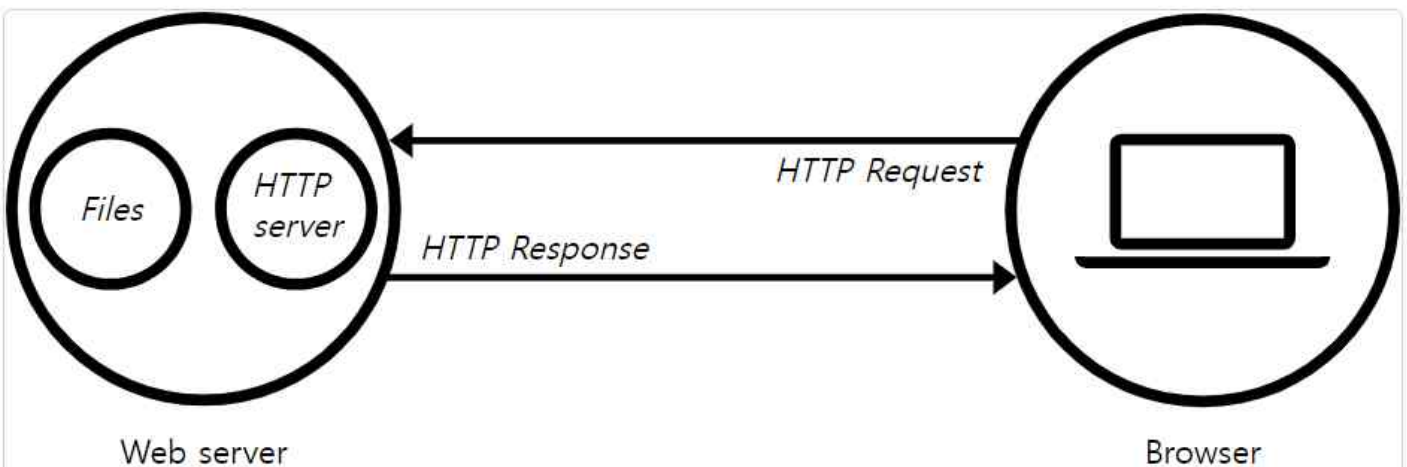


  - Live Server

- VS code setting
  - File -> Preferences -> Settings `Ctrl + ,`



- 웹서버란


Files
HTTP server
HTTP Request
HTTP Response
Web server
Browser

웹서버 만들기

- node.js 는 웹서버 기능을 내장

```
const http = require('http');
const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

```
D:\node_work> node app.js
Server running at http://127.0.0.1:3000/
```

```
// 1. http 모듈 포함
const http = require('http');

// 2. http 서버 생성
const server = http.createServer((req, res) => {
  /* 5. 새로운 요청이 수신되면 request 이벤트가 호출되고
   http.IncomingMessage 객체와 http.ServerResPonse 객체를 넘겨준다
   req : 요청 상세정보( 요청 헤더와 요청 data )
   res : 클라이언트(호출자)에게 데이터를 반환 ( 상태코드, contentType, 응답 데이터 ) */

});

// 3. 지정된 포트 및 호스트이름으로 수신 대기
server.listen(3000, '127.0.0.1', () => {
  // 4. 서버가 준비되면 콜백함수 호출
});
```

ECMA 스크립트

- ECMA
  - European Computer Manufactures Association International : 유럽 컴퓨터 시스템 표준화 기구
- ECMA-262
  - 자바스크립트의 표준
  - 브라우저간의 호환성 문제 해소. 각 브라우저 개발사들이 ECMAScript 표준을 따라 브라우저를 구현
- 버전
  - ES5 = 2009년
  - ES6 = ES2015
    - let,const/Arrow function/for~of/default parameter/spread operator(…)/template literal/ Destructuing Assignment/promise/Map/Set/Module/Symbol/class
  - ES8 = ES2017
    - async,await/String padding
- 바벨(Bebel)
  - 구 브라우저에서도 최신 자바스크립트 코드를 작동하도록 변환해주는 트랜스파일러
- 폴리필(Polyfill)
  - 기능을 지원하지 않는 웹 브라우저에 최신 표준의 자바스크립트 기능을 구현해주는 호환성 구현 라이브러리 코드

서버프로그램을 위한 자바스크립트

- 변수선언자
- Arrow Function
- Array 내장 함수
- Template Literals
- Spread 연산자
- Object Destructing
- Array Destructing
- Default Function Paramter
- Rest Parameter
- Promise
- Async/Await
- class
- Regular Expression

https://nodejs.dev/en/learn/how-much-javascript-do-you-need-to-know-to-use-nodejs
https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Statements

- 객체리터럴
  - 함수연결할 때 콜론과 function 생략 가능
  - 속성명과 변수명이 겹치는 경우 생략

```
let sayNode = function () {
  console.log("node");
};
```

```
let oldObject = {
  sayJS: function () {
    console.log("js old");
  },
  sayNode: sayNode,
};
```

