**Node.js**

목차

[개요 2](#_Toc482115323)

[NPM (Node Packaged Modules) 2](#_Toc482115324)

[Process 객체 2](#_Toc482115325)

[Exports 객체 3](#_Toc482115326)

[console 객체 4](#_Toc482115327)

# 개요

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

NPM (Node Packaged Modules)

= Linux의 apt 패키지와 비슷함, 모듈 다운로드 패키지 매니저

# 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’);

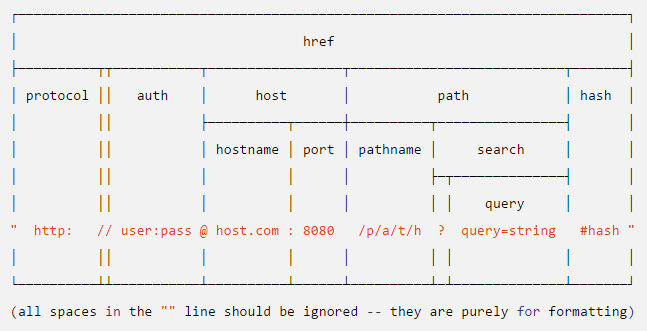
console.timeEnd(‘atime’); // atime: 0.015ms

# 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);

});

* 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, () => {

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

});

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