

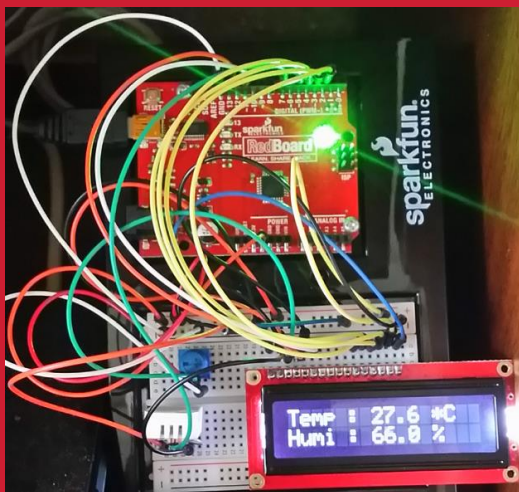
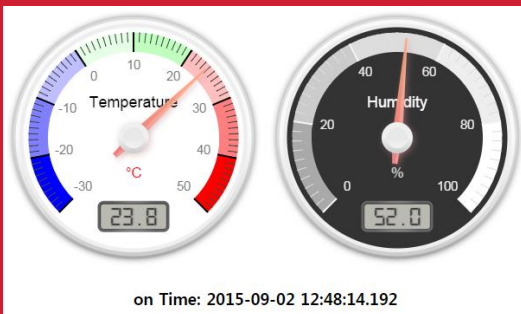


HW-SW-Connectivity

[wk03]

portable **Node.js**

web server & Express

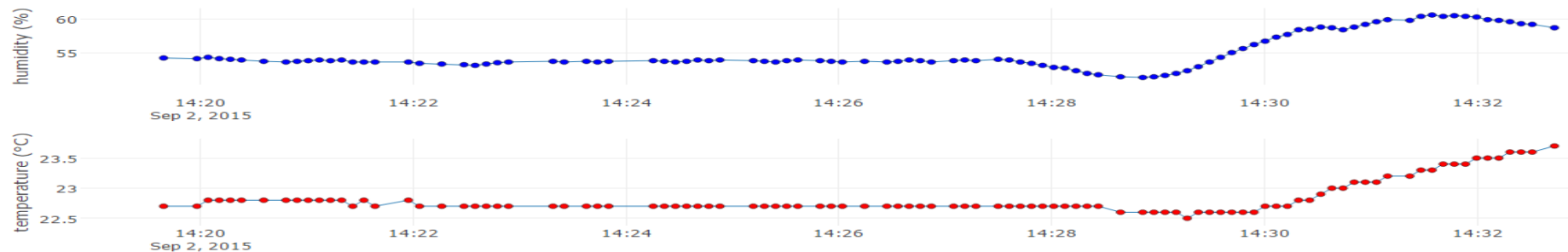


Basic HW and SW Integration using
Arduino & Javascript

COMSI, INJE University

2nd semester, 2017

Email : yish@inje.ac.kr





주간계획서

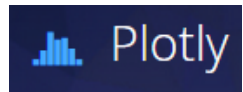
2017학년도 2학기

주간계획서			
주차	수업방법	수업내용	과제물
1	이론	교육과정 안내	가상강의 등록
2	이론/실습	모바일 서버 기초 : node.js 설치	
3	이론/실습	모바일 서버 프로그래밍 : node.js 응용	점검과제
4	이론/실습	기초 하드웨어: 아두이노 소개	점검과제
5	이론/실습	기초 하드웨어 프로그래밍: 아두이노 센서	프로젝트 1
6	이론/실습	시각화 프로그래밍: plotly.js	점검과제
7	이론/실습	아두이노 센서 신호 처리 및 시각화	프로젝트 1I
8	시험 또는 실습과제	중간고사	
9	이론/실습	아두이노 센서 신호 시각화 고급 프로그래밍	점검과제
10	이론/실습	모바일 데이터베이스 I: mongoDB 설치	점검과제
11	이론/실습	모바일 데이터베이스 II: mongoDB 응용 프로그래밍, Compass 활용	프로젝트 1II
12	이론/실습	모바일 클라이언트 프로그래밍 I: Angular.js 기초	점검과제
13	이론/실습	모바일 클라이언트 프로그래밍 II: Angular.js 응용 프로그래밍	점검과제
14	이론/실습	하드웨어와 소프트웨어 융합 IOT 프로젝트	프로젝트 IV
15	이론/실습	기말고사	

Weekly schedule of HSC– 2nd semester, 2017



- **wk01** : Introduction to class and enrollment in cyber class (**Sublime text 3 install**)
- **wk02** : Basic mobile server : **node.js install** and test
- **wk03** : Mobile server programming : node.js App
- **wk04** : Basic HW : Arduino I. – circuit & programming, **Arduino SW install**
- **wk05** : Basic HW : Arduino II. – sensor circuit & programming
- **wk06** : Visualization using Javascript – **plyly.js** and gauge.js
- **wk07** : Project-1 : Handling and visualization of signals from various sensors
- **wk08** : Mid-term exam.
- **wk09** : Advanced programming to visualize signals from sensors
- **wk10** : Mobile database I : **Mongo DB install**
- **wk11** : Mobile database II : Mongo DB App.
- **wk12** : Mobile client programming I : **Angular.js install**
- **wk13** : Mobile client programming II : Angular.js App.
- **wk14** : Project-2 : IOT project fusing HW & SW
- **wk15** : Final exam.



◆ [Target of this week]

My Info using node module – aanninfo.js

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

제출파일명 : AAnn_Rpt01.zip

■ 압축할 파일들

- ① **AAnn_package.png**
- ② **AAnn_info.png**
- ③ **aanninfo.js**

[\[참고\] Node local module 만들기](#)



[practice] local module : aanninfo.js

index_aann.js uses local module **aanninfo.js**.

```
E:\Coder\NodeJS\Portable\Data\aa00\start\index_aa00.js (src, Data) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  src
  Data
    aa00
      start
        aa00info.js
        hello.js
        hello_call.js
        hello_function.js
        hello_module.js
        index_aa00.js
        package.json
      node_modules
      settings
      npm
      npm.cmd
      PortableApps.comLauncher

index_aa00.js
1 // index_aann.js
2
3 var myinfo = require('./aa00info');
4
5 myinfo('aa00', 'Redwoods', '010-8558-5527');
6
7 myinfo('aa88', 'COMSI', '010-1234-5678');
8
```

```
My Info
ID : aa00
Name : Redwoods
Phone : 010-8558-5527
```

```
My Info
ID : aa88
Name : COMSI
Phone : 010-1234-5678
```

[Finished in 0.2s]

```
1 // aannninfo.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 }
```

◆ [Target of this week]

