



# Arduino-IoT

[wk02]

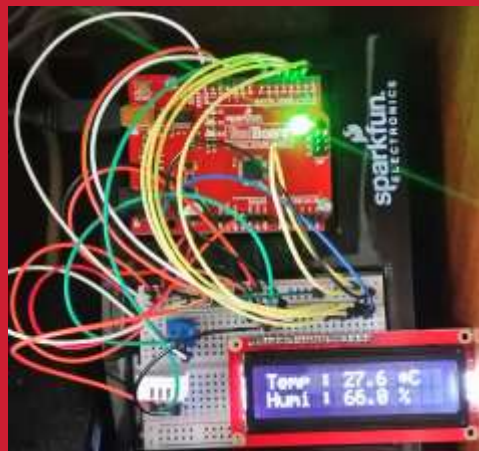
## Start node.js

Visualization of Signals using Arduino, Node.js & storing signals in MongoDB & mining data using Python

Drone-IoT-Comsi, INJE University

2<sup>nd</sup> semester, 2021

Email : chaos21c@gmail.com





# My ID

## ID를 확인하고 github에 repo 만들기

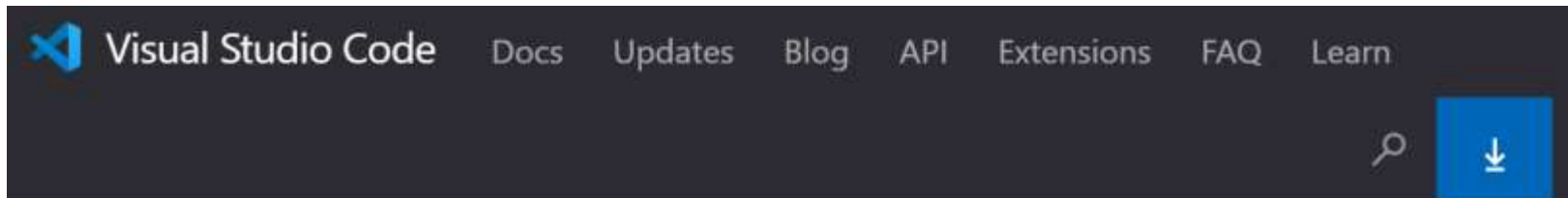
AA01	김준수	AA13	조재윤
AA02	김현서	AA14	고태승
AA03	박영훈	AA15	이한글
AA04	박윤호	AA16	장세진
AA05	성은지	AA17	장태호
AA06	손윤우	AA18	정지원
AA07	오세윤	AA19	진우태
AA08	우승철	AA20	황혁준
AA09	윤현석	AA21	장이제
AA10	이예주	AA22	박상현
AA11	강지환	AA23	정은성
AA12	성인제	AA24	김경영

위의 id를 이용해서 github에 repo를 만드시오.

Option: 아두이노응용 실습 과제 - AAnn

Public, README.md check

# New editor: Vscode portable by MS



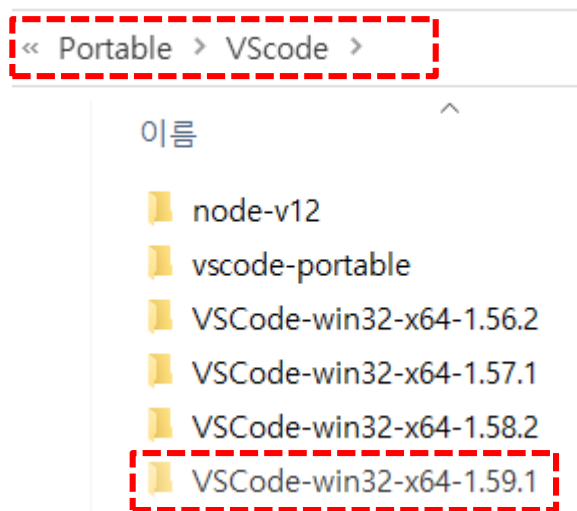
User Installer	64 bit	32 bit	ARM
System Installer	64 bit	32 bit	ARM
.zip	64 bit	32 bit	ARM



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

<https://code.visualstudio.com/download>

# New editor: Vscode portable (MS)



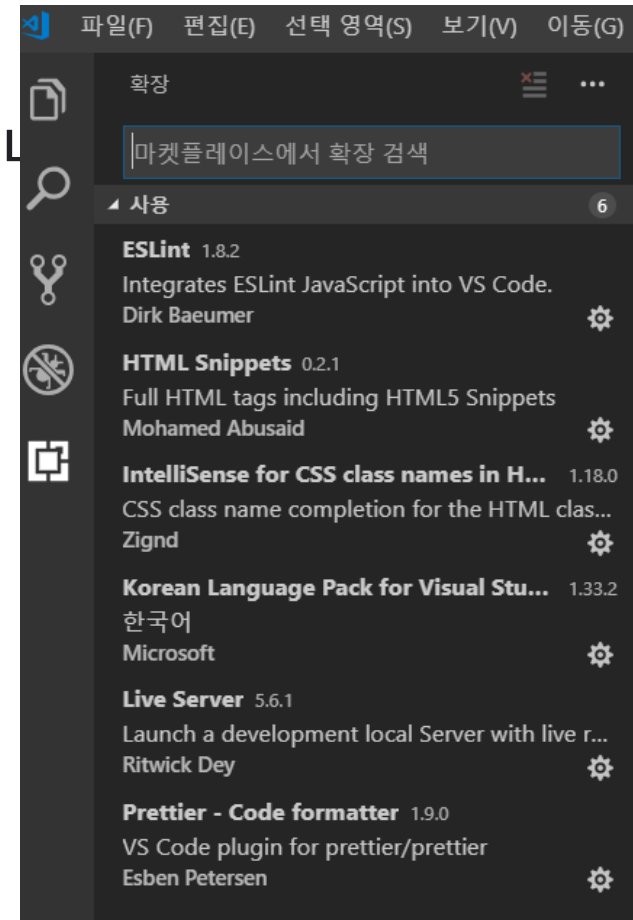
하드디스크 D에 **portable** 폴더를 만들고  
**vscode** 폴더에  
압축을 풀어서 사용

# New editor: VScode

**확장프로그램 설치** (각각 설치 후 vscode 재시작)

1. Korean language pack
2. HTML snippet
3. IntelliSense for CSS class names in HTML
4. Javascript (ES6)
5. Prettier
6. Live server (for HTML preview)
7. GitLens, Git History
8. Material Icon Theme
9. Python

C, C++, Java,  
Node.js  
Python, Jupyter  
... all coding!



# New editor: VScode

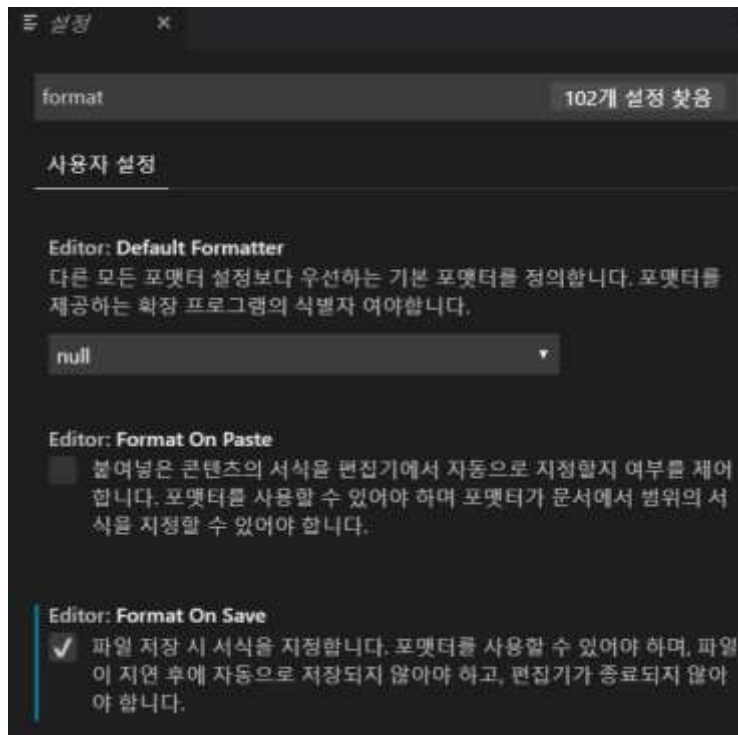
**확장프로그램 설정** (각각 설정 후 종료/재시작)

