1. require를 사용한 모듈 로딩과 기본 경로
  1)Node 어플리케이션에 모듈을 포함시키려면 require문을 사용하여 모듈에 대한 식별자 문자열을 전달한다.
    var http = require('http');

  2)모듈의 전체 개체가 아닌 특정 개체만을 포함시킬 수도 있다.
    var spawn = require('child_process').spawn;

  3)모듈 식별자(예를 들면 'http', 'util' 같은)를 부여하여 Node가 본래 가지고 있는 코어 모듈(기본모듈)이나 node_modules 폴더에 있는 모듈을 로드할 수 있다.

  4)만일 코어 모듈에 속하지도 않고 node_module 폴더에도 포함되지 않은 모듈(예를 들면 사용자 모듈)들은 경로를 나타내기 위해 '/'를 포함해야 한다.

  5)다음의 예를 보자. 다음 명령은 Node 어플리케이션과 동일한 디렉토리에서 mymodule.js라는 모듈을 찾으려는 시도를 한다.
    require('./mymodule');

  6)또는 전체 경로를 사용할 수 있다.
    require('/home/myname/myapp/mymodule.js');

  7)모든 모듈은 .js, .node, .json 파일 확장자 중 하나를 가지게 된다.

  8)코어 모듈(기본 모듈)은 외부 모듈보다 더 높은 우선순위를 가진다.
    -만약 http라는 사용자 정의 모듈을 로드하려고 시도하면, Node는 코어 http 모듈을 로드하게 된다.
    -그렇지 않으려면 다른 모듈 식별자를 사용하거나, 전체 경로를 제공해야 한다.

  9)모듈 검색 순서
    -만일 경로가 없이 Node 식별자를 지정할 경우 해당 모듈은 코어 모듈이 아니므로 Node에서는 먼저 어플리케이션 하위에 node_modules 폴더를 찾는다.
    -이 폴더에서 해당 모듈을 찾지 못하면, 부모 하위 디렉토리 node_modules 폴더를 찾게 된다.
    -만일 모듈 이름이 module 이고, 어플리케이션이 아래의 경로를 갖는다고 가정하자.
      /home/myname/myprojects/myapp
    -Node는 아래의 순서대로 해당 모듈을 찾는다.
      --/home/myname/myprojects/myapp/node_modules/mymodule.js
      --/home/myname/myprojects/node_modules/mymodule.js
      --/home/myname/node_modules/mymodule.js
      --/home/node_modules/mymodule.js
      --/node_modules/mymodule.js

  10)require의 두가지 변형된 형태
    -require.resolve
      --주어진 모듈에 대한 탐색을 수행하기는 하지만, 모듈을 로드하는 대신 탐색된 파일 이름만을 리턴한다.
    -require.cache
      --로딩된 모든 모듈의 캐시된 버전을 갖고 있다.
      --동일한 컨텍스트에서 모듈을 다시 로딩하면 캐시로부터 로드된다.
      --강제로 새로 로드하려면 캐시를 삭제해야 한다.
        var circle = require('/absolutepath/circle.js');
      --다음과 같이 삭제한다.
        delete require.cache('./circle.js');
      --이렇게 하면 다음 번에 require가 호출될 때 강제로 모듈을 새로 로드하게 된다.

2. 외부 모듈과 Node 패키지 관리자
  1)Node와 관련된 풍부한 기능들 대부분은 서드파티 모듈을 통해서 제공된다.

2)외부 모듈을 설치하는 여러 방법(예를 들면 github같은...)중 가장 간단한 방법은 Node 패키지 관리자(npm, Node Package Manager)를 사용하는 것이다.

3. npm
1)여러 문서들
-Home page : https://www.npmjs.com
-기본적인 지침 : https://docs.npmjs.com/policies/README
-사용 설명서 : https://docs.npmjs.com/
-모듈의 local 설치와 global 설치와의 차이점 : https://nodejs.org/en/blog/npm/npm-1-0-global-vs-local-installation/

2)Issac Z. Schlueter(아이작 슐레터)가 만든 Node.js를 위한 패키지 매니저
-Node Package Manager의 약자는 아니다.
-npm faq를 보면 npm is not an acronym.이라는 강조가 있다.
-npm 문서에 따르면 npm은 키보드에서 오른손만으로 명령어와 옵션을 위한 대쉬(-)를 입력할 수 있게 지어진 것이라고 한다.

