노드 내장모듈 사용하기 필기노트

2021년 1월 20일 수요일 오후 3:51

1. os
   운영체제의 정보를 담고 있음, 모듈은 require로 가져옴

```
const os = require('os');
console.log('운영체제 정보-------------------------');
console.log('os.arch():',os.arch());
console.log('os.platform():',os.platform());
console.log('os.type():',os.type());
console.log('os.uptime():',os.uptime());
console.log('os.hostname():',os.hostname());
console.log('os.release():',os.release());
console.log('경로---------------------------------');
console.log('os.homedir():',os.homedir());
console.log('os.tmpdir():',os.tmpdir());
console.log('cpu 정보---------------------------');
console.log('os.cpus():',os.cpus());
console.log('os.cpus().length:',os.cpus().length);
console.log('메모리 정보--------------------------');
console.log('os.freemem():',os.freemem());
console.log('os.totalmem():',os.totalmem());
```

```
os.arch(): x64
os.platform(): win32
os.type(): Windows_NT
os.uptime(): 404696
os.hostname(): DESKTOP-IGM1AKI
os.release(): 10.0.19042
경로---------------------------------
os.homedir(): C:\Users\jhn06
os.tmpdir(): C:\Users\jhn06\AppData\Local\Temp
cpu 정보---------------------------
os.cpus(): [
 {
  model: 'AMD Ryzen 5 2600 Six-Core Processor          ',
  speed: 3400,
  times: {
   user: 34646265,
   nice: 0,
   sys: 41187046,
   idle: 122954328,
   irq: 7119359
  }
 },
 {
  model: 'AMD Ryzen 5 2600 Six-Core Processor          ',
 },
]
os.cpus().length: 12
메모리 정보--------------------------
os.freemem(): 10260762624
os.totalmem(): 17126883328
```

나머지 코어의 정보

- os.cpus() : 컴퓨터의 코어 정보를 보여줍니다
  node는 싱글 스레드를 사용하므로 효율적으로 서버를 구현하려면 서버를 코어 개수만큼 돌리면 된다

2. path
   폴더와 파일의 경로를 쉽게 조작하도록 도와주는 모듈
   운영체제 별로 구분자가 다름 (Windows : '₩', POSIX : '/')

```
const path = require('path')
const string = __filename;
console.log('path.sep:',path.sep);
console.log('path.delmiter:',path.delimiter);
console.log("----------------------");
console.log("path.dirname():",path.dirname(string));
console.log("path.extname():",path.extname(string));
console.log("path.basename():",path.basename(string));
console.log("path.basename - extname:",path.basename(string, path.extname(strin
g)));
console.log("-------------------");
console.log("path.parse()",path.parse(string));
console.log("path.format():",path.format({
   dir: "E:\프로그래밍\Nodejs 교과서\3. 노드 기능",
   name: 'path',
   ext: '.js',
}));
console.log('path.normalize():',path.normalize("E:\프로그래밍\Nodejs 교과서\3. 노
드 기능\path.js"));
console.log("--------------------------");
console.log("path.isAbsolute(C:\\):",path.isAbsolute("C:\\"));
console.log("path.isAbsolute(./home):",path.isAbsolute("./home"));
console.log("--------------------------");
console.log("path.relative():",path.relative("E:\프로그래밍\Nodejs 교과서\3. 노
드 기능\path.js","E:\\"));
console.log("path.join()",path.join(__dirname,"..","..","/users",".","/jhn06"))
;
console.log("path.resolve():",path.resolve(__dirname,"..","users",".","/jhn06")
);
```

```
path.sep: \
path.delmiter: ;
----------------------
path.dirname(): e:\프로그래밍\Nodejs 교과서\3. 노드 기능\os와path
path.extname(): .js
path.basename(): path.js
path.basename - extname: path
-------------------
path.parse() {
 root: 'e:\\',
 dir: 'e:\\프로그래밍\\Nodejs 교과서\\3. 노드 기능\\os와path',
 base: 'path.js',
 ext: '.js',
 name: 'path'
}
path.format(): E:프로그래밍Nodejs 교과서. 노드 기능\path.js
path.normalize(): E:프로그래밍Nodejs 교과서. 노드 기능path.js
--------------------------
path.isAbsolute(C:\): true
path.isAbsolute(./home): false
```

```
--------------------------
path.relative(): ..\..\..
path.join() e:\프로그래밍\Nodejs 교과서\users\jhn06
path.resolve(): e:\jhn06
```