```
let newObject = {
  sayJS() {
    console.log("js new");
  },
  sayNode,
};
```

- 모듈
  - 특정기능을 하는 하나의 코드 묶음 단위
- 캡슐화
  - 모듈 안의 모든 기능들은 모듈 안에서만 동작하며, 모듈 밖에서는 접근이 허용된 속성이나 메서드만 사용가능
- 모듈시스템 규칙
  - 모듈은 파일 단위로 구성
  - 모듈의 변수, 함수 클래스 등은 export 키워드로 노출하고 import로 가져다 사용한다.
  - 모듈 이름은 중복 안됨
  - 모듈은 순환 참조를 할 수 없음
  - 모듈도 하나의 객체로서 임포트 시점에 모듈객체의 참조 주소를 변수에 할당
  - export 키워드로 내보낼 수 있는 모듈은 var. let, const, function, class 임

- app.html

```
 <script type="module" src="./main.js"></script>
```

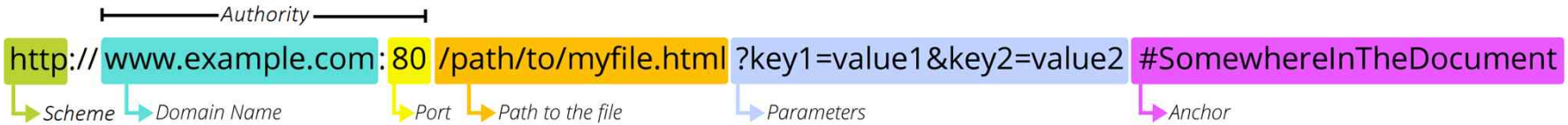
SyntaxError: Cannot use import statement outside a module

- main.js

```
import { module } from "./module.js";
module("module run");
```

- module.js

```
export function module(msg) {
  console.log("msg:" + msg);
}
```



- scheme(protocol)
  - 통신규칙 : ftp, https
- domain name(host)
  - 인터넷에 연결되어 있는 컴퓨터의 주소
- port
  - 한 대의 컴퓨터에 여러 개의 서버가 있는 경우 서버를 구분
- path
  - 경로
- parameter(query string)
  - 서버로 전달되는 데이터. ?name=value&name=value로 구성

참고사이트 : https://developer.mozilla.org/ko/docs/Learn/Common_questions/What_is_a_URL

```
const http = require("http");
const server = http.createServer(function (req, res) {
  const myURL = new URL("http://127.0.0.1:3000" + req.url);
  let pathname = myURL.pathname;
  if (pathname == "/") {
    res.statusCode = 200;
    res.setHeader("Content-Type", "text/plain");
    res.end("hello");
  } else if (pathname == "/info") {
    res.statusCode = 200;
    res.setHeader("Content-Type", "text/html");
    let template = `<!DOCTYPE html><html lang='ko'> <head><meta charset="UTF-8"></head>
    <body><h1 style='color:blue'>node 서버</h1></body></html>`;
    res.end(template);
  } else {
    res.statusCode = 404;
    res.end("error");
  }
}).listen(3000, function () {  console.log("server runtime http://localhost:3000");});
```

- URLSearchParams

```
const http = require("http");
const server = http.createServer(function (request, response) {
  const url = request.url;
  const myURL = new URL("http://127.0.0.1" + url);
  console.log("pathname", myURL.pathname);
  console.log("search", myURL.searchParams);
  console.log("id", myURL.searchParams.get("id"));
  response.end("hello");
});
```

- Node Package Manager
  - 대부분의 자바스크립트 프로그램은 패키지라는 이름으로 npm 서버에 등록되어 있으므로 찾아서 설치
  - https://www.npmjs.com/
- package.json
  - 설치한 패키지의 버전을 관리하는 파일
  - 패키지간의 의존관계 관리

```
npm init
npm run test
```

- yarn
  - npm 대체자로 페이스북이 내놓은 패키지 매니저
  - 따로 설치해야 하며 npm이 느릴 경우 yarn 패키지로 대신 설치 가능

- Process Manager
  - node.js 어플리케이션 프로세스 메니저 https://pm2.keymetrics.io/
  - 무중단 서비스 지원 -가동 중지시간없이 응용프로그램을 영원히 활성상태로 유지(자동으로 리로드)
  - 현재 디렉터리 또는 하위 디렉터리에서 파일이 수정될 때 애플리케이션을 자동으로 다시 시작.
- npm 설치

```
npm install pm2 -g
```

- application 시작

```
pm2 start app.js [--watch]
```

- application 관리

```
pm2 list
pm2 stop    app_name
pm2 restart app_name
pm2 delete  app_name
```

- log 보기

```
pm2 logs [app-name]
pm2 logs --time
pm2 logs --json
```

- 실행중인 모든 프로세스 모니터링

```
 pm2 monit
```