1. Prettier

2. Live server (for HTML preview)

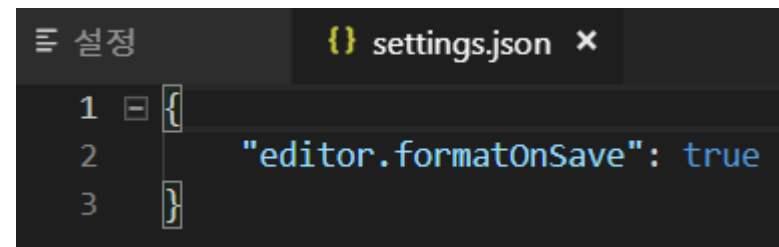
3. GitLens, Git History

**파일 > 기본설정 > 설정 → 사용자 설정**



검색: format

Editor: Format on Save (check)



**설정 → Ctrl + Shift + P**

# 실습 준비: aann/aann-rpt01 폴더에서 js code

7

## [파일] 메뉴

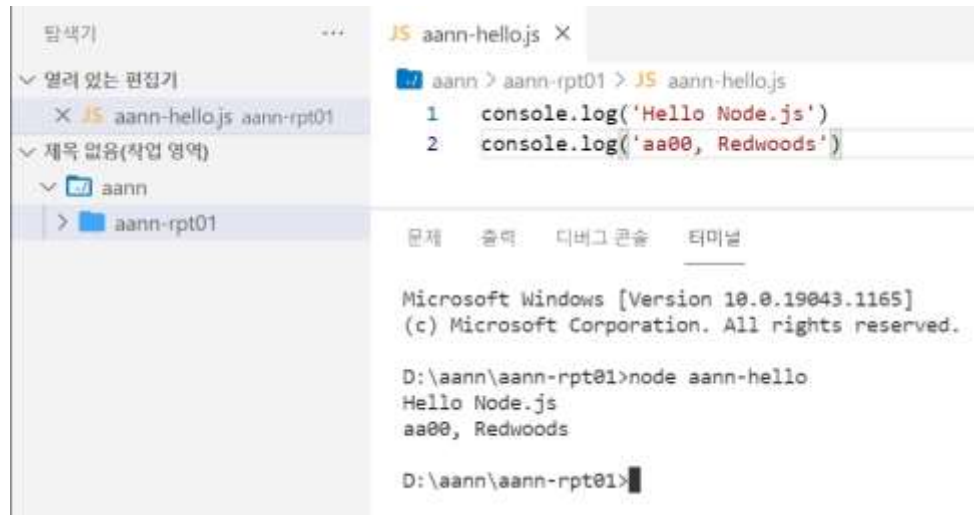
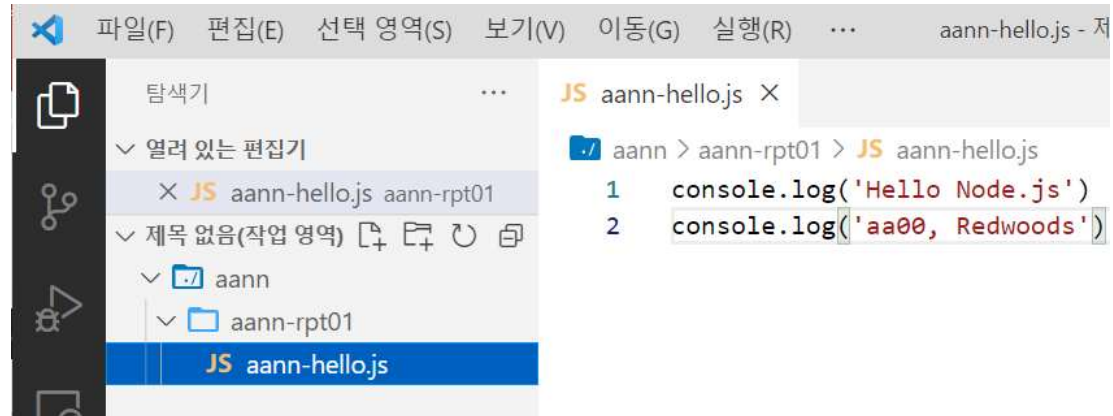
→ Node js 소스 만들기

aann-rpt01 폴더 내에  
aann-hello.js 파일 소스 생성.

→ Node js 소스 실행

aann-rpt01 폴더를 우클릭  
통합터미널 열기 선택  
다음 명령 실행

`node aann-hello.js`



# [Project]

## ◆ [wk01]

- upload all work of this week
- Make repo “aann” in github
- upload folder “aann-rpt01” in your github.



# 실습 : 결과를 나의 github에 올리기

9

따라서 함께 해봅시다.

◆ Github.com 에 각자의 public 계정을 만드시오.  
(이미 github 계정이 있으면 불필요)

1. 실습 결과를 올리는 github repo를 "aann"로 만드시오. (반드시 README.md 를 추가)
2. README.md에는 "아두이노응용 실습 과제" 입력
3. "aann" repo 에 aann-rpt01 폴더 upload
4. 각자의 github 주소를 이메일로 보내시오.  
<https://github.com/accountName/aann>

Email : [chaos21c@gmail.com](mailto:chaos21c@gmail.com)

# Purpose of AA

주요 수업 목표는 다음과 같다.

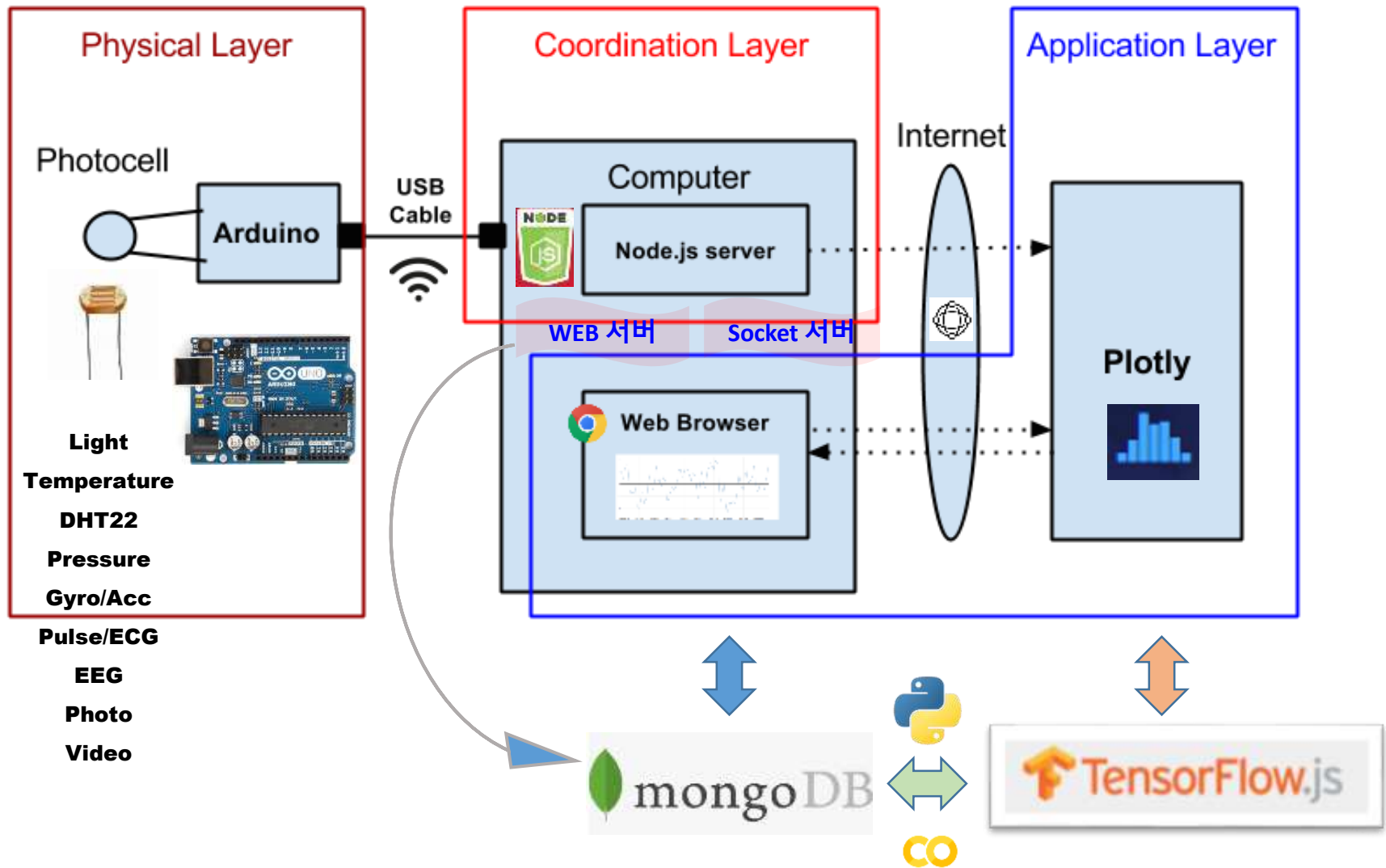
1. Node.js를 이용한 아두이노 센서 신호 처리
2. Plotly.js를 이용한 아두이노 센서 신호 시각화
3. MongoDB에 아두이노 센서 데이터 저장 및 처리



