**Node.js**

목차

[개요 3](#_Toc482542020)

[NPM (Node Packaged Modules) 3](#_Toc482542021)

[Node Supervisor 3](#_Toc482542022)

[객체 3](#_Toc482542023)

[Process 객체 3](#_Toc482542024)

[Exports 객체 4](#_Toc482542025)

[console 객체 5](#_Toc482542026)

[모듈 5](#_Toc482542027)

[url 모듈 5](#_Toc482542028)

[Query String 모듈 6](#_Toc482542029)

[Util 모듈 7](#_Toc482542030)

[crypto 모듈 8](#_Toc482542031)

[File System 모듈 8](#_Toc482542032)

[http 모듈 9](#_Toc482542033)

[callback 함수 10](#_Toc482542034)

[동기와 비동기 (Synchronous and Asynchronous) 10](#_Toc482542035)

[Express 11](#_Toc482542036)

[// example hello world 11](#_Toc482542037)

[express 정적 파일 제공 12](#_Toc482542038)

[// example static file 12](#_Toc482542039)

[express 동적 파일 제공 13](#_Toc482542040)

[Template & Jade 14](#_Toc482542041)

[URL을 이용한 정보 전달 14](#_Toc482542042)

# 개요

JavaScript 기반, 단일 Thread, V8 JavaScript 엔진, 비동기 방식, 빠른 속도

JavaScript와 연동하여 FS, HTTP, OS (주) 기능을 제공

NPM (Node Packaged Modules)

= Linux의 apt 패키지와 비슷함, 모듈 다운로드 패키지 매니저

Node Supervisor

* 소스 수정 발생시 자동으로 재시동. 꿀
* npm install supervisor -g
* supervisor filename.js
* <https://www.npmjs.com/package/supervisor>

# 객체

Process 객체

= JavaScript에 없는 객체. 프로그램과 관련된 정보를 나타냄

* process.version = Node.js의 버전 보기
* process.versions = Node.js와 종속된 프로그램 버전
* process.argv = 실행중인 매개 변수 출력 ex ) 경로/파일명.exe argv1 argv2
* process.env = 컴퓨터 환경과 관련된 정보
* process.arch = 프로세서의 아키텍처
* process.platform = 플랫폼
* process.exit() = 프로그램 종료
* process.memoryUsage() = 메모리 사용 정보 리턴
* process.uptime() = 프로그램 실행시간 리턴
* Ex ) process.argv

process.argv.forEach( function (item, index) {

console.log(index + “ : ” + typeof (item) + “ : ”, item);

if( item == ‘--exit’)

{

console.log(‘test’);

var exitTime = Number(process.argv[index + 1]);

setTimeout(

function ()

{

process.exit();

}

, exitTime);

}

});

cf ) setTimeout(func(), time) = time이 지나면 func() 실행

forEach() = 각 argv마다 실행

Exports 객체

= 객체의 속성이나 method를 지정하여 기능을 확장

* Ex) sub.js

exports.multiple = function (a,b) {

return a\*b;

};

Main.js

var module = require(‘./sub.js’);

console.log(module.multiple(2,3));

console 객체

= console 출력 객체

* console.log(‘’); -> printf와 사용방법 동일, 출력 포맷을 설정할 수 있으며 %j (json)도 있음

System.out.println 같이도 쓸 수 있음

console.log(‘\u001b[숫자m’, ‘string’); 으로 글자색 표현가능(숫자 = ANSI)

0 = 초기화, 1 = 색 밝게, 30~37 글자색, 40 ~ 47 배경색

* console.time(‘’), console.timeEnd(‘’);
* Ex) console.time(‘atime’);

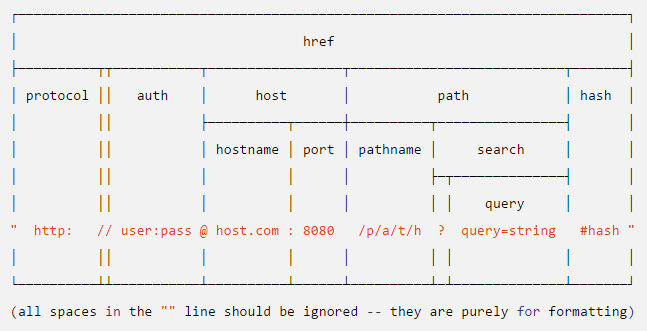
console.timeEnd(‘atime’); // atime: 0.015ms

