

# Part 2 Node.js 기본

## 4장 기본 내장 모듈

---

2017.09.27 작성

---

멋사 자바스크립트 수원팀

---

서유진

---

# 목차

4.1 os 모듈

---

4.2 url 모듈

---

4.3 Query String 모듈

---

4.4 util 모듈

---

4.5 crypto 모듈

---

4.6 File System

---

4.6.1 파일 읽기

4.6.2 파일 쓰기

4.6.3 예외 처리

# OS 모듈

모듈의 기본적인 사용법을 익히기에 가장 적당한 모듈

Os 모듈의 메서드

```
JS 20170927.js
1 var os = require('os');
2
3 console.log(os.hostname()); //seoyujin-ui-MacBook-Pro.local
4 console.log(os.type()); //Darwin
5 console.log(os.platform()); //darwin
6 console.log(os.arch()); //x64
7 console.log(os.release()); //16.7.0
8 console.log(os.uptime()); //70881
9 console.log(os.loadavg()); //[ 1.20849609375, 1.4345703125, 1.55517578125 ]
10 console.log(os.totalmem()); //8589934592
11 console.log(os.freemem()); //234942464
12 console.log(os.cpus());
13 console.log(os.networkInterfaces());
14
```

운영체제의 호스트 이름, 이름, 플랫폼, 아키텍처, 버전, 운영체제가 실행된 시간, 로드 에버리지 정보, 시스템의 총 메모리, cpu의 정보를 담은 객체, 네트워크 인터페이스 정보

# url 모듈

- URL
  - URL Strings and URL Objects
    - `urlObject.href`
    - `urlObject.protocol`
    - `urlObject.slashes`
    - `urlObject.host`
    - `urlObject.auth`
    - `urlObject.hostname`
    - `urlObject.port`
    - `urlObject.pathname`
    - `urlObject.search`
    - `urlObject.path`
    - `urlObject.query`
    - `urlObject.hash`
  - `url.format(urlObject)`
  - `url.parse(urlString[, parseQueryString[, slashesDenoteHost]])`
  - `url.resolve(from, to)`
  - Escaped Characters

# url 모듈

url 정보를 객체로 가져와 분석하거나(parse) url 객체를 문자열로 바꿔주는 기능(format, resolve)을 수행

```
15
16 var url = require('url');|
17 var parsedObject = url.parse('http://www.google.com');
18 console.log(parsedObject); //url 객체 정보
19 console.log(format(parsedObject)); //url 객체를 문자열로 출력
20
```

```
[seoyujin-ui-MacBook-Pro:node getitbeauty$ node 20170927.js
```

```
Url {
  protocol: 'http:',
  slashes: true,
  auth: null,
  host: 'www.google.com',
  port: null,
  hostname: 'www.google.com',
  hash: null,
  search: null,
  query: null,
  pathname: '/',
  path: '/',
  href: 'http://www.google.com/' }
```

Cliquez pour ajouter des comm

# url 모듈

url 문자열(urlStr)을 url 객체로 변환하여 리턴. parseQueryString과 slashesDenoteHost는 **false**가 기본값

## parseQueryString

- true : url 객체의 query 속성을 객체 형식으로 가져옵니다.(querystring 모듈을 사용합니다.)
- false : url 객체의 query 속성을 문자열 형식으로 가져옵니다.

## slashesDenoteHost

- true : urlStr이 '//foo/bar' 인 경우 foo는 host, /bar는 path로 인식합니다.
- false : urlStr이 '//foo/bar' 인 경우 //foo/bar 전체를 path로 인식하고 host는 null 입니다.