## 4. 저장된 IoT 데이터의 마이닝 (파이썬 코딩)



# Layout [H S C]





# 1.0 What is node.js?





# 1.0 What is node.js?

← → ↻ 🏠 안전함 | [https://www.w3schools.com/nodejs/nodejs\\_intro.asp](https://www.w3schools.com/nodejs/nodejs_intro.asp)

🏠 HTML CSS JAVASCRIPT SQL PHP BOOTSTRAP HOW TO JQUERY MORE ▼

Node.js Tutorial  
Node.js HOME  
**Node.js Intro**  
Node.js Get Started  
Node.js Modules  
Node.js HTTP Module  
Node.js File System  
Node.js URL Module  
Node.js NPM  
Node.js Events  
Node.js Upload Files  
Node.js Email

Node.js MySQL  
MySQL Get Started  
MySQL Create Database  
MySQL Create Table

## Node.js Introduction

◀ Previous

### What is Node.js?

- Node.js is an open source server framework
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server

### Why Node.js?

**Node.js uses asynchronous programming!**

[https://www.w3schools.com/nodejs/nodejs\\_intro.asp](https://www.w3schools.com/nodejs/nodejs_intro.asp)



# 1.0 What is node.js?

# Javascript on Server

Node.js is an open-source, runtime environment for developing **server-side and network applications in JavaScript**.

Its **event-driven architecture and non-blocking I/O model** makes it ideal for building **real-time applications that run across distributed devices**.



# 1.1 What is node.js?



홈 | ABOUT | 다운로드 | 문서 | 재단 | 참여하기 | 보안 | 뉴스

## Node.js®에 대해서

비동기 이벤트 주도 JavaScript 런타임으로써 Node는 확장성 있는 네트워크 애플리케이션을 만들 수 있도록 설계되었습니다. 다음 "hello world" 예제는 다수의 연결을 동시에 처리할 수 있습니다. 각 연결에서 콜백이 실행되는데 실행할 작업이 없다면 Node는 대기합니다.

```
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\n');
});

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



## 1.2 What is node.js?

- **Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.**
- **Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.**
- **Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.**





# 1.3 Non-blocking IO

## Blocking Code

```
var contents = readFile('/path/some/file.txt');  
console.log(contents);  
console.log('Another independent operation');
```

## Non-Blocking Code

```
var contents = readFile('/path/some/file.txt', function(err, contents){  
    console.log(err || contents);  
});  
  
console.log('Another independent operation');
```

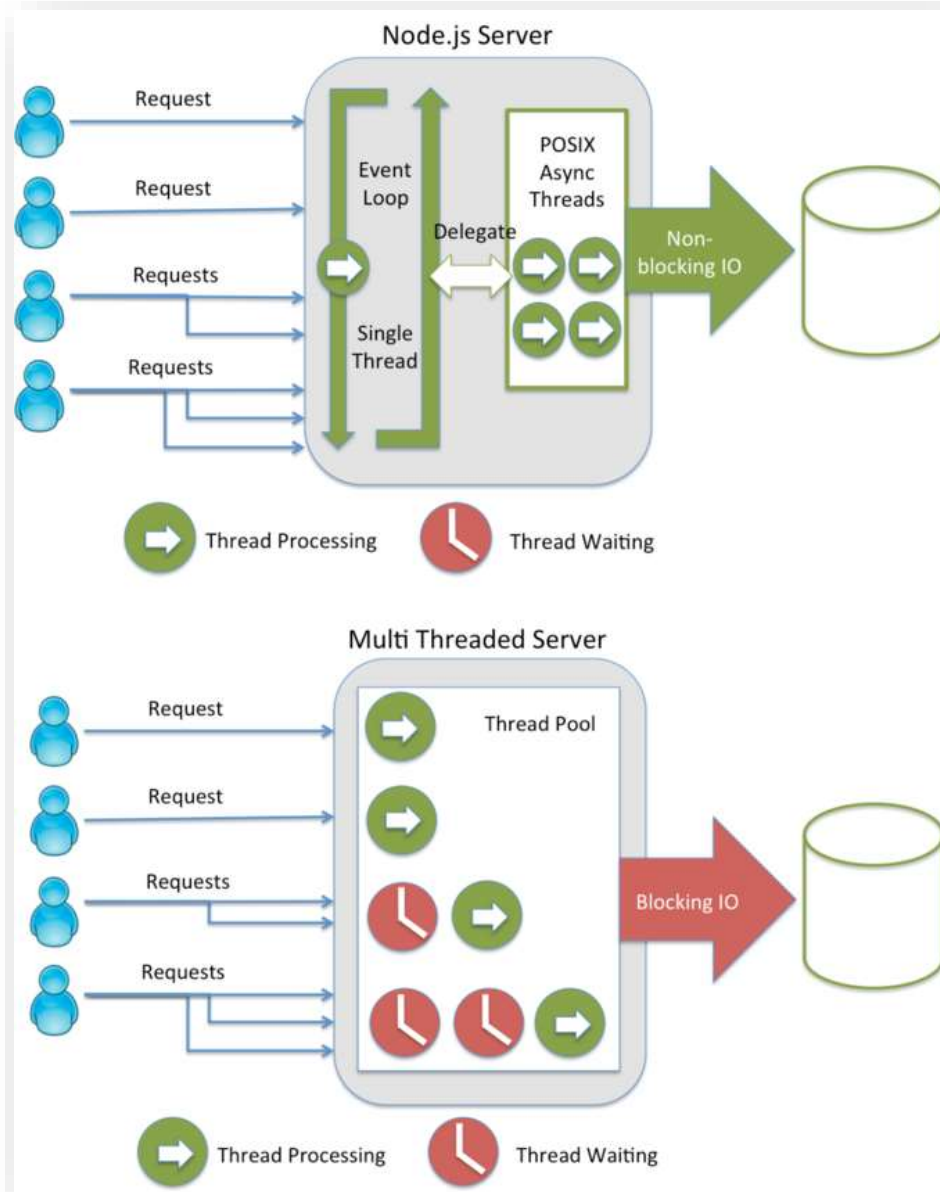
**[Quiz]** 계란 1개를 삶을 때 10분이 걸린다.

그러면 계란 10개를 삶을 때 걸리는 시간은?

**Blocking : ?**

**Non-blocking : ?**

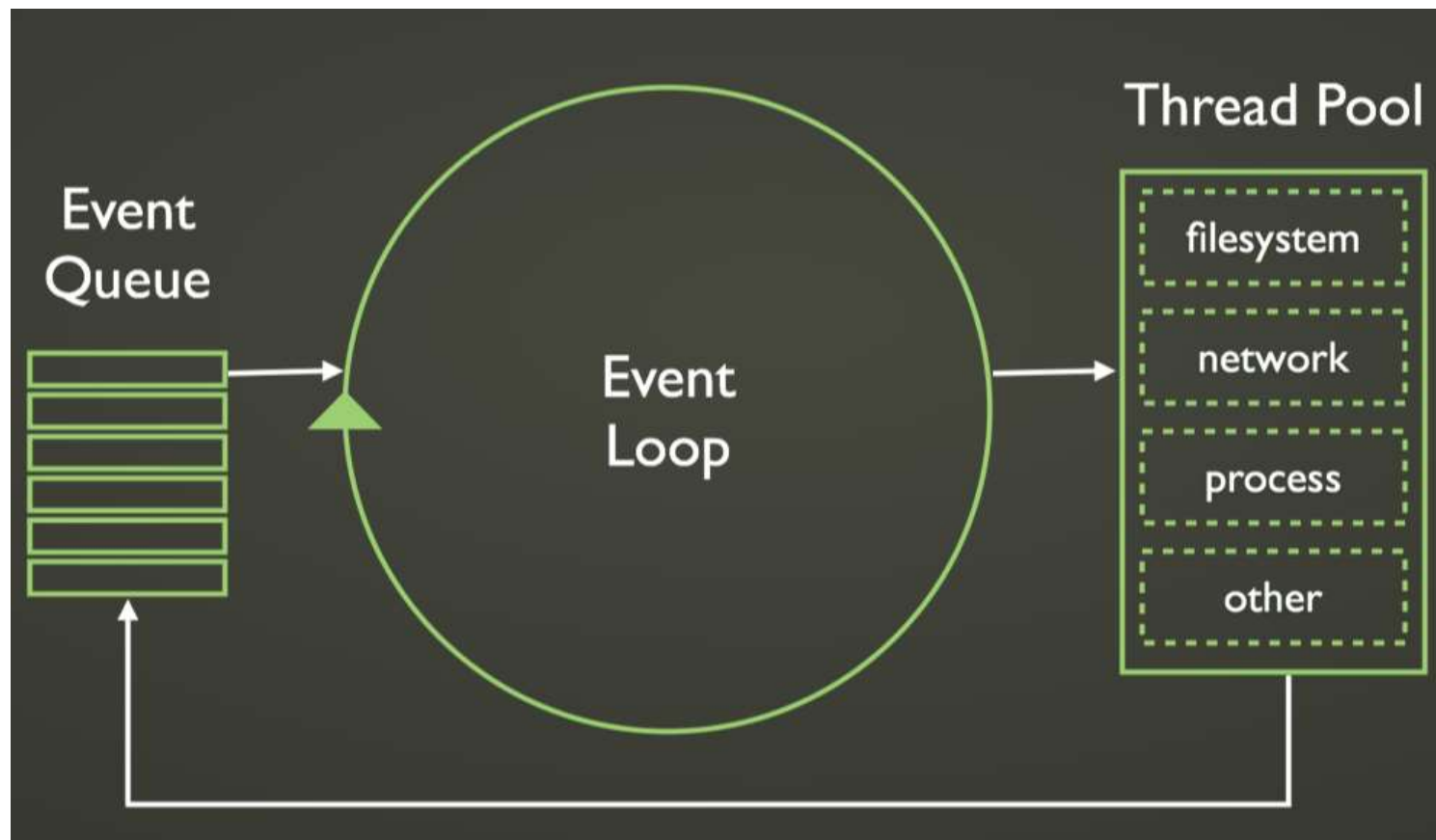
# 1.3 Non-blocking IO



**Single thread**  
**Non-blocking IO**

**Multi thread**  
**Blocking IO**

# 1.4 Non-blocking event loop





## 2.1 Install node.js



### 다운로드

최신 LTS 버전:

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



Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
macOS Binary (.tar.gz)	64-bit	
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8

<https://nodejs.org/ko/download/>



## 2.2 Verify installation

C:\> 명령 프롬프트