My Info using node module – aanninfo.js

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

제출파일명 : AAnn_Rpt01.zip

■ 압축할 파일들

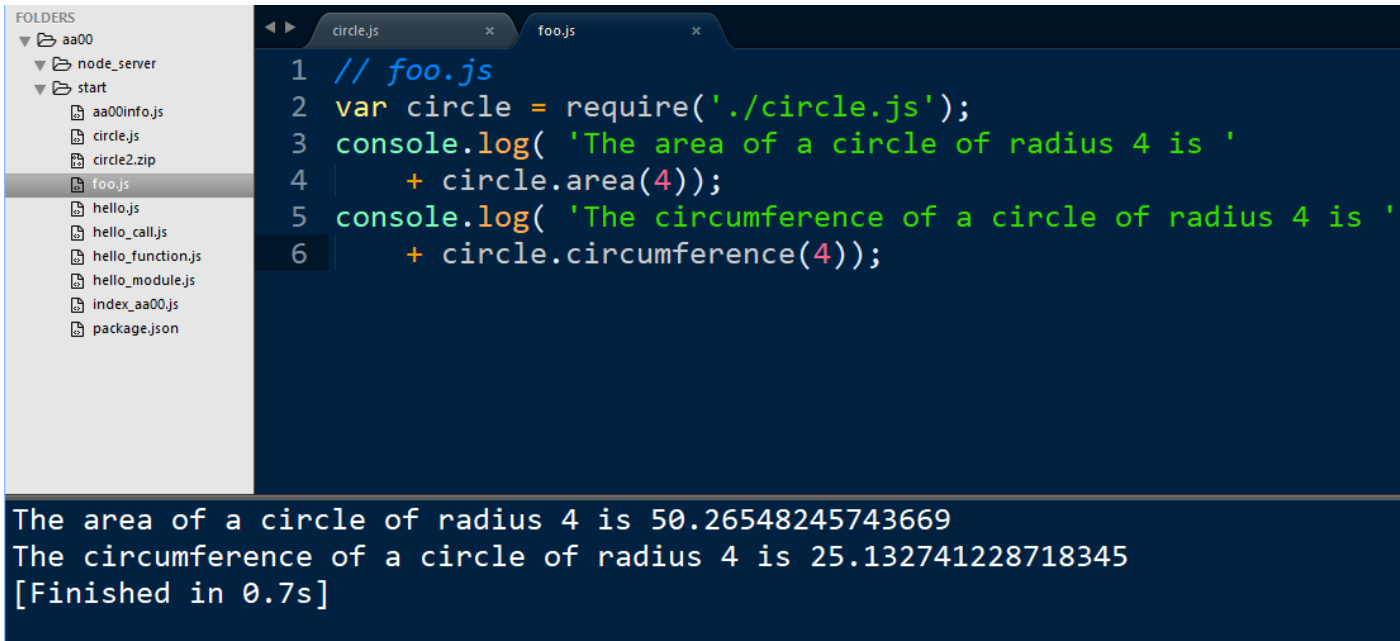
- ① **AAnn_package.png**
- ② **AAnn_info.png**
- ③ **aanninfo.js**

[\[참고\] Node local module 만들기](#)



[extra code] local module : circle.js

foo.js uses local module **circle.js**.



```
1 // foo.js
2 var circle = require('./circle.js');
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));
```

The area of a circle of radius 4 is 50.26548245743669
The circumference of a circle of radius 4 is 25.132741228718345
[Finished in 0.7s]

```
1 // circle.js
2
3 var PI = Math.PI;
4
5 exports.area = function (r) {
6   return PI * r * r;
7 };
8
9 exports.circumference = function (r) {
10   return 2 * PI * r;
11 };
```



What is targets of this week?

Node.js Server

1. http, tcp, file

2. Express



6. Node Server

Node Server I.

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



6.1.1 http server

FOLDERS

- ▼ aa00
 - ▼ node_server
 - ▼ HTTPServer
 - index.js
 - ▼ start
 - aa00info.js
 - circle.js
 - circle2.zip
 - foo.js
 - hello.js
 - hello_call.js
 - hello_function.js
 - hello_module.js
 - index_aa00.js
 - package.json

```
index.js  foo.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.end();
12
13 });
14
15 server.listen(port);
16 console.log("Server Running on " + port +
17     ".\nLaunch http://localhost:" + port);
18
```

```
Server Running on 3000.
Launch http://localhost:3000
```