- path.join(경로, ...): 여러 인자를 넣으면 하나의 경로로 합쳐준다. 상대 경로인 ..(부모 디렉터리)와 .(현 위치)도 알아서 처리
- path.resolve(경로, ..): path.join()와 비슷하지만, join은 절대경로 무시, resolve는 절대 경로를 존중하여 앞의 것들이 무시
- path.sep : 경로의 구분자, 윈도우는 ₩, posix는 / 사용
- path.dirname(경로) : 폴더명 추출
- path.extname(경로) : 확장자 추출
- path.basename(경로, 확장자) : 파일의 이름을 확장자 포함하여 보여줌
- path.parse(경로) : 분해
- path.format(객체) : 합침
- path.normalize(경로) : /나 ₩를 실수로 여러 번 사용하였을 경우 정상 경로로 반환
- path.isAbsolute(경로) : 파일의 경로를 절대경로/상대경로인지 True/false로 반환
- path.relative(기준경로, 비교경로):경로를 두개 넣으면 첫 번째 경로에서 두 번째 경로로 가는 방법을 알려줌

3. url
   인터넷 주소를 쉽게 조작하도록 도와주는 모듈
   url처리에 크게 두 가지 방식이 있음 ->기존 노드 방식, WHATWG방식
   URL | Node.js v14.15.4 Documentation
   외우지 말고 옆에 두고 사용하면서 익히는것이 좋음

기존 Node

WHATWG

```
┌─────────────────────────────────────────────────────────────────────────────────────────────┐
│                                            href                                             │
├──────────┬──┬─────────────────────┬────────────────────┬───────────────────────────┬──────┤
│ protocol │  │        auth         │        host        │           path            │ hash │
│          │  │                     ├─────────────┬──────┼─────────────┬─────────────┤      │
│          │  │                     │   hostname  │ port │  pathname   │   search    │      │
│          │  │                     │             │      │             ├┬────────────┤      │
│          │  │                     │             │      │             ││   query    │      │
"  https:   //    user   :   pass   @ sub.example.com : 8080   /p/a/t/h  ?  query=string   #hash "
│          │  │                     │   hostname  │ port │             │             │      │
│          │  │                     ├─────────────┴──────┤             │             │      │
│ protocol │  │ username │ password │        host        │             │             │      │
├──────────┴──┼──────────┴──────────┼────────────────────┤             │             │      │
│   origin    │                     │       origin       │  pathname   │   search    │ hash │
├─────────────┴─────────────────────┴────────────────────┴─────────────┴─────────────┴──────┤
│                                            href                                             │
└─────────────────────────────────────────────────────────────────────────────────────────────┘
(All spaces in the "" line should be ignored. They are purely for formatting.)
```

```
const url = require('url');
const {URL} = url;
const myURL = new URL('http://www.gilbut.co.kr/book/bookList.aspx?sercate1=001001000#anchor');
console.log('new URL():',myURL);
console.log('url.format():',url.format(myURL));
console.log('----------------------');
const parsedUrl = url.parse('http://www.gilbut.co.kr/book/bookList.aspx?sercate1=001001000#anchor');
console.log('url.parse():',parsedUrl);
console.log('url.format():',url.format(parsedUrl));
```