```
Microsoft Windows [Version 10.0.19041.450]  
(c) 2020 Microsoft Corporation. All rights reserved.
```

```
C:\Users\life21c>node -v  
v14.9.0
```

```
C:\Users\life21c>npm -v  
6.14.8
```

```
C:\Users\life21c>_
```

**node -v**

**npm -v**



## 2.3 Testing node.js – node shell

명령 프롬프트

```
C:\Users\life21c>node
Welcome to Node.js v14.9.0.
Type ".help" for more information.
> a=2
2
> b=5
5
> c=a+b
7
> a*b
10
> a/b
0.4
> a%b
2
> b%a
1
>
(To exit, press ^C again or ^D or type .exit)
>
C:\Users\life21c>_
```

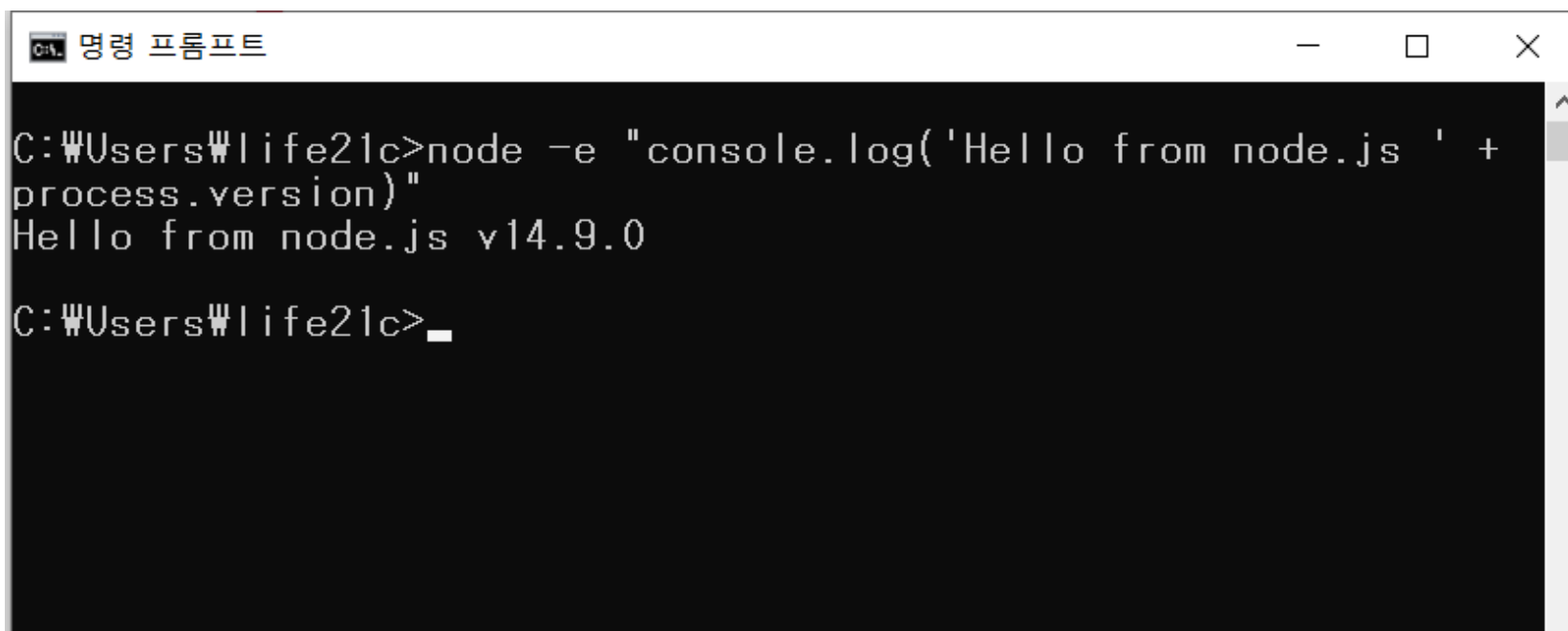
**// node shell**  
**node**  
**>**  
**>**  
**// exit**  
**^C^C**



## 2.4 Testing node.js – final test

**Final test if Node.js was installed correctly:**

```
node -e "console.log('Hello from node.js ' + process.version)"
```



A screenshot of a Windows Command Prompt window. The title bar shows the icon, the text "명령 프롬프트", and standard window controls. The command prompt shows the following sequence of text: the prompt "C:\Users\life21c>", the command "node -e \"console.log('Hello from node.js ' + process.version)\"", the output "Hello from node.js v14.9.0", and the prompt "C:\Users\life21c>\_" with a cursor.

```
C:\Users\life21c>node -e "console.log('Hello from node.js ' +  
process.version)"  
Hello from node.js v14.9.0  
C:\Users\life21c>_
```



# Node.js Project

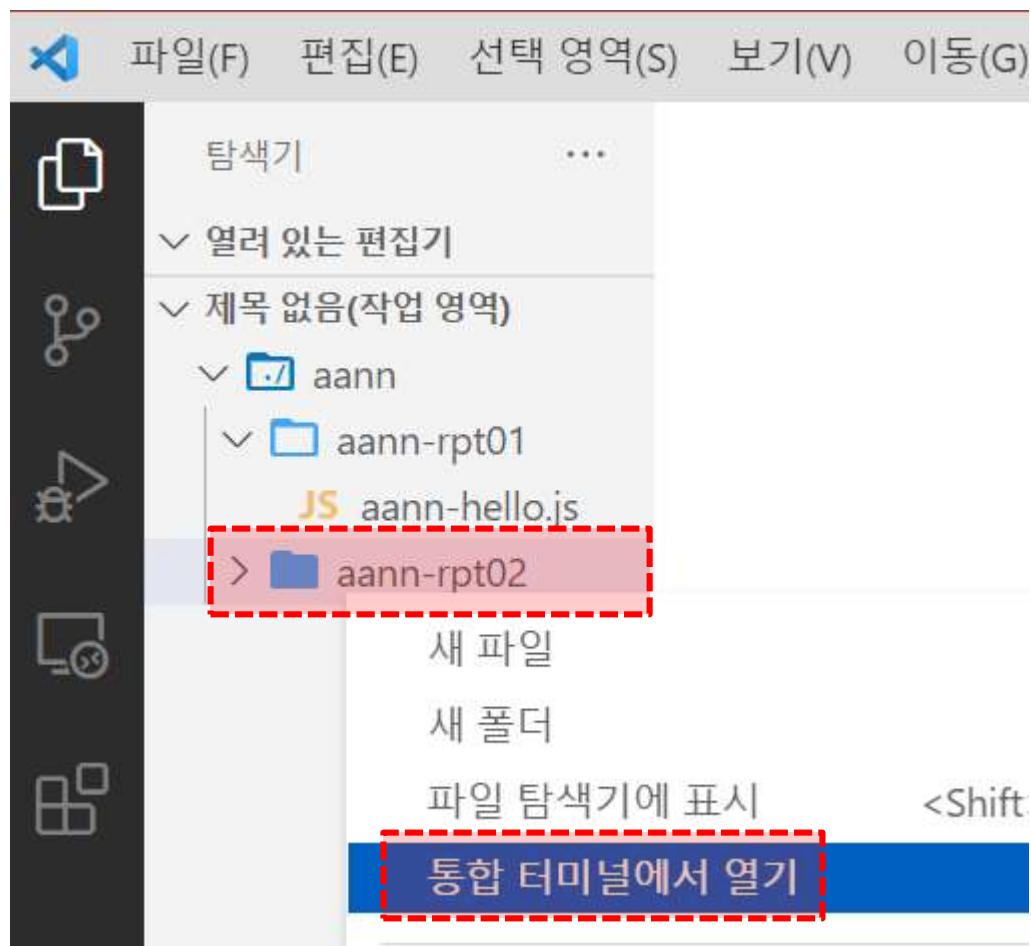
**npm init**





## 4.1 Node project : start terminal

// VScode  
작업영역에  
aann-rpt02  
폴더 만들고  
터미널 열기





## 4.1 Node project : start terminal

문제 출력 디버그 콘솔 터미널

cmd + v [icon] [icon] [icon] [icon]

Microsoft Windows [Version 10.0.19043.1165]  
(c) Microsoft Corporation. All rights reserved.