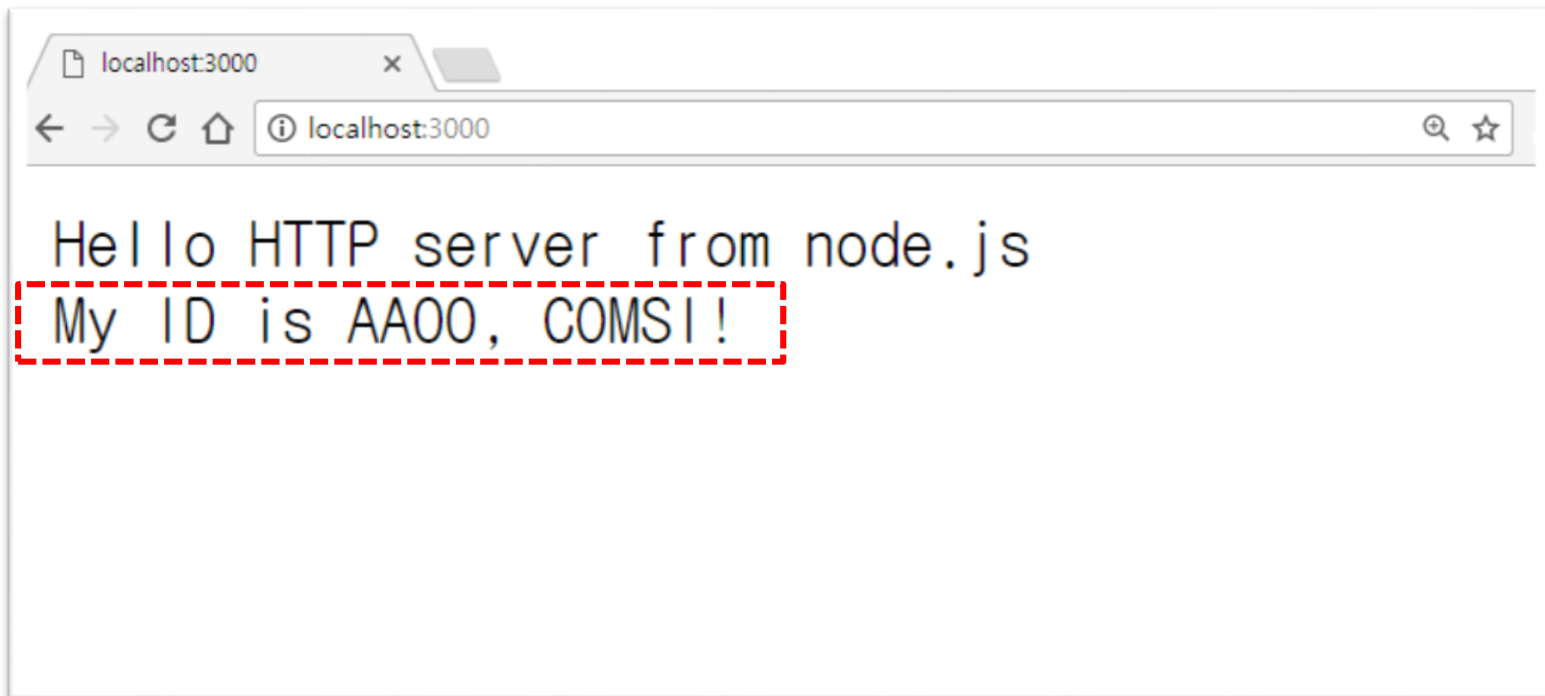


6.1.2 http server : result 1



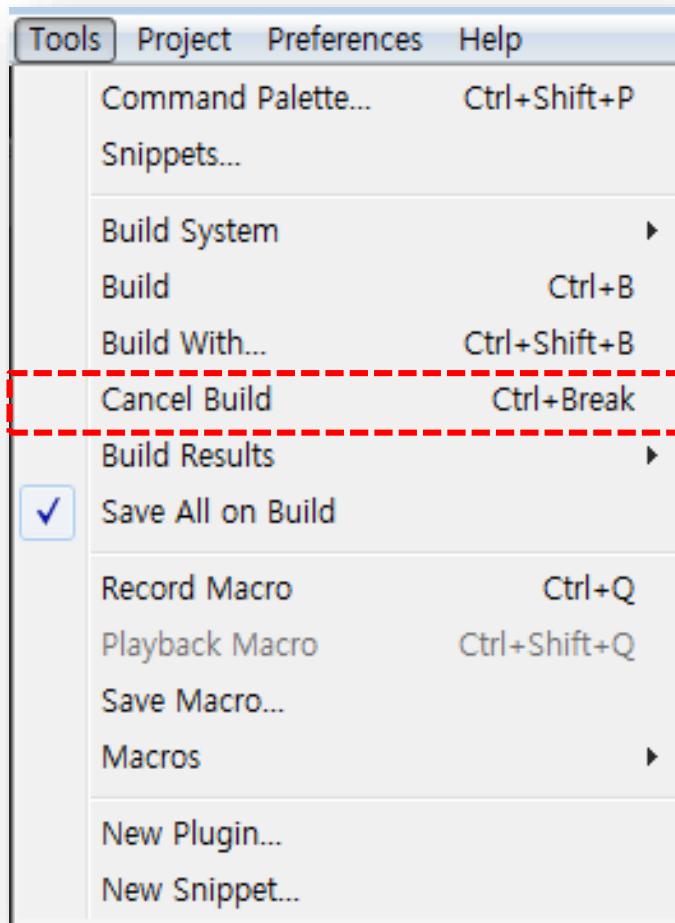


6.1.2 http server : result 2



Save file
AAnn_HTTP.png

6.1.3 http server – stop



```
Server Running on 3000.  
Launch http://localhost:3000  
[Cancelled]
```

[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



6.2.1 tcp server (socket connection)

FOLDERS

- aa00
 - node_server
 - HTTPServer
 - TCPServer
 - server.js
 - wk03_code.txt
 - start

```
index.js  wk03_code.txt  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 COMSI! from localhost:3000");
9  });
10
11 server.listen(port, "127.0.0.1");
12 console.log("Network server started at port : " + port);
13
```

Network server started at port : 3000



6.2.2 tcp server [cmd start: node server.js]

```
NodeJS - node server
D:\Portable\NodeJSPortable\Data>dir
D 드라이브의 볼륨: DATA
볼륨 일련 번호: 7A01-106A

D:\Portable\NodeJSPortable\Data 디렉터리

2017-09-12 오후 05:52 <DIR> .
2017-09-12 오후 05:52 <DIR> ..
2016-02-26 오전 04:36 6,148 .DS_Store
2017-09-09 오전 11:42 <DIR> aa00
2017-09-06 오후 01:40 <DIR> node_modules
2015-10-15 오전 07:21 296 npm
2015-10-15 오전 07:21 204 npm.cmd
2017-09-12 오후 05:52 88 PortableApps.comLauncherRuntimeData-NodeJSPortable.ini
2017-09-06 오후 01:40 <DIR> settings
4개 파일 6,736 바이트
5개 디렉터리 925,290,119,168 바이트 남음

D:\Portable\NodeJSPortable\Data>cd aa00
D:\Portable\NodeJSPortable\Data\aa00>cd n*
D:\Portable\NodeJSPortable\Data\aa00\node_server>cd T*
D:\Portable\NodeJSPortable\Data\aa00\node_server\TCPServer>dir
D 드라이브의 볼륨: DATA
볼륨 일련 번호: 7A01-106A

D:\Portable\NodeJSPortable\Data\aa00\node_server\TCPServer 디렉터리

2017-09-12 오후 05:46 <DIR> .
2017-09-12 오후 05:46 <DIR> ..
2017-09-12 오후 05:48 371 server.js
1개 파일 371 바이트
2개 디렉터리 925,290,119,168 바이트 남음

D:\Portable\NodeJSPortable\Data\aa00\node_server\TCPServer>node server
Network server started at port : 3000
```

**Node cmd에서
TCP 서버 실행**



6.2.3 tcp client

FOLDERS

- ▼ aa00
 - ▼ node_server
 - ▶ HTTPServer
 - ▼ TCPServer
 - client.js
 - server.js
 - wk03_code.txt
 - ▶ start

```
index.js x wk03_code.txt x server.js x client.js x
1 // tcp client
2 var net = require('net');
3 var port = 3000;
4 var client = new net.Socket();
5
6 client.connect(port, "127.0.0.1");
7
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 });
17
```

```
Data: Hello COMSI! from localhost:3000
Connection closed
[Finished in 0.3s]
```



6.2.4 tcp client : result

```
NodeJS - node server

D:\Portable\NodeJSPortable\Data\aa00\node_server\TCPServer 디렉터리

2017-09-12 오후 05:46 <DIR> .
2017-09-12 오후 05:46 <DIR> ..
2017-09-12 오후 05:48      371 server.js
      1개 파일      371 바이트
      2개 디렉터리  925,290,119,168 바이트 남음

D:\Portable\NodeJSPortable\Data\aa00\node_server\TCPServer>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
■
```

```
Data: Hello COMSI! from localhost:3003
```

```
Connection closed
[Finished in 0.1s]
```