3)js package portal
-확장 모듈(패키지) 검색
-node.js 생태계에서 많은 개발자가 만들어 공유하고 있는, 확장 모듈의 추가설치 등 관리를 쉽게 할 수 있게 도와주는 기능을 한다.
-JavaScript 개발시 직접 작성한 JavaScript로만 개발하지 않고 jQuery나 AngularJS등 라이브러리를 많이 사용하듯이 Node.js를 개발할 때도 여러 가지 확장 모듈을 많이 이용하게 된다.
-확장 모듈은 많은 개발자가 사용하여 검증된 효과가 있을뿐만 아니라 생산성에도 많은 도움을 준다.
-npm은 사용자가 접근할 수 있는 중앙저장소를 제공하고 각 확장 모듈 개발자가 자신이 만든 확장 모듈을 패키징해서 중앙서버로 등록하며, npm 클라이언트를 통해 중앙저장소에서 원하는 확장 모듈을 가져다가 설치하는 방식으로 사용된다.

4)npm 사이트
-모듈 검색
-모듈 상세 정보
-모듈 설치 방법
-API 설명, 예제

5)패키니매니저 : npm
-모듈 설치, 삭제(확장 모듈)
-모듈 검색
-모듈 정보
-패키지 정보 작성

6)콘솔 명령
-$ npm -v
3.10.10
-$ npm help npm
--Linux : npm man page
--Windows : file:///C:/Program%20Files/nodejs/node_modules/npm/html/doc/cli/npm.html
-$ npm
Usage: npm <command>