```
D:\aann\aann-rpt02>md start
```

```
D:\aann\aann-rpt02>cd start
```

```
D:\aann\aann-rpt02\start>█
```

**// node cmd**

**md start  
cd start**



## 4.2 Node project : **npm init**

```
D:\aann\aann-rpt02\start>npm init
```

This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields  
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.

Press ^C at any time to quit.

```
package name: (start)
version: (1.0.0)
description: strat node project
entry point: (index.js)
test command:
git repository:
keywords: test
author: aa00
license: (ISC) MIT
```

**// node project**  
**npm init**



## 4.3.1 Node project : package.json

```
{
  "name": "start",
  "version": "1.0.0",
  "description": "strat node project",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "test"
  ],
  "author": "aa00",
  "license": "MIT"
}
```

Is this OK? (yes)

D:\aann\aann-rpt02\start>dir

D 드라이브의 볼륨: DATA

볼륨 일련 번호: 82D1-4852

D:\aann\aann-rpt02\start 디렉터리

2021-09-07	오후 01:47	<DIR>	.
2021-09-07	오후 01:47	<DIR>	
2021-09-07	오후 01:47		255 package.json
		1개 파일	255 바이트
		2개 디렉터리	2,467,748,704,256 바이트 남음



## 4.3.2 Node project : package.json

### package.json

Node 프로젝트 설정 파일  
- json file

```
1  {
2    "name": "start",
3    "version": "1.0.0",
4    "description": "strat node project",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [
10     "test"
11   ],
12   "author": "aa00",
13   "license": "MIT"
14 }
```

Save as  
AAnn\_package.png

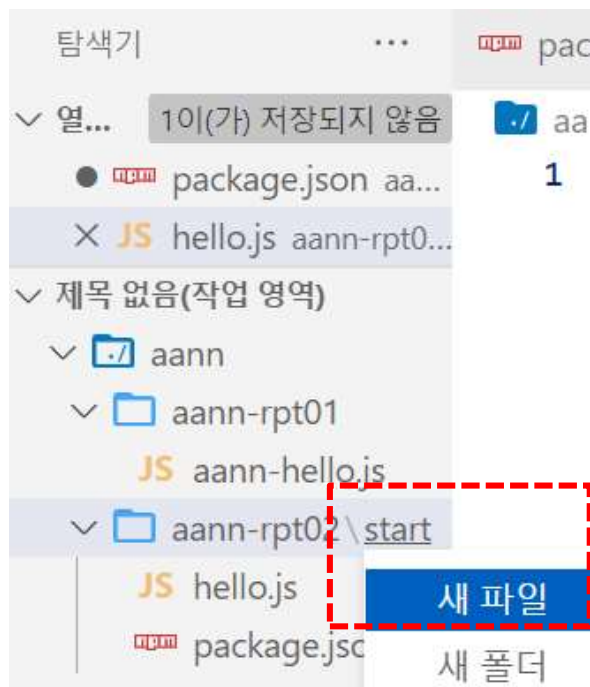


## 5. Node Apps

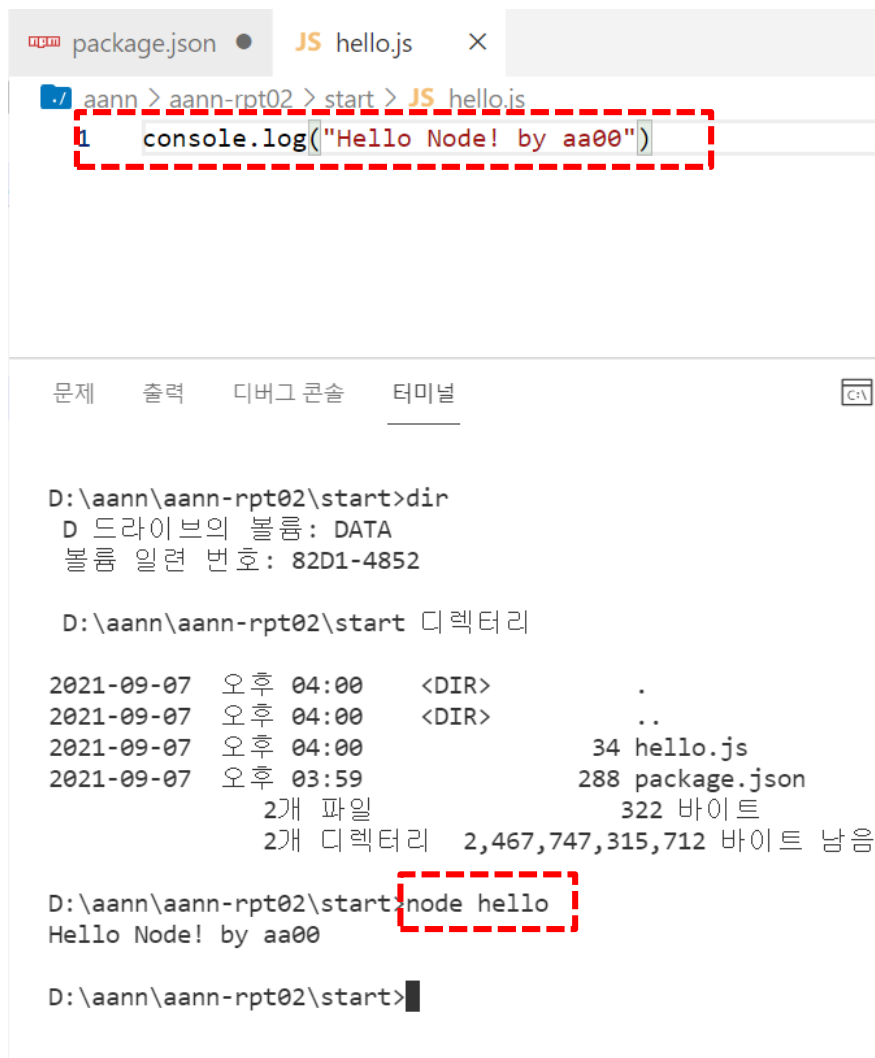
# Node Apps



# 5.1.1 Hello Node! – hello.js



**start 폴더 안에  
hello.js 파일**



**VSCode 터미널에서 실행 : node js-file-name**



## 5.1.2 Hello Node! – npm start 로 실행

package.json X JS hello.js

./ aann > aann-rpt02 > start > package.json > {} scripts > start

```
1  {
2    "name": "start",
3    "version": "1.0.0",
4    "description": "strat node project",
5    "main": "index.js",
6    "scripts": {
7      "start": "node hello.js",
8      "test": "echo \"Error: no test specified\" && exit 1"
9    },
10   "keywords": [
11     "test"
12   ],
13   "author": "aa00",
14   "license": "MIT"
15 }
16
```

문제

출력

디버그 콘솔

터미널

cmd + v

D:\aann\aann-rpt02\start>npm start

> start@1.0.0 start D:\aann\aann-rpt02\start

> node hello.js

Hello Node! by aa00





# Using function in node



## 5.2.1 Using function in node

package.json

JS hello\_function.js X

JS hello.js

aann > aann-rpt02 > start > JS hello\_function.js > ...