Save file
AAnn_TCP.png



6.3.1 file upload using module 'formidable'

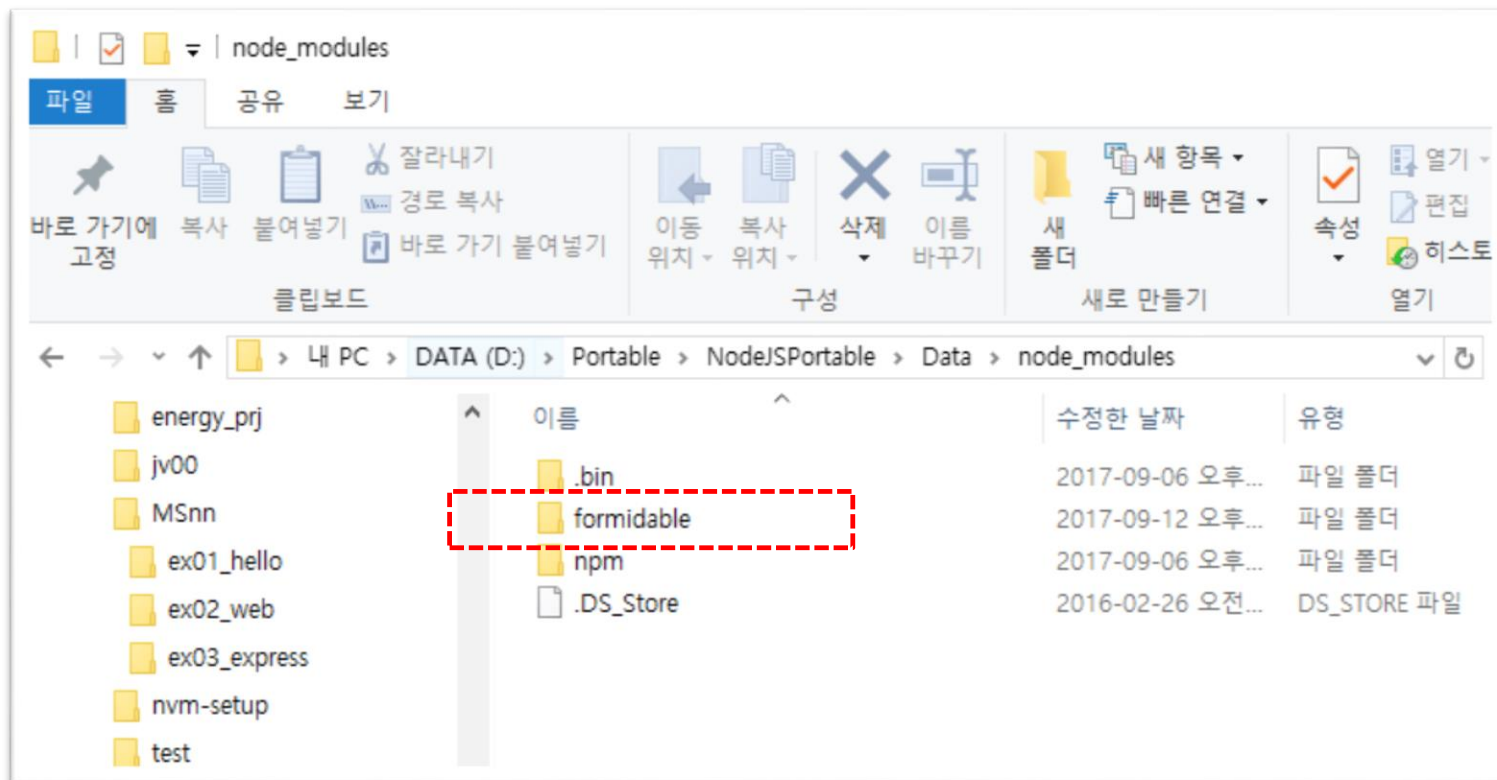
FOLDERS

- ▼ aa00
 - ▼ node_server
 - ▼ FileUpload
 - file_server.js
 - HTTPServer
 - index.js
 - TCPServer
 - client.js
 - server.js
 - wk03_code.txt
 - ▶ start

```
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
20  res.writeHead(200, {'content-type': 'text/html'});
21  res.end(
22    '<form action="/upload" enctype="multipart/form-data" method="post'
```

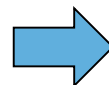
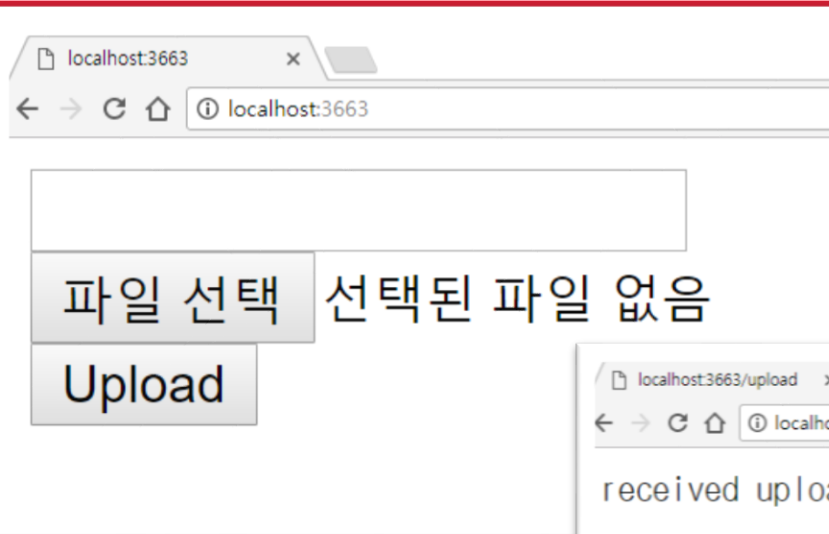
Server Running on 3663.
Launch <http://localhost:3663>

6.3.2 file upload : npm install formidable



**Portable node.js에서는
외부 node module이
Data/node_modules에 저장된다.**

6.3.3 file upload



파일 선택 onprint.pdf

Upload



Save file

AAnn_Upload.png



7. Node Server

Node Server II.