# Query String 모듈

- Query String
  - `querystring.escape(str)`
  - `querystring.parse(str[, sep[, eq[, options]]])`
  - `querystring.stringify(obj[, sep[, eq[, options]]])`
  - `querystring.unescape(str)`

# Query String 모듈

## 매개 변수 [sep], [eq]

매개 변수를 먼저 살펴 보겠습니다. sep는 separator로 기본값은 '&' 입니다. 쿼리가 여러 개가 있을 때 쿼리와 쿼리 간의 구분자 역할을 합니다. eq는 assignment 역할을 하는 기호로 기본값은 '=' 입니다.

```
//모듈 추출
var url = require('url');
var querystring = require('querystring');
//모듈 사용
var parsedObject=url.parse('http://www.megabox.co.kr/?show=detail&rtnShowMovieCode=012360');
console.log(querystring.parse(parsedObject.query));
```

```
seoyujin-ui-MacBook-Pro:node getitbeauty$ node 20170927.js
{ show: 'detail', rtnShowMovieCode: '012360' }
```



# Query String 모듈

## `querystring.parse()`

: 쿼리 문자열을 쿼리 객체로 바꿔주는 역할

`options`는 객체형식

`maxKeys` 속성을 통해 최대 개수를 제한가능

(기본값은 1000이며 0으로 설정할 경우 그 제한을 없앨 수 있음)

## `querystring.stringify()`

: 쿼리 객체를 쿼리 문자열로 바꿔주는 역할

# Query String 모듈

url.parse() 메서드의 두 번째 매개변수를 활용해서 같은 결과값 얻기

```
22 var url = require('url');
23 var querystring = require('querystring');
24 // 모듈 사용
25 var parsedObject=url.parse('http://www.megabox.co.kr/?show=detail&rtnShowMovieCode=012360');
26 console.log(querystring.parse(parsedObject.query));
27
28 console.log(url.parse('http://www.megabox.co.kr/?show=detail&rtnShowMovieCode=012360'));
```

```
[seoyujin-ui-MacBook-Pro:node getitbeauty$ node 20170927.js
{ show: 'detail', rtnShowMovieCode: '012360' }
Url {
  protocol: 'http:',
  slashes: true,
  auth: null,
  host: 'www.megabox.co.kr',
  port: null,
  hostname: 'www.megabox.co.kr',
  hash: null,
  search: '?show=detail&rtnShowMovieCode=012360',
  query: { show: 'detail', rtnShowMovieCode: '012360' },
  pathname: '/',
  path: '/?show=detail&rtnShowMovieCode=012360',
  href: 'http://www.megabox.co.kr/?show=detail&rtnShowMovieCode=012360' }
```

# util 모듈

node.js의 보조적인 기능을 모아둔 모듈

## Util

- `util.debuglog(section)`
- `util.deprecate(function, string)`
- `util.format(format[, ...args])`
- `util.inherits(constructor, superConstructor)`
- `util.inspect(object[, options])`
  - Customizing `util.inspect` colors
  - Custom inspection functions on Objects
  - `util.inspect.defaultOptions`
  - `util.inspect.custom`

### deprecated



형용사

1. [신조어]중요도가 떨어져 더 이상 사용되지 않고 앞으로는 사라지게 될 (컴퓨터 시스템 기능 등)

## Deprecated APIs

- `util.debug(string)` deprecated
- `util.error([...strings])` deprecated
- `util.isArray(object)` deprecated
- `util.isBoolean(object)` deprecated
- `util.isBuffer(object)` deprecated
- `util.isDate(object)` deprecated
- `util.isError(object)` deprecated
- `util.isFunction(object)` deprecated
- `util.isNull(object)` deprecated
- `util.isNullOrUndefined(object)` deprecated
- `util.isNumber(object)` deprecated
- `util.isObject(object)` deprecated
- `util.isPrimitive(object)` deprecated
- `util.isRegExp(object)` deprecated
- `util.isString(object)` deprecated
- `util.isSymbol(object)` deprecated
- `util.isUndefined(object)` deprecated
- `util.log(string)` deprecated
- `util.print([...strings])` deprecated
- `util.puts([...strings])` deprecated
- `util._extend(target, source)` deprecated

# util 모듈

node.js의 보조적인 기능을 모아둔 모듈

```
var util = require('util');
var data = util.format('%d + %d = %d', 7, 14, 7+14);
console.log(data);
```

화면에 출력을 하는 console.log() 와 유사한 기능을 함  
차이점은 util.format은 문자열을 반환함

- %s - String.
- %d - Number (both integer and float).
- %j - JSON.
- % - single percent sign ('%'). This does not consume an argument.