```
1 // hello_function.js
2 function hello(what) {
3   |   console.log("Hello " + what + " !");
4 }
5
6 hello("aa00");
7 hello("redwoods, 홍길동");
8
```

문제

출력

디버그 콘솔

터미널

cmd

D:\aann\aann-rpt02\start>node hello\_function

Hello aa00 !

Hello redwoods, 홍길동 !

D:\aann\aann-rpt02\start>



## 5.2.2 Using user-module: hello\_user\_module.js

사용자 정의 모듈

열... 1이(가) 저장되지 않음

1 그룹

- package.json aa...
- JS hello\_user\_mod...

2 그룹

- × JS hello\_call\_modul...

제목 없음(작업 영역)

- ✓ aann
  - ✓ aann-rpt01
    - JS aann-hello.js
  - ✓ aann-rpt02\start
    - JS hello\_call\_mod...
    - JS hello\_function.js
    - JS hello\_user\_mo...
    - JS hello.js
    - package.json

aann > aann-rpt02 > start > JS hello\_user\_module.js > ...

```
1 // hello_user_module.js
2 module.exports = function(what) {
3   ...
4   console.log("Hello " + what + " !");
5 }
```

JS hello\_call\_module.js ×

aann > aann-rpt02 > start > JS hello\_call\_module.js > ...

```
1 // hello_call_module.js
2 var olleh = require('./hello_user_module.js');
3
4 olleh("Node");
5 olleh("aa00");
6
7
```

문제 출력 디버그 콘솔 터미널

cmd + v

```
D:\aann\aann-rpt02\start>node hello_call_module
Hello Node !
Hello aa00 !
```

```
D:\aann\aann-rpt02\start>
```

Call user-module 'hello\_user\_module.js' from hello\_call\_module.js



# [extra code] local module : circle.js

**circle\_info.js** uses local module **circle.js**.

The screenshot shows the VS Code editor interface with the following components:

- File Explorer (Left):** Displays the project structure. The file `circle_info.js` is selected under the `start` folder.
- Editor Tabs:** The active tab is `circle.js`. Other tabs include `package.json`, `hello_user_module.js`, and `circle_info.js`.
- Code Editor:**
  - circle.js:** Contains the module definition:

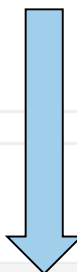
```
1 // circle.js
2 var PI = Math.PI;
3
4 module.exports.area = function (r) {
5   return PI * r * r;
6 };
7
8 module.exports.circumference = function (r) {
9   return 2 * PI * r;
10 };
```

The function definitions are highlighted with a red dashed box.
  - circle\_info.js:** Contains the code that uses the local module:

```
1 // circle_info.js
2 var circle = require('./circle');
3 console.log( 'The area of a circle of radius 4 is '
4   + circle.area(4));
5 console.log( 'The circumference of a circle of radius 4 is '
6   + circle.circumference(4));
```
- Terminal (Bottom):** Shows the command prompt output:

```
D:\aann\aann-rpt02\start>cd ..
D:\aann\aann-rpt02>node circle_info
The area of a circle of radius 4 is 50.26548245743669
The circumference of a circle of radius 4 is 25.132741228718345
```

**circle.js**



**circle\_info.js**



# Node.js Server

**1. http, tcp, file**

**2. Express**



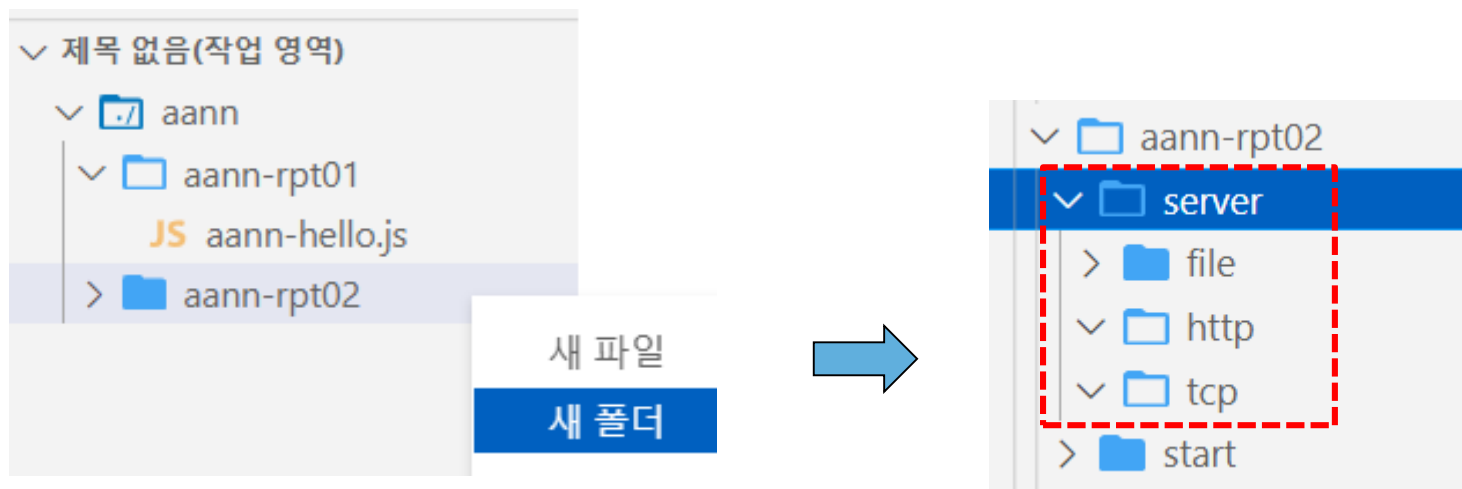
## 6. Node Server

# Node Server I.

- 1. HTTP server**
- 2. TCP server**
- 3. File upload**



## 6.1 Node server : working folders





## 6. Node Server

# Node Server I.

**1. HTTP server**

2. TCP server

3. File upload





## 6.1.1 http server

JS index.js X

aann > aann-rpt02 > server > http > JS index.js > ...

```
1 // http server : index.js
2
3 var http = require('http');
4 port = 3000;
5
6 var server = http.createServer(function(request, response) {
7     response.writeHead(200, {
8         "Content-Type": "text/plain"
9     });
10    response.write("Hello HTTP server from node.js"); // WEB response
11    response.write("\nMy ID is AA00!");
12    response.end();
13
14 });
15
16 server.listen(port);
17 console.log("Server Running on " + port +
18     ".\nLaunch http://localhost:" + port);
```

문제 출력 디버그 콘솔 터미널

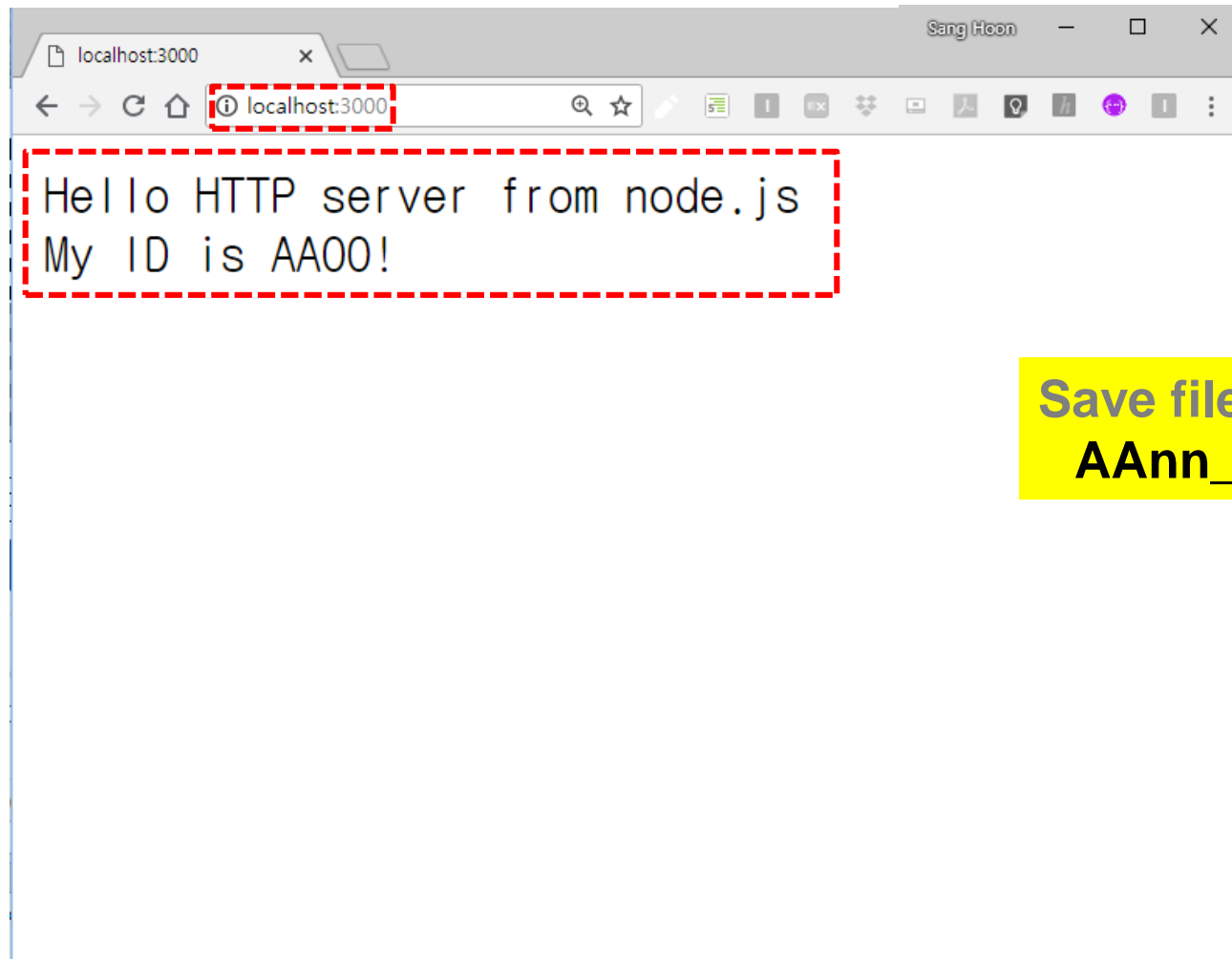
```
D:\aann\aann-rpt02\server\http>node index
Server Running on 3000.
Launch http://localhost:3000
```



## 6.1.2 http server : result 1

Server Running on 3000.

Launch `http://localhost:3000`



**Save file**

**AAnn\_HTTP.png**



## 6.1.3 http server – stop server !!! → ^C

문제      출력      디버그 콘솔      터미널