- 1. Express server**
- 2. Full Express App**
- 3. My Express App**



7.1.1 Express server test

Step 1 : npm init

Step 2 : npm install --save express

Step 3 : Write Express code

Step 4 : Run app.js

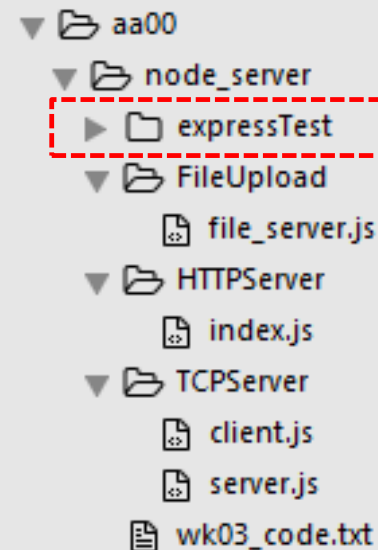
Step 5 : http://localhost:3000



7.1.2 Express server test: npm init

1. Make folder expressTest
2. Go to the folder, "expressTest"
3. npm init

FOLDERS



cmd: npm

```
D:\Portable\NodeJSPortable\Data\aa00\node_server>dir
```

```
D 드라이브의 볼륨: DATA  
볼륨 일련 번호: 7A01-106A
```

```
D:\Portable\NodeJSPortable\Data\aa00\node_server 디렉터리
```

```
2017-09-13 오전 08:13 <DIR> .  
2017-09-13 오전 08:13 <DIR> ..  
2017-09-13 오전 08:13 <DIR> expressTest  
2017-09-12 오후 06:17 <DIR> FileUpload  
2017-09-09 오전 11:53 <DIR> HTTPServer  
2017-09-12 오후 06:05 <DIR> TCPServer  
2017-09-13 오전 08:07 1,901 wk03_code.txt  
1개 파일 1,901 바이트  
6개 디렉터리 925,280,841,728 바이트 남음
```

```
D:\Portable\NodeJSPortable\Data\aa00\node_server>cd e*
```

```
D:\Portable\NodeJSPortable\Data\aa00\node_server\expressTest>dir
```

```
D 드라이브의 볼륨: DATA  
볼륨 일련 번호: 7A01-106A
```

```
D:\Portable\NodeJSPortable\Data\aa00\node_server\expressTest 디렉터리
```

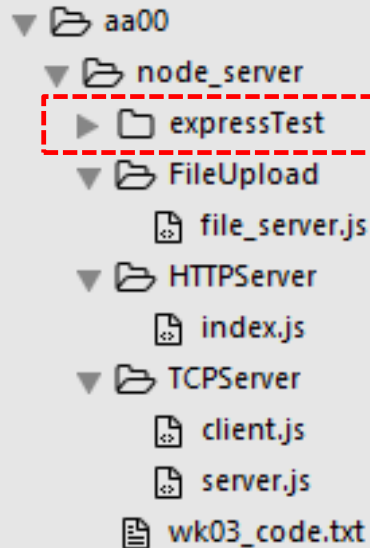
```
2017-09-13 오전 08:13 <DIR> .  
2017-09-13 오전 08:13 <DIR> ..  
0개 파일 0 바이트  
2개 디렉터리 925,280,841,728 바이트 남음
```



7.1.2 Express server test: npm init

1. Make folder **expressTest**
2. Go to the folder, **“expressTest”**
3. **npm init**

FOLDERS



```
npm
D:\Portable\NodeJSPortable\Data\aa00\node_server\expressTest>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 json' for definitive documentation on these fields
and exactly what they do.

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

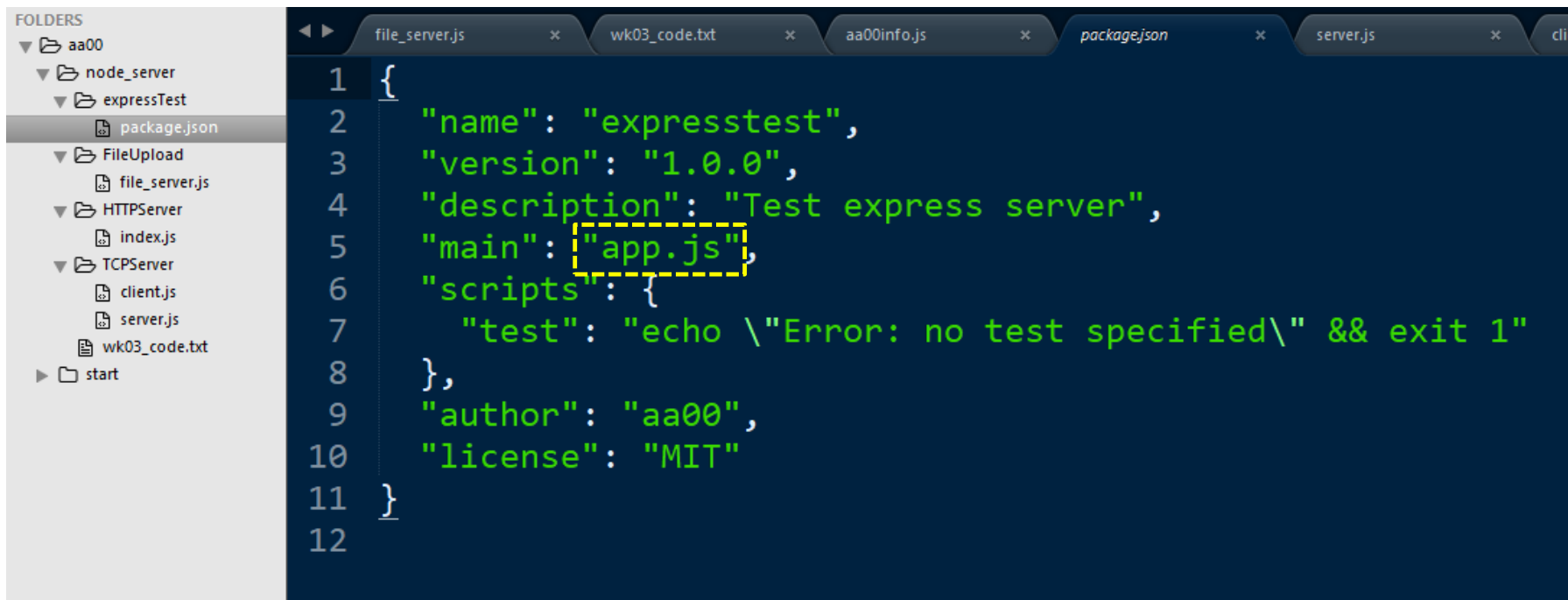
Press ^C at any time to quit.
name: (expressTest)
Sorry, name can no longer contain capital letters.
name: (expressTest) expresstest
version: (1.0.0)
description: Test express server
entry point: (index.js) app.js
test command:
git repository:
keywords:
author: aa00
license: (ISC) MIT
About to write to D:\Portable\NodeJSPortable\Data\aa00\node_server\expressTest\package.json:
{
  "name": "expresstest",
  "version": "1.0.0",
  "description": "Test express server",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "aa00",
  "license": "MIT"
}

Is this ok? (yes)
```



7.1.3 Express server test: package.json

package.json



```
1 {  
2   "name": "expresstest",  
3   "version": "1.0.0",  
4   "description": "Test express server",  
5   "main": "app.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "author": "aa00",  
10  "license": "MIT"  
11 }  
12
```




7.1.4 Express server test: express install

npm install **-save** express

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 like 'aa00', 'node_server', 'expressTest', 'FileUpload', 'HTTPServer', 'TCPServer', and 'start'. The 'expressTest' folder is expanded, showing 'node_modules' and 'package.json'. The 'package.json' file is selected and its content is displayed in the code editor. The code in 'package.json' is as follows:

```
1 {
2   "name": "expresstest",
3   "version": "1.0.0",
4   "description": "Test express server",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "author": "aa00",
10  "license": "MIT",
11  "dependencies": {
12    "express": "^4.15.4"
13  }
14 }
```

expressTest 폴더 내의 **node_modules** subfolder에 **express server modules**들이 저장되어 서버와 연동된다. 그리고 **package.json**에 **express** 모듈 정보가 저장된다.