# crypto 모듈

해시 생성과 암호화를 수행하는 모듈



사용자가 제출한 내용을 서버는 해시화해서 저장.  
서버에 원본이 남지 않음  
사용자가 서버에 제출할 때마다 해시화해서 일치하는지 비교

**눈사태 효과(avalanche effect) :**

원본 문자열이 조금이라도 다르면 해시의 형태가 완전히 달라지는 것

**암호화와 인증(해시)의 차이점 : 복호화를 할 수 있는지의 여부**

<http://brownbears.tistory.com/73>

# crypto 모듈

암호화 알고리즘에 사용되는 용어들

Plaintext (cleartext; 평문): 전달해야 할 내용

**ciphertext (암호문):** 암호화한 내용

encryption (**encipher; 암호화**): 어떤 내용을 위장하는 것

decryption (**decipher; 복호화**): 암호문을 평문으로 복구하는 것

cryptographic algorithm (**cipher, 암호화 알고리즘**): 암호화와 복호화를 위해 사용하는 수학 함수.

**key(키):** 암호화 알고리즘의 파라미터.

# crypto 모듈

## 대표적인 해시 알고리즘

: MD5(Message Digest)  
SHA(Secure Hash Algorithm)

## SHA256 알고리즘

[https://seed.kisa.or.kr/iwt/ko/bbs/EgovReferenceDetail.do?bbsId=BBSMSTR\\_000000000002&nttlId=79](https://seed.kisa.or.kr/iwt/ko/bbs/EgovReferenceDetail.do?bbsId=BBSMSTR_000000000002&nttlId=79)

## Message Authentication Code

: 원문 메시지에 보안키를 추가하여 verification data를 생성  
제3자가 메시지를 변경하면 수신자가 알아챌 수 있음  
인증과 무결성을 동시에 제공  
암호화에 비해 연산이 빠르다

## 인코딩 Base64 vs Hexadecimal

<http://egloos.zum.com/dialup/v/170497>

Base64 : A~Z a~z 0~9 + / (총 64개)

Hexadecimal : 0~9 A~F (총 16개)



# crypto 모듈

## 해시 생성

```
36  //// * crypto 모듈 예제 1 *
37  var crypto = require('crypto');
38
39  var shasum = crypto.createHash('sha256');
40  shasum.update('crypto_hash');
41  var output = shasum.digest('hex');
42
43  console.log('crypto_hash: ', output);
44
```

- `crypto.createHash(algorithm)`

`crypto.createHash`

[https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto\\_crypto\\_createhash\\_algorithm](https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto_crypto_createhash_algorithm)

- Class: Hash

- `hash.digest([encoding])`

- `hash.update(data[, input_encoding])`

`.update`

[https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto\\_hash\\_update\\_data\\_input\\_encoding](https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto_hash_update_data_input_encoding)

`.digest`

[https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto\\_hash\\_digest\\_encoding](https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto_hash_digest_encoding)



# crypto 모듈

## 해시 생성하기

---

- crypto.createHash(algorithm)
- Class: Hash
  - hash.update(data, [input\_encoding])
  - hash.digest([encoding])

Crypto.createHash() 메소드를 호출하면 매개 변수로 전달한 알고리즘에 해당하는 Hash 클래스가 반환됨

Hash 클래스의 update() 메소드를 통해 data를 해싱하고

digest() 메소드로 encoding 방식(hex)에 따라서 결과 값을 가져올 수 있다.