# 모듈

= require(‘’) 로 모듈 지정 후 사용 가능

ex ) const var url = require(‘url’);

url 모듈



* const var url = require(‘url’);
* 위와 같이 url 모듈을 통해 쪼개져 변환됨
* format(urlObj)

= url 객체를 url 문자열로 변환하여 리턴

* parse(urlStr[, parseQueryString = false[, slashesDenoteHost = false]]);

= urlStr -> 파싱할 주소

= parseQueryString -> 기본값 false, true시 파싱 값 중 query가 객체 형태로 분석됨

= slashesDenoteHost = /를 기점으로 path를 host와 경로로 구분해준다.

http 사용시 영향 x, //host.com/path일 경우 사용

* resolve(from, to)

= 매개변수를 조합하여 url 변환

= ex) url.resolve(‘/1/2/3/4’, ‘5’) // /1/2/3/5

url.resolve(‘/1/2/3/4/’, ‘5’) // /1/2/3/4/5

Query String 모듈

* const var querystring = require(‘querystring’);
* url parse와 겹쳐서 많이 사용 x

Util 모듈

* const var util = require(‘util’);
* var val = util.format(‘%d:%s’, 123, “abc”); // 포맷 지정, 저장

console.log(val);

* util.log(data) // 시간정보 + 내용정보 출력

crypto 모듈

* const var crypto = require(‘crypto’);
* 해시와 암복호화 수행
* 암호화

var cipher = crypto.createCipher(‘aes192’, key);

// 암호 객체 생성, key는 그냥 key값

cipher.update(input, ‘utf8’, ‘base64’);

var output = cipher.final(‘base64’);

* 복호화

var decipher = crypto.createDecipher(‘aes192’, key);

decipher.update(output, ‘base64’, ‘utf8’);

var dec\_output = decipher.final(‘utf8’);

File System 모듈

* const var fs = require(‘fs’);
* 파일 접근자. 동기와 비동기로 나뉜다., 비동기는 callback함수를 추가해야함
* fs.open(‘path’, ‘r or w’, function(error, fd){ // 파일 열기

});

* fs.readFile(‘file.txt’, ‘utf8’, function(error, data){

console.log(data);

});

// fs.readFile(‘file.txt’, {encoding : ‘utf8’}, function(error, data){ 도 가능

* var text = fs.readFileSync(‘file.txt’, ‘utf8’);

console.log(text);

* fs.writeFile(‘file.txt’, ‘불라불라’, ‘utf8’, function(error){

console.log(‘writefile 비동기’);

)};

* fs.writeFileSync(‘file.txt’, ‘동기 불라’, ‘utf8’);

console.log(‘writefilesync 동기’);

* fs.exists(‘경로’, function(exits));
* fs.exitsSync(‘경로’);
* createReadStream, createWriteStream 도 존재
* <http://mudchobo.tistory.com/542>

http 모듈

* 서버 구축
* const http = require(‘http’);

const hostname = ‘127.0.0.1’;

const port = 1337;

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

res.writeHead(200, { ‘Content-Type’ : ‘text/plain’ });

res.end(‘Hello World\n’);

}).listen(port, hostname, () => { // listen이 완료되면 callback 발동

console.log(‘Server running at http://${hostname}:${port}/’);

});

* <http://blog.naver.com/eun033/220963561825>

# callback 함수

ex) a = [3,1,2]; a.sort(); console.log(a); // result = [1,2,3]

[3,2,1]로 출력하고 싶으면

sort() 함수 내에 인풋 값으로 함수를 받음

함수 형식 = function input(val1, val2){return …}; // … 내에 val1-val2(오름) 혹은 val2-val1(내림) (양수 혹은 음수)혹은 0

a = [3,1,2]; function b(val1, val2){return val2-val1}; a.sort(b); console.log(a);

of

a = [3,1,2]; a.sort(function(v1, v2){return v1-v2}); console.log(a);

* 어떤 함수의 인자로 함수(callback 함수)를 넘겨줌

# 동기와 비동기 (Synchronous and Asynchronous)

결국 동기화, 비동기화

file system과 같은 곳에서 동기, 비동기 둘다 지원 but, 비동기를 지향

# Express

Node.js 의 서버 개발 framework, <http://expressjs.com/ko/>

// example hello world

const express = require('express');

const app = express();

const ip = '127.0.0.1';

const port = 1338;

// 127.0.0.1:1338/home으로 들어오면 hello world 출력

app.get('/home', function (req, res){ // 127.0.0.1:1338/home으로 들어옴

res.send('hello world');