7.1.5 Express server test: app.js

app.js

```
1 // app.js, expres cerver
2 var express = require('express');
3 var app = express();
4 var port = 3000;
5
6 app.get('/', function(req, res) {
7   res.send('<a href="/hello">Hello Page</a>');
8 });|
9
10 app.get('/hello', function(req, res) {
11   res.send('Hello HSC (하소연)');
12 });
13
14 var server = app.listen(port, function() {
15   console.log('Listening on port %d', server.address().port);
16 });
17
```




7.1.6 Express server test: run app.js

^B (실행 전 Tools > Cancel Build 해제 확인!)

D:\Portable\NodeJS\Portable\Data\aa00\node_server\expressTest\app.js (aa00) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

▼ aa00
 ▼ node_server
 ▼ expressTest
 ► node_modules
 app.js
 package.json
 ▼ FileUpload
 file_server.js
 ▼ HTTPServer
 index.js
 ▼ TCPServer
 client.js
 server.js
 wk03_code.txt
 ► start

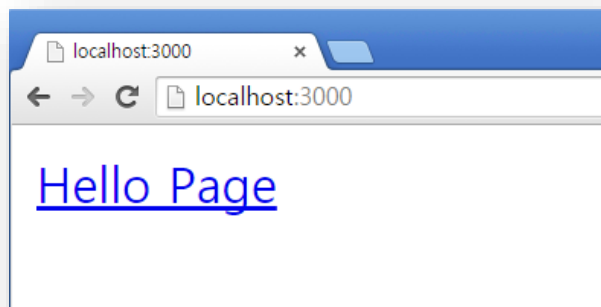
```
1 // app.js, expres cerver
2 var express = require('express');
3 var app = express();
4 var port = 3000;
5
6 app.get('/', function(req, res) {
7   res.send('<a href="/hello">Hello Page</a>');
8 });
9
10 app.get('/hello', function(req, res) {
11   res.send('Hello HSC (하소연)');
12 });
13
14 var server = app.listen(port, function() {
15   console.log('Listening on port %d', server.address().port);
16 });
17
```