```
D:\aann\aann-rpt02\server\http>node index
```

```
Server Running on 3000.
```

```
Launch http://localhost:3000
```

```
^C
```

```
D:\aann\aann-rpt02\server\http>
```



## 6.1.4 http server – ES6 version

JS index\_ES6.js X

./ aann > aann-rpt02 > server > http > JS index\_ES6.js > server > http.createServer() callback

```
1 // http server : index_ES6.js
2
3 var http = require('http');
4 port = 3000;
5
6 var server = http.createServer((request, response) => {
7     response.writeHead(200, {
8         "Content-Type": "text/plain"
9     });
10    response.write("Hello HTTP server from node.js, ES6"); // WEB response
11    response.write("\nMy ID is AA00!");
12    response.end();
13
14 });
15
16 server.listen(port);
17 console.log("Server Running on " + port +
18     ".\nLaunch http://localhost:" + port);
19
```

문제

출력

디버그 콘솔

터미널

node + v

D:\aann\aann-rpt02\server\http>node index\_ES6

Server Running on 3000.

Launch http://localhost:3000

# [Tip] port number

★사용해도 되는 포트번호와 사용할 수 없는 포트 번호

1. 잘 알려진 포트는 0~1023 까지입니다.

(특정 프로그램들이 사용하기로 예약되어 있기 때문에 쓸 수 없는 포트 번호)

2. 등록된 포트는 1024~49151 까지입니다. (사용가능)

3. 동적 및/또는 개인 포트는 49152~65535 까지입니다. (사용가능)

참고: <http://support.microsoft.com/kb/174904/ko>

# [Tip] listen EADDRINUSE 오류 해결 -1

## 1. listen **EADDRINUSE** 오류

-사용중인 포트이거나 포트를 중지시키지 않고 종료시켰을 경우 계속 포트가 사용되고 있는데 연결 시키려고 할 때 나타나는 오류

```
Server Running on 3000.  
Launch http://localhost:3000  
events.js:154  
    throw er; // Unhandled 'error' event  
    ^  
  
Error: listen EADDRINUSE :::3000  
    at Object.exports._errnoException (util.js:856:11)  
    at exports._exceptionWithHostPort (util.js:879:20)
```

## 2. 해결 방법

- (1) cmd창에서 **netstat -ano**를 입력한 후, 로컬주소에서 사용 중인 포트 번호를 확인
- (2) 사용중인 포트번호를 확인하고 그 해당포트의 **pid번호를 확인**한다.

# [Tip] listen EADDRINUSE 오류 해결 -2

```
Node
v5.7.0
D:\Portable\nodejs\portable\data>netstat -ano

활성 연결

프로토콜  로컬 주소          외부 주소          상태          PID
TCP       0.0.0.0:135     0.0.0.0:0          LISTENING     1040
TCP       0.0.0.0:445     0.0.0.0:0          LISTENING     4
TCP       0.0.0.0:3000    0.0.0.0:0          LISTENING     14332
TCP       0.0.0.0:14430   0.0.0.0:0          LISTENING     24316
TCP       0.0.0.0:14440   0.0.0.0:0          LISTENING     24316
TCP       0.0.0.0:17500   0.0.0.0:0          LISTENING     25096
TCP       0.0.0.0:21300   0.0.0.0:0          LISTENING     16228
TCP       0.0.0.0:30403   0.0.0.0:0          LISTENING     3660
TCP       0.0.0.0:30409   0.0.0.0:0          LISTENING     3668
```

## 2. 해결 방법 (cmd에서 다음 명령 실행 후 port 3000의 pid가 제거됨을 확인) **taskkill /pid PID\_number**

```
D:\Portable\nodejs\portable\data>taskkill /pid 14332
성공: 프로세스(PID 14332)에 종료 신호를 보냈습니다.
```

```
D:\Portable\nodejs\portable\data>netstat -ano
```

활성 연결

프로토콜	로컬 주소	외부 주소	상태	PID
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	1040
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:14430	0.0.0.0:0	LISTENING	24316



## 6. Node Server

# Node Server I.

1. HTTP server

**2. TCP server**

3. File upload



## 6.2.1 tcp server (socket connection)

JS server.js X

./ aann > aann-rpt02 > server > tcp > JS server.js > ...

```

1  // tcp server (network server)
2  var net = require('net');
3  var port = 3000;
4
5  // Network connection using socket
6  var server = net.createServer(function(socket) {
7      console.log("Connection from " + socket.remoteAddress);
8      socket.end("Hello AA00! from localhost:3000");
9  });
10
11 server.listen(port, "127.0.0.1");
12 console.log("Network server started at port : " + port);
13

```

Socket으로 전송

문제 출력 디버그 콘솔 터미널

node + v

D:\aann\aann-rpt02\server\tcp>node server

Network server started at port : 3000

## 6.2.2 tcp client

```
JS server.js JS client.js x
aann > aann-rpt02 > server > tcp > JS client.js > ...

1 // tcp client
2 var net = require('net');
3 var port = 3000;
4 var client = new net.Socket();
5 // Connection using socket
6 client.connect(port, "127.0.0.1");
7 // Receive data from socket
8 client.on('data', function (data) {
9   console.log('Data: ' + data);
10  client.destroy();
11 });
12
13 // Add a 'close' event handler for the client socket
14 client.on('close', function () {
15   console.log('Connection closed');
16 });
```

Socket으로 전송되는 데이터를 처리하고 종료

문제 출력 디버그 콘솔 터미널

```
D:\aann\aann-rpt02\server\tcp>node server
Network server started at port : 3000
Connection from 127.0.0.1
█
```

```
D:\aann\aann-rpt02\server\tcp>node client
Data: Hello AA00! from localhost:3000
Connection closed
```

```
D:\aann\aann-rpt02\server\tcp>
```



## 6.2.3 tcp server & client : result

```
JS server.js x JS client.js x
aann > aann-rpt02 > server > tcp > JS server.js > ...
1 // tcp server (network server)
2 var net = require('net');
3 var port = 3000;
4
5 // Network connection using socket
6 var server = net.createServer(function(socket)
7   console.log("Connection from " + socket.remoteAddress);
8   socket.end("Hello AA00! from localhost:3000");
9 });
10
11 server.listen(port, "127.0.0.1");
12 console.log("Network server started at port : " + port);
13

aann > aann-rpt02 > server > tcp > JS client.js > ...
1 // tcp client
2 var net = require('net');
3 var port = 3000;
4 var client = new net.Socket();
5 // Connection using socket
6 client.connect(port, "127.0.0.1");
7 // Receive data from socket
8 client.on('data', function (data) {
9   console.log('Data: ' + data);
10  client.destroy();
11 });
12
13 // Add a 'close' event handler for the client
14 client.on('close', function () {
15   console.log('Connection closed');
16 });

D:\aann\server\tcp>node server
Network server started at port : 3000
Connection from 127.0.0.1
Connection from 127.0.0.1
Connection from 127.0.0.1
Connection from 127.0.0.1

D:\aann\server\tcp>node client
Data: Hello AA00! from localhost:3000
Connection closed

D:\aann\server\tcp>node client
Data: Hello AA00! from localhost:3000
Connection closed

