

- Node.js란 자바스크립트 런타임 실행환경으로 웹서버를 구축하기 위해
- 개발환경 구성
- 서버프로그램을 위한 자바스크립트
- Node.js 시작하기
- Node.js 내장 모듈과 객체

npm: 노드에서 사용되는 패키지(라이브러리) 의존성 관리도구

모듈화: 기능들을 모아 함수화하여 자바스크립트에 쓸 수 있게 하는것?

내장된 모듈:http 등 외장모듈은 npm install로 다운로드

서버생성 createserver에 들어가는 핸들러: requesthandler는 function(req,res){}이고 (req,res)=>{}라고 표현가능

http.createServer((req,res)=>{})

서버가동은 server.listen(포트번호,콜백함수"콜백함수 생략가능")

- 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

- 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



- VS code Extension 설치

- JavaScript (ES6) snippets
 - 자바스크립트 코드 자동완성



JavaScript (ES6) code snippets v1.8.8
charalampos karypidis | 10,565,934 | ★★★★★ (36)
Code snippets for JavaScript in ES6 syntax
[Install](#) ⓘ

- ESLint
 - 자바스크립트 문법 오류 체크



ESLint v2.2.8
Microsoft | 24,047,686 | ★★★★★ (211)
Integrates ESLint JavaScript into VS Code.
[Install](#) ⓘ
★ This extension is recommended based on the files you recently opened.

- Prettier - Code Formatter
 - 미리 지정된 코드포맷 스타일로 자동변경

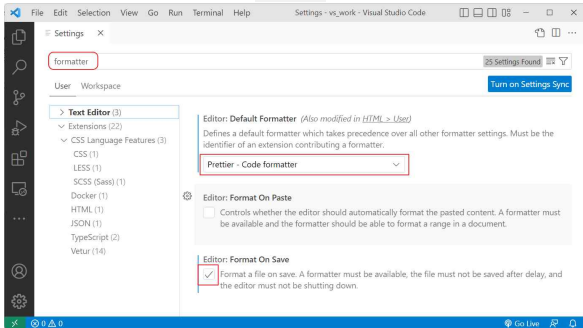


Prettier - Code formatter v2.1.0
Prettier | 27,510,141 | ★★★★★ (370) | ❤️ Sponsor
Code formatter using prettier
[Install](#) ⓘ [Uninstall](#) ⓘ ⓘ
This extension is enabled globally.

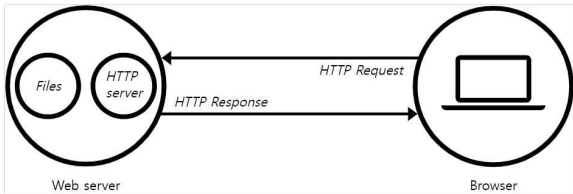
- Live Server

- VS code setting

- File -> Preferences -> Settings **Ctrl + ,**



- 웹서버란



- 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
});
```



- **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

```
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");
});
```

```
const server = http.createServer(function (req, res) {  
  res.statusCode = 200;  
  res.setHeader("Content-Type", "text/html");  
  res.writeHead(200, { 'Content-Type': "text/html" });  
  res.write(`<html lang='ko'><head></head><body>node 서버</body></html>`);  
  res.end(template);  
})
```

- **StatusCode(응답코드)**
 - 200(OK), 301/302(redirect), 403(forbidden), 404(not found), 500(internal server error)
- **setHeader**
 - MIME 타입 : 웹브라우저에게 문서를 어떻게 출력할지 결정하는 부분
- **write()**
 - 클라이언트에 실제로 보내고자 하는 데이터를 전송. 여러 번 사용 가능
- **end()**
 - 해당 데이터를 보낸 뒤 서버와 클라이언트와의 접속이 종료

- 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
- package.json 파일 생성

```
npm init
```

```
{
  "scripts": {
    "start": "node server.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "type": "module",
  "dependencies": {
    "pm2": "^5.2.2"
  }
}
```

```
>npm start 인수
```

- 패키지 설치

```
npm install mysql  
npm install mysql@latest           //최신버전  
npm install mysql@2.11.0
```

- 패키지 제거

```
npm uninstall mysql
```

- package.json에 있는 모든 의존성이 있는 모듈을 자동으로 설치

```
npm install  
npm update
```

- node_modules 폴더

- 다운로드 받아서 설치된 모듈파일 위치

- yarn

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

- node_modules의 위치 확인

```
npm root
```

```
D:\node_work\node_week1>npm root
D:\vs_work\node_week1\node_modules
```

- 패키지 조회

```
npm ls
```

```
D:\vs_work\node_week1>npm ls
node_week1@ D:\vs_work\node_week1
├── global@4.4.0
└── pm2@5.2.2
```


- export/require
- export/import

- Process Manager

- node.js 어플리케이션 프로세스 매니저 <https://pm2.keymetrics.io/>
- 무중단 서비스 지원 -가동 중지시간없이 응용프로그램을 영원히 활성상태로 유지(자동으로 리로드)
- 현재 디렉터리 또는 하위 디렉터리에서 파일이 수정될 때 어플리케이션을 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
```