Listening on port 3000

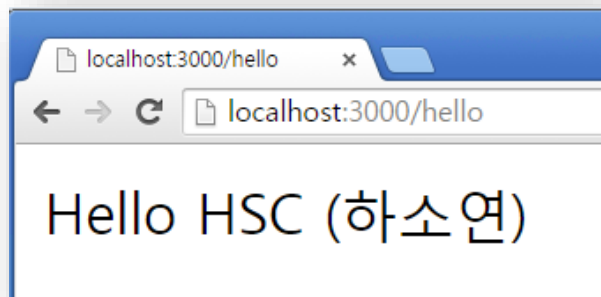


7.1.7 Express server test: test server **routing**

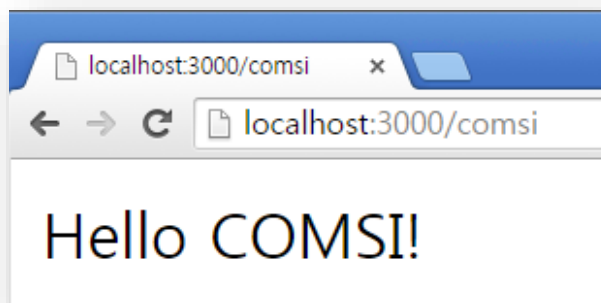
localhost:3000



localhost:3000/hello



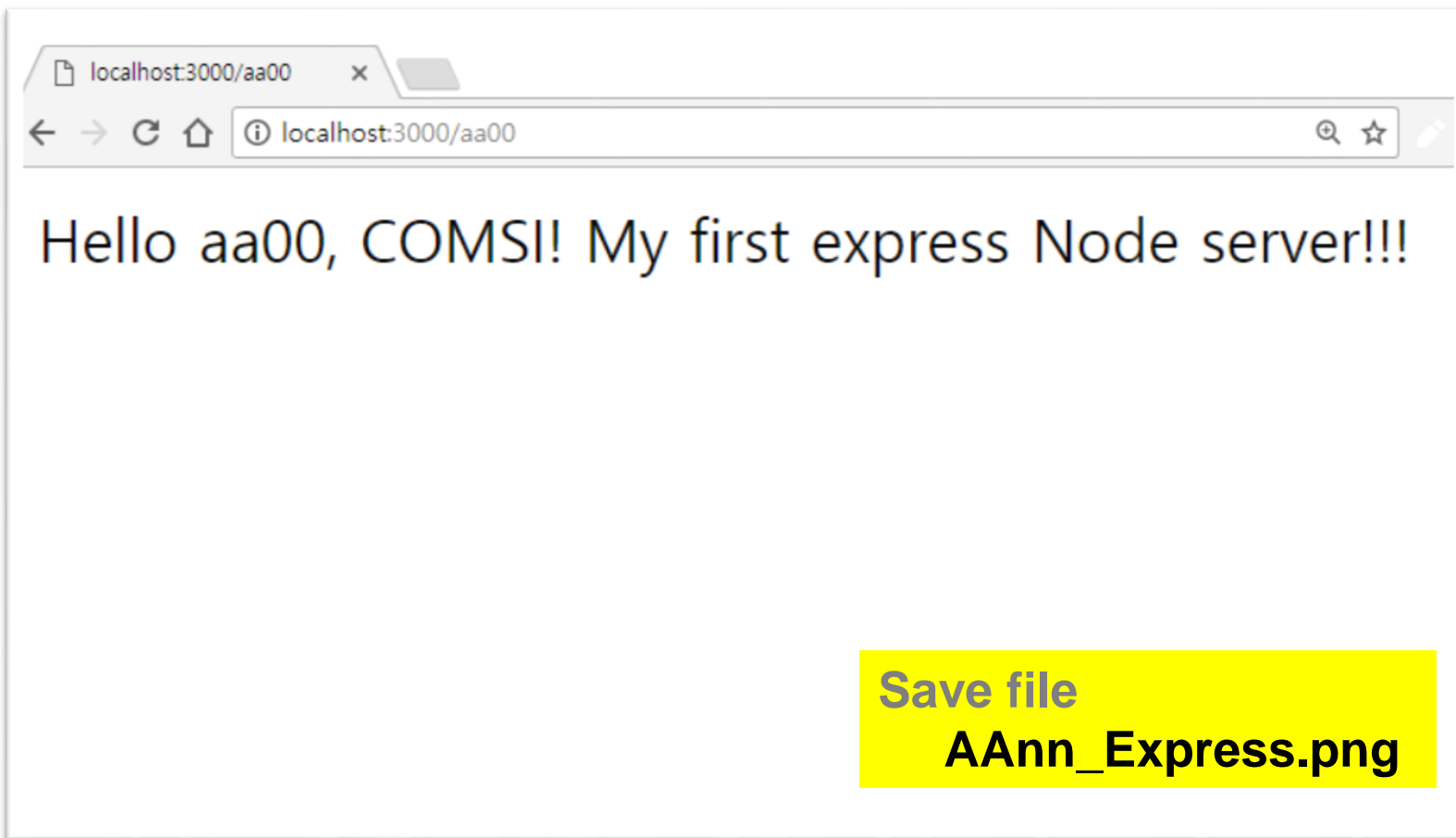
localhost:3000/comsi



^B (실행 전 Tools > Cancel Build 해제 확인!)



7.1.8 Express server test: run `app.js`



Routing: 라우팅은 애플리케이션 엔드 포인트(**URI**)의 정의, 그리고 **URI**가 클라이언트 요청에 응답하는 방식



7.2.1 Full Express App using generator

Step 1 : make folder 'express'

Step 2 : go to the folder

Step 3 : `npm install -g express-generator`

Step 4 : `express`

Step 5 : `npm install`

Step 6 : `node ./bin/www`

Step 6 : <http://localhost:3000>



7.2.2 Install Express-generator

1. **Make folder express**
2. **Go to the folder, “express”**
3. **npm install -g express-generator**
 - **express**

```
D:\Portable\NodeJSPortable\Data\aa00\node_server\expressTest 디렉터리
```

```
2017-09-13 오전 08:46 <DIR> .  
2017-09-13 오전 08:46 <DIR> ..  
2017-09-13 오전 09:04      580 app.js  
2017-09-13 오전 08:23    <DIR> node_modules  
2017-09-13 오전 08:23     278 package.json  
                2개 파일          858 바이트  
                3개 디렉터리   925,266,681,856 바이트 남음
```

```
D:\Portable\NodeJSPortable\Data\aa00\node_server\expressTest>cd ../express
```

```
D:\Portable\NodeJSPortable\Data\aa00\node_server\express>npm install -g express-generator
```

```
loadRequestedDeps → fetch ? ??????????????????????????????????????????????????  
loadRequestedDeps → fetch ? ??????????????????????????????????????????????????  
loadRequestedDeps → after ? ??????????????????????????????????????????????????  
loadDep:sorted-object → r ? ??????????????????????????????????????????????????  
loadDep:sorted-object → 2 ? ??????????????????????????????????????????????????  
loadDep:sorted-object → r ? ??????????????????????????????????????????????????  
loadDep:sorted-object → n ? ??????????????????????????????????????????????????  
loadDep:sorted-object → f ? ??????????????????????????????????????????????????  
loadDep:sorted-object → f ? ??????????????????????????????????????????????????  
loadDep:sorted-object → a ? ??????????????????????????????????????????????????  
loadDep:sorted-object → a ? ??????????????????????????????????????????????????  
loadDep:sorted-object → n ? ??????????????????????????????????????????????????  
loadDep:sorted-object → a ? ??????????????????????????????????????????????????  
loadDep:minimist → reques ? ??????????????????????????????????????????????????  
loadDep:minimist → 200 ? ??????????????????????????????????????????????????  
loadDep:minimist → fetch ? ??????????????????????????????????????????????????  
loadDep:minimist → fetch ? ??????????????????????????????????????????????????  
extract → gunzTarPerm ? ??????????????????????????????????????????????????  
extract:commander → gunzI ? ??????????????????????????????????????????????????  
extract:commander → gunzI ? ??????????????????????????????????????????????????  
extract:mkdirp → gentlyRm ? ??????????????????????????????????????????????????
```

```
D:\Portable\NodeJSPortable\Data\express -> D:\Portable\NodeJSPortable\Data\node_modules\ex  
press-generator\bin\express-cli.js  
D:\Portable\NodeJSPortable\Data
```

```
├── express-generator@4.15.0  
│   ├── commander@2.9.0  
│   │   └── graceful-readlink@1.0.1  
│   ├── ejs@2.5.6  
│   ├── mkdirp@0.5.1  
│   │   └── minimist@0.0.8  
│   └── sorted-object@2.0.1
```



7.2.3 Run Express-generator : express

```
NodeJS
D:\Portable\NodeJSPortable\Data\aa00\node_server\express>express

warning: the default view engine will not be jade in future releases
warning: use '--view=jade' or '--help' for additional options

create : .
create : ./package.json
create : ./app.js
create : ./routes
create : ./routes/index.js
create : ./routes/users.js
create : ./public
create : ./views
create : ./views/index.jade
create : ./views/layout.jade
create : ./views/error.jade
create : ./bin
create : ./bin/www
create : ./public/javascripts
create : ./public/stylesheets
create : ./public/stylesheets/style.css

install dependencies:
> cd . && npm install

run the app:
> SET DEBUG=express:* & npm start

create : ./public/images

D:\Portable\NodeJSPortable\Data\aa00\node_server\express>
```

FOLDERS

```
▼ aa00
  ▼ node_server
    ▼ express
      ► bin
      ► public
      ► routes
      ► views
      ► app.js
      ► package.json
```