D:\aann\server\tcp>node client
Data: Hello AA00! from localhost:3000
Connection closed

D:\aann\server\tcp>
```

Save file

AAnn\_TCP\_Log.png



## 6. Node Server

# Node Server I.

1. HTTP server
2. TCP server
- 3. File upload**

## 6.3.1 file upload using module 'formidable'

JS file\_server.js X

```
aann > aann-rpt02 > server > file > JS file_server.js > ...
1 // File upload using formidable node module
2 var formidable = require('formidable'),
3     http = require('http'),
4     util = require('util'),
5     port = 3663;
6
7 http.createServer(function(req, res) {
8   if (req.url == '/upload' && req.method.toLowerCase() == 'post') {
9     // parse a file upload
10    var form = new formidable.IncomingForm();
11
12    form.parse(req, function(err, fields, files) {
13      res.writeHead(200, {'content-type': 'text/plain'});
14      res.write('received upload:\n\n');
15      res.end(util.inspect({fields: fields, files: files}));
16    });
17    return;
18  }
19  // show a file upload form
```

문제 출력 디버그 콘솔 터미널

node + v

```
D:\aann\aann-rpt02\server\file>node file_server
File server Running on 3663.
Launch http://localhost:3663
```



## 6.3.2 file upload : npm install formidable

문제 출력 디버그 콘솔 터미널

cmd + v [icon] [icon] [icon] [icon]

```
D:\aann\aann-rpt02\server\file>npm install formidable
npm WARN saveError ENOENT: no such file or directory, open 'D:\aann\aann-rpt02\server\file\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'D:\aann\aann-rpt02\server\file\package.json'
npm WARN file No description
npm WARN file No repository field.
npm WARN file No README data
npm WARN file No license field.
```

```
+ formidable@1.2.2
added 1 package and audited 1 package in 0.96s
```

```
1 package is looking for funding
  run `npm fund` for details
```

```
found 0 vulnerabilities
```

```
D:\aann\aann-rpt02\server\file>
```

```
D:\aann\aann-rpt02\server\file>dir
```

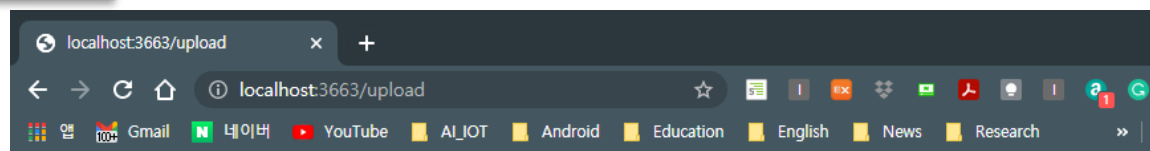
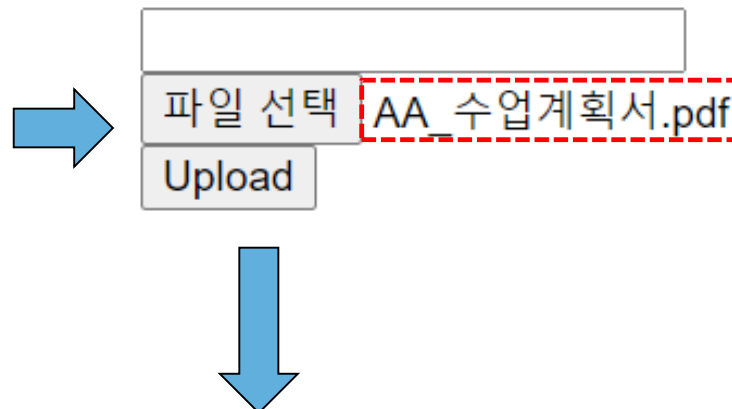
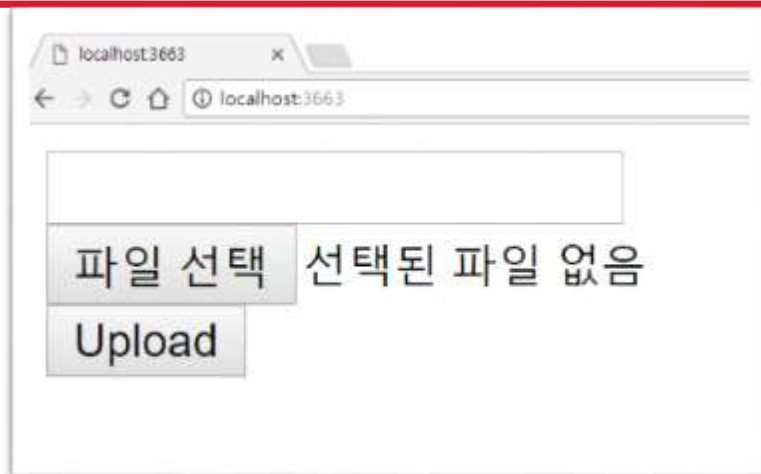
```
D 드라이브의 볼륨: DATA
볼륨 일련 번호: 82D1-4852
```

```
D:\aann\aann-rpt02\server\file 디렉터리
```

2021-09-07	오후 10:45	<DIR>	.
2021-09-07	오후 10:45	<DIR>	..
2017-12-29	오후 05:21		1,048 file_server.js
2021-09-07	오후 10:45	<DIR>	node_modules
2021-09-07	오후 10:45		323 package-lock.json
	2개 파일		1,371 바이트
	3개 디렉터리		2,467,732,140,032 바이트 남음



# 6.3.3 file upload



received upload:

```
{
  fields: { title: '' },
  files: {
    upload: File {
      _events: [Object: null prototype] {},
      _eventsCount: 0,
      _maxListeners: undefined,
      size: 112398,
      path: 'C:\\Users\\lfe21c\\AppData\\Local\\Temp\\upload_694cba275a78a8edfed4e771256fb455',
      name: 'AA_&#49688;&#50629;&#44228;&#54925;&#49436;.pdf',
      type: 'application/pdf',
      hash: null,
      lastModifiedDate: 2021-09-07T13:57:11.868Z,
      _writeStream: [WriteStream],
      [Symbol(kCapture)]: false
    }
  }
}
```

**Save file**  
**AAnn\_Upload.png**



# [Practice]

## ◆ [wk02]

- Node module : **aanninfo.js**
- Upload folder: **aann-rpt02**





# [practice] local module : aanninfo.js

**index\_aann.js** uses local module **aanninfo.js** in start subfolder.

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders 'aa00', 'aann', 'server', and 'start'. The 'start' folder contains several JavaScript files, including 'index\_aann.js' which is selected. The code editor shows the content of 'index\_aann.js' with line numbers 1 through 7. The code uses the 'require' function to load the 'aanninfo' module and calls 'myinfo' with different parameters.

```
1 // index_aann.js
2
3 var myinfo = require('./aanninfo');
4
5 myinfo("aa00", "Redwoods", '010-1234-5678');
6
7 myinfo("aa55", "Comsi", '010-5678-1234');
```

My Info  
ID : aa00  
Name : Redwoods  
Phone : 010-1234-5678

My Info  
ID : aa55  
Name : Comsi  
Phone : 010-5678-1234

[Finished in 0.2s]

Save as  
**AAnn\_info.png**



# [practice] local module : **aanninfo.js**

How to make aanninfo.js in start subfolder.

1. **Make local module – aanninfo.js**
2. **Call aanninfo.js from index\_aann.js.**
3. **Capture your result.**

```
index_aann.js  x  aanninfo.js  x
1 // aanninfo.js
2
3 module.exports = function(id, name, phone) {
4     console.log("My Info");
5     console.log("ID : " + id);
6     console.log("Name : " + name);
7     console.log("Phone : " + phone + "\n");
8 }
```

[\[참고\] Node local module 만들기](#)

## ◆ [Target of this week]

**My Info using node module – aanninfo.js**

### **Upload folder : aann-rpt02**

#### **- 제출할 파일들**

- ① **AAnn\_package.png**
- ② **AAnn\_HTTP.png**
- ③ **AAnn\_TCP\_Log.png**
- ④ **AAnn\_Upload.png**
- ⑤ **AAnn\_info.png**
- ⑥ **start folder**
- ⑦ **server folder**

# [Upload to github]

## ◆ [wk02]

- upload all work of this week
- Use repo “aann” in github
- upload folder “aann-rpt02”  
in your github.

## ● References & good sites

- ✓ <http://www.arduino.cc> Arduino Homepage
- ✓ <http://www.nodejs.org/ko> Node.js
- ✓ <https://plot.ly/> plotly
- ✓ <https://www.mongodb.com/> MongoDB
- ✓ <https://www.anaconda.com/> Anaconda
- ✓ <http://www.github.com> GitHub
- ✓ <https://colab.research.google.com/> Colab