where <command> is one of:
access, adduser, bin, bugs, c, cache, completion, config,
ddp, dedupe, deprecate, dist-tag, docs, edit, explore, get,
help, help-search, i, init, install, install-test, it, link,
list, ln, login, logout, ls, outdated, owner, pack, ping,

```
        prefix, prune, publish, rb, rebuild, repo, restart, root,
        run, run-script, s, se, search, set, shrinkwrap, star,
        stars, start, stop, t, tag, team, test, tst, un, uninstall,
        unpublish, unstar, up, update, v, version, view, whoami

    npm <cmd> -h     quick help on <cmd>
    npm -l           display full usage info
    npm help <term>  search for help on <term>
    npm help npm     involved overview

    Specify configs in the ini-formatted file:
        C:\Users\Instructor\.npmrc
    or on the command line via: npm <command> --key value
    Config info can be viewed via: npm help config

    npm@3.10.10 C:\Program Files\nodejs\node_modules\npm

7)npm 주요 옵션
  -init : 패키지 준비. package.json 생성
  -install : 패키지에 필요한 모듈 설치
  -install [MODULE] : 개별 패키지 설치
  -list : 설치된 모듈 목록 보기
  -info : 모듈 정보
  -search : 모듈 검색
  -update : 모듈 업데이트
  -uninstall : 모듈 삭제

8)모듈 설치
  -전역 설치
    --한번 설치로 모든 프로젝트에서 사용
    --라이브러리 폴더. (ex. /usr/lib/node_modules)
    --관리자 권한 필요
    --g 옵션
    --중요하지만 필요하지 않으면 최대한 피하는 것이 좋다.
  -지역 설치
    --프로젝트마다 설치
    --현재 폴더  내 node_modules 폴더

9)모듈 지역 설치
  -$ npm install [Module][@Version]
  -설치 예
    $ npm install async
       |---async@2.3.0
          |-------lodash$4.17.4
  -만일 패키지가 버전을 가지고 있을 경우 특정 버전을 설치할 수 있다.
    $ npm install jade@1.0.0

    $ npm install request
  -node_modules 폴더 내 모듈 설치된다.

10)설치된 모듈 목록
  -모듈 목록 보기
    $ npm list
      C:\NodeHome
```

```
          |-----async@2.3.0
             |--------lodash@4.17.4
          |-----request@2.81.0
              |
              |....

    $ npm list async
       |-------async@2.3.0

11)모듈 검색
 -$ npm search
  --$ npm search html5 parser  <--html5와 parser 모듈 검색
    npm WARN Building the local index for the first time, please be patient npm WARN notice
    update to the newest npm client for improved search results: npm js.com/get-npm
    NAME              DESCRIPTION                                          AU
    352-fabric        Object model for HTML5 canvas, and SVG-to-canvas parser.... =d
    browser-x         A partial implementation of the W3C DOM API on top of an... =a
    context-parser    HTML5 Context Parser                                 =n
    dedom             A partial implementation of the W3C DOM API on top of an... =m
    fabric            Object model for HTML5 canvas, and SVG-to-canvas parser.... =a
    gumbo-node        A node.js wrapper for Google's gumbo html5 parser    =e
    html5             HTML5 HTML parser, including support for SVG and MathML...  =a
    html5-papandreou  HTML5 HTML parser, including support for SVG and MathML...  =m
    html5-parser      html5 parser for node.js and browsers                =n
    id3js             A modern ID3 parser written completely in JavaScript,...    = 4
    kaj               A Kaj to HTML5 parser for Kaj Markup Language.       =j
    mp4js             A modern MP4 parser written completely in JavaScript,...    =l
    neutron-html5parser Small Pure-JS HTML5 Parser                         =m
    parse-srcset      A spec-conformant JavaScript parser for the HTML5 srcset... =a
    parsonic          Ultra fast HTML5 parser, using native chromium functions... =  o
    react-native-html5  html, html5 parser for react-native              =p
    safe-html         A whitelist-based HTML santizier based on the...     =a

12)모듈 정보 보기
 -$ npm info

13)모듈 업데이트
 -$ npm update [모듈명]
 -새로운 모듈이 있는지 확인하고, 만약 있으면 업데이트 수행

14)모듈 삭제
 -$ npm uninstall [모듈명]

15)설치된 패키지와 종속 관계를 나열
 -$ npm ls
  /home/instructor/NodeHome
    |------async@2.3.0
       |------lodash@4.17.4

16)전역 모듈
 -전역 모듈
  node.js의 중앙 라이브러리에 설치 : /usr/lib/modules
 -전역 모듈 다루기
  --npm 명령에 -g 옵션만 추가
```

```
206       --관리자 권한
207
208    17)전역 모듈 설치와 삭제
209      -설치
210        [sudo] npm install -g nodemon
211      -삭제
212        [sudo] npm uninstall -g nodemon
213
214    18)전역 모듈과 지역 모듈 중
215      -지역모듈권장
216        --개발에 필수 라이브러리
217        --특정 버전 모듈에 의존적인 상황
218      -유틸성(mocha, nodemon)은 전역 설치 권장
219
220    19)Lab
221      $ npm install async
222      $ npm install jade
223
224      $ npm list
225      $ npm uninstall jade
226
227      $ npm install -g mocha
228        In Windows,
229        C:\Windows\system32>npm install mocha -g
230        C:\Users\Instructor\AppData\Roaming\npm\_mocha ->
           C:\Users\Instructor\AppData\Roaming\npm\node_modules\mocha\bin\_mocha
231        C:\Users\Instructor\AppData\Roaming\npm\mocha ->
           C:\Users\Instructor\AppData\Roaming\npm\node_modules\mocha\bin\mocha
232        C:\Users\Instructor\AppData\Roaming\npm
233
234        In Linux,
235        /usr/bin/mocha -> /usr/lib/node_modules/mocha/bin/mocha
236        /usr/bin/_mocha -> /usr/lib/node_modules/mocha/bin/_mocha
237        /usr/lib
238
239      //package.json 생성하기
240      $ npm init
241
242
243  4. 확장 모듈 찾기
244    1)node.js에서 많이 사용되는 주요 모듈
245      -현재 공개된 npm 모듈은 5만개이상이다.
246
247    2)Modules ranking : https://nodejsmodules.org/tags/build
248
249    3)29 NODE.JS FRAMEWORKS FOR FAST WEB APPLICATION DEVELOPMENT
250      -http://www.queness.com/post/16219/29-nodejs-frameworks-for-fast-javascript-development
251
252    4)추천 모듈들
253      -Underscore : 일반적인 JavaScript 유틸리티 함수들 제공
254      -Coffee-script : JavaScript로 컴파일되는 언어인 CoffeeScript을 사용할 수 있게 해줌
255      -Request : 단순화된 HTTP 요청 클라이언트
256      -Express : 프레임워크
257      -Optimist : 경량화된 옵션 해석을 제공
```