# crypto 모듈

## 암호화 수행

```
33 var crypto = require('crypto');
34
35 var key = '여기를 나만 아는 키로 적어놓는다면!'; //비밀키
36 var input = 'PASSWORD';
37
38 // 암호화
39 var cipher = crypto.createCipher('aes192', key);
40 cipher.update(input, 'utf8', 'base64');
41 var cipheredOutput = cipher.final('base64');
42
43 // 암호화 해제
44 var decipher = crypto.createDecipher('aes192', key);
45 decipher.update(cipheredOutput, 'base64', 'utf8');
46 var decipheredOutput = decipher.final('utf8');
47
48 // 출력
49 console.log('원래 문자열:' + input );
50 console.log('암호화 :' + cipheredOutput);
51 console.log('암호화 해제 : ' + decipheredOutput);
52
```

```
원래 문자열 :PASSWORD
암호화 :FD1DAeKf5Jh2ER574HktKw==
암호화 해제 : PASSWORD
```

`crypto.createCipher`

[https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto\\_crypto\\_createcipher\\_algorithm\\_password](https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto_crypto_createcipher_algorithm_password)

`cipher.update`

[https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto\\_hash\\_update\\_data\\_input\\_encoding](https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto_hash_update_data_input_encoding)

`cipher.final`

[https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto\\_cipher\\_final\\_output\\_encoding](https://nodejs.org/dist/latest-v6.x/docs/api/crypto.html#crypto_cipher_final_output_encoding)

# crypto 모듈

## 암호화 및 복호화

---

- crypto.createCipher(algorithm, password)
- crypto.createCipheriv(algorithm, key, iv)
- Class: Cipher
  - cipher.update(data, [input\_encoding], [output\_encoding])
  - cipher.final([output\_encoding])
  - cipher.setAutoPadding(auto\_padding=true)
- crypto.createDecipher(algorithm, password)
- crypto.createDecipheriv(algorithm, key, iv)
- Class: Decipher
  - decipher.update(data, [input\_encoding], [output\_encoding])
  - decipher.final([output\_encoding])
  - decipher.setAutoPadding(auto\_padding=true)

crypto.createCipher( algorithm: 암호화에 사용할 알고리즘 이름, 암호화 및 복호화에 사용할 키 )

cipher.update( 암호화하고 싶은 데이터 )

cipher.final() : 암호화된 값 얻기

Crypto.createDechipher()

# File System 모듈

## 4.6.1 파일 읽기

- `fs.readFile(filename, [options], callback)`

`filename`의 파일을 `[options]`의 방식으로 읽은 후 `callback`으로 전달된 함수를 호출합니다. (비동기적)

- `fs.readFileSync(filename, [options])`

`filename`의 파일을 `[options]`의 방식으로 읽은 후 문자열을 반환합니다. (동기적)

**Sync**가 붙은 것은 동기적 읽기, 붙지 않은 것은 비동기적 읽기입니다. 파일을 읽는데 시간이 오래 걸릴 수도 있습니다. 동기적 읽기로 읽게 되면 파일을 읽으면서 다른 작업을 동시에 할 수 없습니다. 하지만 비동기적으로 읽으면 파일을 읽으면서 다른 작업도 동시에 수행할 수 있고 파일을 다 읽으면 매개변수 `callback`으로 전달한 함수가 호출됩니다.

`[options]`에는 보통 인코딩 방식이 오게 되며 웹에서는 `utf8`을 주로 사용합니다.

파일을 읽어야 하므로 `text.txt`라는 이름으로 텍스트 파일을 하나 준비합니다.

# File System 모듈

## 4.6.1 파일 읽기 Textfile.txt 파일을 UTF-8 방식으로 읽고 출력하기

```
65
66  //// File system - readFileSync()
67  var fs = require('fs');
68
69  var text = fs.readFileSync('textfile.txt', 'utf8');
70  console.log(text);
71
```

```
72  //// File system - readFile()
73  var fs = require('fs');
74  fs.readFile('textfile.txt', 'utf8', function (error, data){
75      console.log(data);
76  });
77
```

# File System 모듈

## 4.6.2 파일 쓰기

- `fs.writeFile(filename, data, [options], callback)`

`filename`의 파일에 `[options]`의 방식으로 `data` 내용을 쓴 후 `callback` 함수를 호출합니다. (비동기적)