7.2.4 package.json

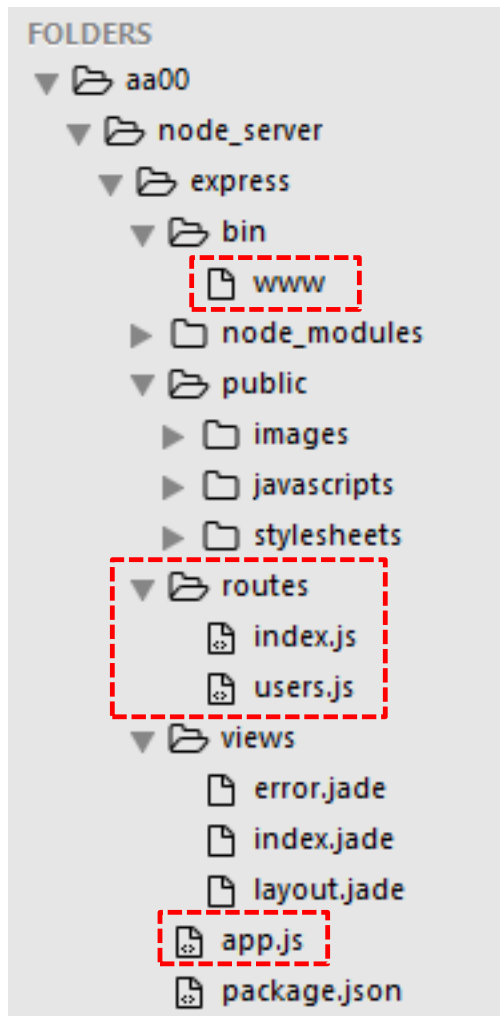
```
1 {
2   "name": "express",
3   "version": "0.0.0",
4   "private": true,
5   "scripts": {
6     "start": "node ./bin/www"
7   },
8   "dependencies": {
9     "body-parser": "~1.17.1",
10    "cookie-parser": "~1.4.3",
11    "debug": "~2.6.3",
12    "express": "~4.15.2",
13    "jade": "~1.11.0",
14    "morgan": "~1.8.1",
15    "serve-favicon": "~2.4.2"
16  }
17 }
18
```

1. Install node modules that is defined in package.json

2. npm install



7.2.5 install node modules



← jade view(default)



7.2.6 run bin/www

```
npm
D:\Portable\NodeJSPortable\Data\aa00\node_server\express>node ./bin/www
Listening on port 3000
^C
D:\Portable\NodeJSPortable\Data\aa00\node_server\express>npm start

> express@0.0.0 start D:\Portable\NodeJSPortable\Data\aa00\node_server\express
> node ./bin/www

Listening on port 3000
█
```

1. Install node modules that is defined in package.json
2. npm install
3. Modules were installed in node_modules subfolder.

4. Run node express web

node ./bin/www

or

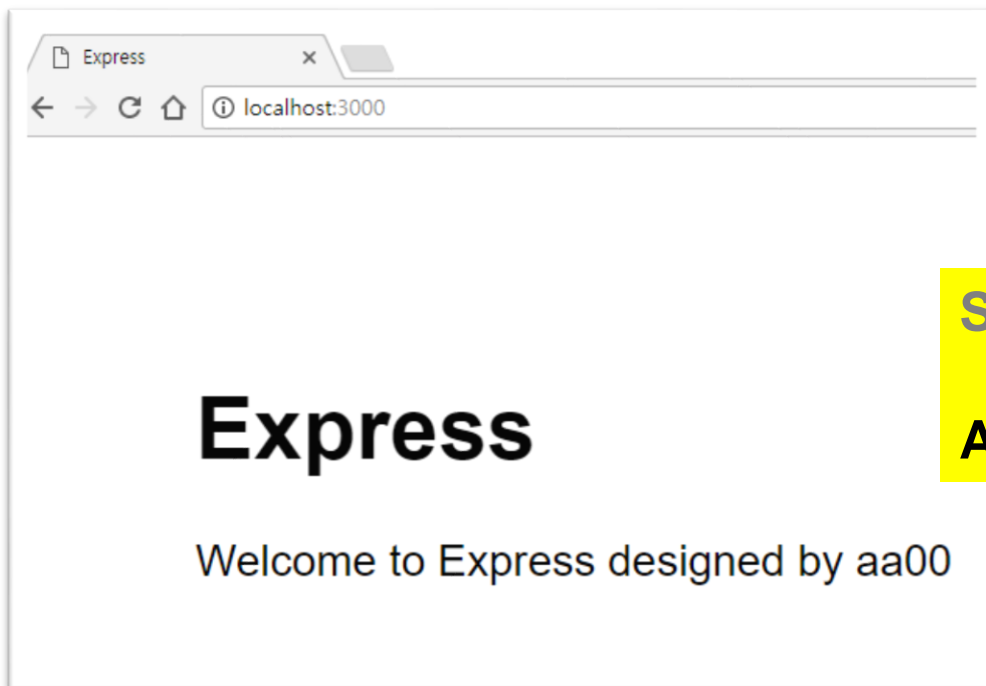
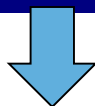
npm start



7.2.7 test app

node ./bin/www

```
D:\Portable\NodeJSPortable\Data\aa00\node_server\express>node ./bin/www
Listening on port 3000
GET / 200 304.727 ms - 187
GET /stylesheets/style.css 304 1.898 ms - -
GET / 304 18.247 ms - -
GET /stylesheets/style.css 304 0.640 ms - -
GET / 304 22.868 ms - -
GET /stylesheets/style.css 304 0.658 ms - -
```



Save file

AAnn_Express_App.png



[Practice]

◆ [wk02]

- NodeJS Server
- Save your results
- File name : AAnn_Rpt02.zip

◆ [Target of this week]

- Check examples of NodeJS server
- Save your outcomes and compress 5 figures

제출파일명 : **AAnn_Rpt02.zip**

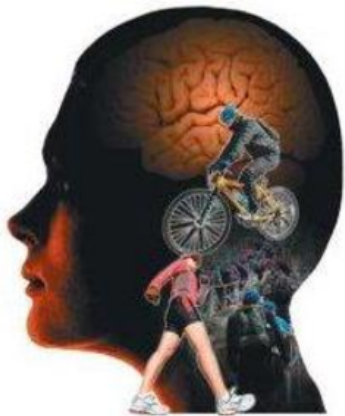
- 압축할 파일들

- ① **AAnn_HTTP.png**
- ② **AAnn_TCP.png**
- ③ **AAnn_Upload.png**
- ④ **AAnn_Express.png**
- ⑤ **AAnn_Express_App.png**

email: chaos21c@gmail.com

● References & good sites

- ✓ <http://www.nodejs.org/ko> Node.js
- ✓ <http://www.arduino.cc> Arduino Homepage
- ✓ <http://www.w3schools.com> By w3schools
- ✓ <http://www.github.com> GitHub
- ✓ <http://www.google.com> Googling

[Features](#)[Business](#)[Explore](#)[Marketplace](#)[Pricing](#)[Sign in](#) or [Sign up](#)

Redwoods Yi

Redwoods

Block or report user

📍 GimHae, Republic of Korea

[Overview](#)[Repositories](#) 5[Stars](#) 2[Followers](#) 0[Following](#) 0

Pinned repositories

dht22-iot-project

lot project to monitor data streaming from DHT22 wired at Arduino.

● HTML

Lec

All lectures by Redwoods in Inje University

arduino-nodejs-plotly-streaming

This repo introduces a simple and efficient way to plot the streaming data from Arduino with Easy Pulse ppg sensor or DHT11 sensor.

● HTML

hw-coding

Resource for lecture of Hardware Programming (2017, Inje university)

● Arduino