-Async : 비동기 코드를 위한 함수와 패턴 제공
-Connect : 미들웨어
-Colors : 콘솔에 색상 추가
-Uglify-js : 파서, 압축, 정리(beautifier)
-Socket.IO : 실시간 클라이언트/서버 통신을 가능하게 해줌
-Redis : Redis 클라이언트
-Jade : 템플릿 엔진
-Commander : 명령줄 프로그램용
-Mime : 파일 확장자와 MIME 매핑에 대한 지원 제공
-JSDOM : W3C DOM을 구현

5. 확장 모듈 로컬로 설치하기
1)Colors(https://github.com/Marak/colors.js)
-console.log 출력에 다른 생상 및 스타일 효과를 제공하는 모듈
-Linux에서 테스트할 것(Windows는 효과 미비)
-설치
$ npm install colors
/home/instructor/NodeHome
|------colors@1.1.2

-새로운 REPL을 열고 colors 라이브러리를 포함시킨다.
> var colors = require('colors');
> console.log('This Node Kicks it!'.rainbow.underline);
--메시지가 여러가지 색상으로 밑줄이 표시되어 랜더링됨.

> console.log('We be Nodin'.zebra.bold);
--zebra와 bold 적용

> console.log('rainbow'.rainbow, 'zebra'.zebra);
--콘솔 메시지의 섹션 별로 스타일 변경 가능

-어디에 유용할까? 특정 모듈에서 발생한 오류와 다른 모듈에서 발생한 경로를 다른 색깔로 표시하는 것과 같이 다양한 이벤트에 포맷을 지정할 수 있다.
-Colors의 사전정의 집합을 정의하거나 사용자 정의 테마를 만들면 된다.
> colors.setTheme({
... mod1_warn: 'cyan',
... mod1_error: 'red',
... mod2_note: 'yellow'
... });
undefined
> console.log('This is a helpful message'.mod2_note);
This is a helpful message <--- 노란색 글자
> console.log('This is a bad message'.mod1_error);
This is a bad message <--- red 글자

2)Optimist(https://github.com/substack/node-optimist)
-특정한 문제를 해결하는 데 초점을 맞춘 모듈
-명령어 옵션 파싱 수행하는 모듈
-Refer to
http://zzoon.github.io/insidejs/2011/12/10/node-js-ec-9c-a0-ec-9a-a9-ed-95-9c-eb-aa-a8-eb-93-88-2-optimist.html