- `fs.writeFileSync(filename, data, [options])`

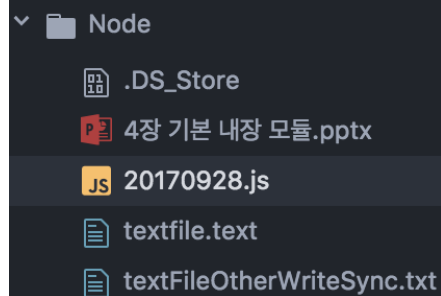
`filename`의 파일에 `[options]`의 방식으로 `data` 내용을 씁니다. (동기적)

사용법이나 동기적/비동기적 차이는 파일 읽기 메소드와 비슷합니다.

# File System 모듈

## 4.6.2 파일 쓰기

```
78  /// File system - writeFileSync() & writeFile()
79  var fs = require('fs');
80  var data = 'hello world!';
81
82  fs.writeFile('textFileOtherWriteSync.txt', data, 'utf8', function (error){
83    console.log('WRITE FILE ASYNC COMPLETE');
84  });
85
86  fs.writeFileSync('textFileOtherWriteSync.txt', data, 'utf8');
87  console.log('WRITE FILE SYNC COMPLETE');
```



Node

- .DS\_Store
- 4장 기본 내장 모듈.pptx
- 20170928.js
- textfile.text
- textFileOtherWriteSync.txt

```
[seoyujin-ui-MacBook-Pro:node getitbeauty$ node 20170928.js
WRITE FILE SYNC COMPLETE
WRITE FILE ASYNC COMPLETE
[seoyujin-ui-MacBook-Pro:node getitbeauty$ ls
20170928.js
4장 기본 내장 모듈.pptx
textFileOtherWriteSync.txt
textfile.text
```

# File System 모듈

## 4.6.3 예외처리

파일 입출력은 매우 다양한 원인으로 예외가 발생할 수 있습니다. 권한이 없다거나 존재하지 않는 파일을 읽는다거나 심지어 하드디스크 용량을 초과할 수도 있습니다. 동기적인 방식과 비동기적인 방식에서 예외를 처리하는 방법이 조금 다르므로 따로 알아보겠습니다.

### 동기적 방식의 예외처리

동기적 방식에서는 자바스크립트의 일반적인 예외처리 방식인 `try ~ catch` 구문으로 처리합니다.

위와 같이 예외가 발생하면 그 예외와 관련된 객체를 `throw`하게 됩니다.

### 비동기적 방식의 예외처리

비동기적 방식에서 예외가 발생하면 `callback` 함수의 매개변수 `err`에 전달되므로 따로 `try ~ catch` 구문을 사용할 필요가 없습니다.

만약 예외가 발생하지 않았다면 `err`에 아무 값도 들어가지 않아서 `if`문이 `false`가 됩니다.



# File System 모듈

## 4.6.3 예외처리

```
89  //// File system - 동기 처리를 하는 메서드의 예외 처리
90  var fs = require('fs');
91  //파일 읽기
92  try {
93      var data = fs.readFileSync('textfile.txt', 'utf8');
94      console.log(data);
95  } catch(e) {
96      console.log(e);
97  }
98  }
99  //파일 쓰기
100 try{
101     fs.writeFileSync('textfile.txt', 'hello world', 'utf8');
102     console.log('FILE WRITE COMPLETE');
103 } catch(e){
104     console.log(e);
105 }
106 }
```

# File System 모듈

## 4.6.3 예외처리

```
108  ///// File system - 비동기 처리를 하는 메서드의 예외 처리
109      var fs = require('fs');
110      // 파일 읽기
111      fs.readFile('textfile.txt', 'utf8', function(error, data){
112          if (error){
113              console.log(error);
114          } else {
115              console.log(data);
116          }
117      })
118      // 파일 쓰기
119      fs.writeFile('textfile.txt', 'hello world', 'utf8', function(error){
120          if (error) {
121              console.log(error);
122          } else {
123              console.log('FILE WRITE COMPELETE');
124          }
125      });
```

감사합니다

---