WHATWG

```
new URL(): URL {
  href: 'http://www.gilbut.co.kr/book/bookList.aspx?sercate1=001001000#anchor',
  origin: 'http://www.gilbut.co.kr',
  protocol: 'http:',
  username: '',
  password: '',
  host: 'www.gilbut.co.kr',
  hostname: 'www.gilbut.co.kr',
  port: '',
  pathname: '/book/bookList.aspx',
  search: '?sercate1=001001000',
  searchParams: URLSearchParams { 'sercate1' => '001001000' },
  hash: '#anchor'
}
url.format(): http://www.gilbut.co.kr/book/bookList.aspx?sercate1=001001000#anchor
----------------------
url.parse(): Url {
```

```
url.format(): http://www.gilbut.co.kr/book/bookList.aspx?sercate1=001001000#anchor
----------
url.parse(): Url {
  protocol: 'http:',
  slashes: true,
  auth: null,
  host: 'www.gilbut.co.kr',
  port: null,
  hostname: 'www.gilbut.co.kr',
  hash: '#anchor',
  search: '?sercate1=001001000',
  query: 'sercate1=001001000',
  pathname: '/book/bookList.aspx',
  path: '/book/bookList.aspx?sercate1=001001000',
  href: 'http://www.gilbut.co.kr/book/bookList.aspx?sercate1=001001000#anchor'
}
url.format(): http://www.gilbut.co.kr/book/bookList.aspx?sercate1=001001000#anchor
```

- 기존 모듈 메서드

  url.parse(주소) : 주소를 분해, WHATWG방식과 비교하면 username과password 대신 auth속성, searchParams 대신 query

  url.format(객체) : WHATWG방식의 url과 기존 url 모두 가능, 분해된 url을 원래대로 조립

- searchParams

  WHATWG 방식에서 쿼리 스트링(search)부분 처리를 도와주는 객체

  ?page=3&limit=10&category=nodejs&category=javascript

```
const {URL} = require('url');
const myURL = new URL('http://www.gilbut.co.kr/?page=3&limit=10
&category=nodejs&category=javascript');
console.log('searchParams:',myURL.searchParams);
console.log('searchParams.getAll():',myURL.searchParams.getAll('category'
));
console.log('searchParams.get():',myURL.searchParams.get('limit'));
console.log('searchParams.has():',myURL.searchParams.has('page'));
console.log('searchParams.keys():',myURL.searchParams.keys());
console.log('searchParams.valuse():',myURL.searchParams.values());
myURL.searchParams.append('filter','es3');
myURL.searchParams.append('filter','es5');
console.log(myURL.searchParams.getAll('filter'));
myURL.searchParams.set('filter','es6');
console.log(myURL.searchParams.getAll('filter'));
myURL.searchParams.delete('filter');
console.log(myURL.searchParams.getAll('filter'));
console.log('searchParams.toString():',myURL.searchParams.toString());
myURL.search = myURL.searchParams.toString();
```

```
searchParams: URLSearchParams {
  'page' => '3',
  'limit' => '10',
  'category' => 'nodejs',
  'category' => 'javascript' }
searchParams.getAll(): [ 'nodejs', 'javascript' ]
searchParams.get(): 10
searchParams.has(): true
searchParams.keys(): URLSearchParams Iterator { 'page',
'limit', 'category', 'category' }
searchParams.valuse(): URLSearchParams Iterator { '3', '10',
'nodejs', 'javascript' }
[ 'es3', 'es5' ]
[ 'es6' ]
[]
searchParams.toString(): page=3&limit=10
&category=nodejs&category=javascript
```

- queryString

  기존 노드 방식에서 쿼리 스트링(search)부분 처리르 ㄹ도와주는 객체

  querystring.parse(쿼리) : url의 query부분을 자바스크립트 객체로 분해해준다

  querystring.stringify(객체) : 분해된 query객체를 문자열로 다시 조립

4. crypto(단방향 암호화)

   암호화는 가능, 복호화는 불가능

   암호화 : 평문을 암호로 만듦　　복호화 : 암호를 평문으로 해독

   단방향 암호화의 대표 주자는 해시 기법

   abcdefgh문자열 -> qvwv

비밀번호	← X	해시 함수	⇄	다이제스트

   해시는 복호화가 안되지만, 해시 함수는 항상 그 단어를 동일하게 변경시킨다

5. Hash 사용하기