});

app.get('/login', function (req, res){ // 127.0.0.1:1338/login

res.send('please login');

});

app.listen(port, ip,function(){

console.log("hello world server listening on loopback:1338");

});

* get()을 router라고 표현, get()이 하는 역할을 routing이라고 표현

express 정적 파일 제공

파일에 html을 넣어놓으면 정적으로 해당 url 접근시 html 표현

파일 내용 수정 시 바로 반영됨

// example static file

const express = require('express');

const app = express();

const ip = '127.0.0.1';

const port = '1337';

app.use(express.static('public')); // 정적인 파일이 위치할 directory 지정

// 127.0.0.1:1337/sinjjang.png

app.get('/home', function (req, res){

res.send('hello static file\'s home');

});

app.get('/image', function (req, res){ // 127.0.0.1:1337/image

res.send('Hello sinjjang, <img src = "/sinjjang.png">');

});

app.listen(port, ip, function(){

console.log('server open in 127.0.0.1:1337');

});

express 동적 파일 제공

변화가 있을 시 코드 수정 후 서버 재시동 필요

html 코드 내에서는 소스코드사용이 불가. for문과 같은걸 사용 x. 그래서 사용

app.get('/dynamic', function (req, res){

var lis = '';

for(var i = 0 ; i < 5 ; ++i){

lis += '<li> coding </li>';

}

var time = Date(); // javascript의 시간 api

var output =

// ''가 아닌 1옆의 '', ${}는 변수라는걸 알려줌

`<!DOCTYPE HTML> // html의 버전을 알려줌 !DOCTYPE은 html5. 항상 위에 있어야함

<html>

<head>

<meta charset = "utf-8">

<title>

dynamic test

</title>

</head>

<body>

hello, dynamic!

<ul>

${lis}

</ul>

${time}

</body>

</html>`

res.send(output);

});

Template & Jade

* jade는 node.js에 맞게 만들어진 template engine 중 하나
* <https://www.npmjs.com/package/jade>
* template엔진 다운 후

app.set('view', './views') // template가 있는 위치를 알려줌. (jade파일이 있는 위치)

app.set('view engin', 'jade'); // template engine load

위와 같이 설정 지정

app.get('/template', function(req, res){

res.render('temp', {time : Date()}); // template엔진을 렌더링. temp라는 template파일을 렌더링을 통해 웹으로 전송

// 두번째 인자로 전달해주고 싶은 객체 전달

});

후에 /views/temp.jade 파일을 렌더링으로 지정하여 웹으로 전송

URL을 이용한 정보 전달

* 사용자의 조작이 있을 때 동작이 일어나게 해줌
* 사용자로부터 정보를 입력 받음
* Query String 사용
* 127.0.0.1:1337/topic?id=2&name=hi

app.get('/topic', function(req,res){ // /topic/:id 면 /topic/2로 입력하면 id = 2 가 됨

// res.send(req.query.id+ ',' + req.query.name); // req.query.id == 사용자가 request한query의 id값

var topics = [ // 배열 선언

'javascript is ...',

'nodejs is ...',

'express is ...'

];

// 링크생성 , 표현되는 정보

var links = `

<a href = "/topic?id=0">JavaScript</a><br>

<a href = "/topic?id=1">Node.js</a><br>

<a href = "/topic?id=2">Express</a><br><br>

${topics[req.query.id]}

`

res.send(links);

});

Post 방식의 정보 전달

* 정보를 서버로 전달
* html의 form 구문의 get방식은 url을 자동생성 후 이동해줌
* doctype html

html

head

meta(charset='utf-8')

body

form(action='/form\_receiver' method='get') // method=’post’도 있음. post사용하면 url이 바뀌는게 아니고 서버로 전송만됨

p

input(type='text' name='title')

p

textarea(name='description')

p

input(type='submit')

* 원래 post방식으로 입력받으려면 req.body.title 이런 식으로 받아야 하는데 body를 지원을 안해서 bodyparser를 사용하여 req.body.title로 받음

const bodyParser = require('body-parser');

app.use(bodyParser.urlencoded({ extended : false})); // 이 앱을 통하는 모든 정보는 bodyparser을 통해 들어옴

* query방식은 url상에 모든 정보를 포함할 수 있어 편함
* post방식은 url에서 어떤 정보를 숨기고 싶을 때 사용 또는 url이 너무 길어질 때