-설치

```
  $ npm install optimist
  /home/instructor/NodeHome
      |-------optimist@0.6.1
          |-------minimist@0.0.10
          |-------wordwrap@0.0.3

-app.js : 짧은 옵션
  var argv = require('optimist').argv;
  console.log(argv.o + ' ' + argv.t);
  -------------------------------------
  $ node app.js -o 5 -t 9
  5  9

-app1.js : 긴 옵션
  var argv = require('optimist').argv;
  console.log(argv.one + '  ' + argv.two);
  -------------------------------------
  $ node app1.js --one='My'  --two='Name'
  'My' 'Name'

-app2.js : 긴 인자 파싱
  var argv = require('optimist').argv;
  if (argv.rif - 5 * argv.xup > 7.138) {
     console.log('Buy more riffiwobbles');
  }else {
     console.log('Sell the xupptumblers');
  }
  -------------------------------------
  $ node app2.js --rif=55 --xup=9.52
  Buy more reiffiwobbles

  $ node app2.js --rif 12  --xup 8.1
  Sell the xupptumblers

-app3.js : 짧은 인자 지원
  var argv = require('optimist').argv;
  console.log('(%d,%d)', argv.x, argv.y);
  -----------------------------------------
   $ node app3.js -x 10 -y 21
   (10, 21)

-app4.js : boolean 인자 지원
  var argv = require('optimist').argv;
  if (argv.s) {
     console.log(argv.fr ? 'Le chat dit: ' : 'The cat says: ');
  }
  console.log(
     (argv.fr ? 'miaou' : 'meow') + (argv.p ? '.' : '')
  );
  -----------------------------------------
  $ node app4.js -s
    The cat says: meow

  $ node app4.js -sp
```

```
363       The cat says: meow.
364
365     $ node app4.js -sp --fr
366       Le chat dit: miaou.
367
368   -app5.js : 하이픈으로 시작하지 않는 옵션 지원
369       var argv = require('optimist').argv;
370       console.log('(%d,%d)', argv.x, argv.y);
371       console.log(argv._);
372     ---------------------------------------------
373     $ node app5 -x 6.82 -y 3.35 moo
374     (6.82,3.35)
375     [ 'moo' ]
376
377     $ node app5 foo -x 0.54 bar -y 1.12 baz
378     (0.54,1.12)
379     [ 'foo', 'bar', 'baz' ]
380
381   -app6.js : .usage(), .demand() 함수 제공
382     var argv = require('optimist')
383     .usage('Usage: $0 -x [num] -y [num]')
384     .demand(['x','y'])
385     .argv;
386     console.log(argv.x / argv.y);
387     ---------------------------------------
388     $ node app6 -x 55 -y 11
389     5
390
391     $ node ./app6.js -x 4.91 -z 2.51
392     Usage: node ./divide.js -x [num] -y [num]
393
394     Options:
395        -x  [required]
396        -y  [required]
397
398     Missing required arguments: y
399
400   -app7.js : .default()를 통해 디폴트 인자 사용 가능
401     var argv = require('optimist')
402     .default('x', 10)
403     .default('y', 10)
404     .argv;
405     console.log(argv.x + argv.y);
406     ---------------------------------------
407     $ node app7  -x 5
408     15
409
410 3)Underscore
411   -Node용 유틸리티 모듬 라이브러리
412   -jQuery나 Prototype.js와 같은 서드파티 라이브러리에서 사용했던 JavaScript 확장 기능을 제공
413   -Underscore라는 이름이 붙여진 이유는 기능에 접근할 때 jQuery의 $와 유사하게 밑줄(_)을 사용하기 때문
414   -설치
415     $ npm install underscore
416     /home/instructor/NodeHome
```

```
        |----------underscore@1.8.3

  -underscore.js
    var _= require('underscore');
    _.each(['apple', 'lime', 'cherry', 'melon'], function(fruit){
      console.log(fruit);
    });
    ---------------------------------------
    apple
    lime
    cherry
    melon

  -위에서 '_' 대신 us로 변경 가능
    var us = require('underscore');
    us.each(['apple', 'lime', 'cherry', 'melon'], function(fruit){

  -이 모듈은 배열, 컬렉션, 함수, 개체, 체인, 일반적인 유틸리티 등의 확장기능등을 제공한다.
  -mixin()을 통해 Underscore를 개발자가 만든 유틸리티 함수로 확장할 수 있는 기능 제공
  -underscore1.js
    var _= require('underscore');
    _.mixin({
        betterWithNode : function(str){
            return str + ' is beter with Node';
        }
    });
    console.log(_.betterWithNode('chocolate'));
    ----------------------------------------------------
    chocolate is beter with Node

4) Read
  -표준입력(stdin)으로부터 쉽게 입력값을 읽기 위한 모듈

  -설치
    $ npm install read

  -Lab1
    var read = require('read');

    read({ prompt : 'Username: ' }, function (err, user) {
      read({ prompt : 'Password: ', silent : true }, function (err, pass) {
        console.log(user, pass);
        process.stdin.destroy();
      });
    });

  -Lab2 : 위의 코드 변환
    var read = require('read');
    var Seq = require('seq');

    Seq()
      .seq(function () {
        read({ prompt : 'Username: ' }, this.into('user'));
      })
```

```
471         .seq(function () {
472          read({ prompt : 'Password: ', silent : true }, this.into('pass'));
473         })
474         .seq(function (pass) {
475          console.log(this.vars.user, this.vars.pass);
476         });
477
478    -다음은 read가 지원하는 옵션들이다.
479      -prompt  - 입력을 받기 전에 출력창에 나타날 값 지정.
480      -silent  - 유저가 입력한 내용을 출력하지 마라.
481      -num     - 터미널에서 읽어 올 최대 문자수.
482      -delim   - 입력 완료를 나타내는 문자. 디폴트: "\n"
483      -timeout - 유저 입력을 기다리는 최대 시간(단위: ms).
484
485    -주의: silent가 true이거나 num이 설정됐거나, delim이 "\n"가 아닌 다른 값으로 되어 있다면, read는 raw모드로
       동작해서, 문자 단위로 읽을 것이다.
486
487  5) Dnode
488    -자유 형태의 rpc 라이브러리
489
490    -설치
491      $ npm install dnode
492
493    -Lab1 :
494      <server.js>
495      var dnode = require('dnode');
496
497      var server = dnode({
498        mul : function (n, m, cb) { cb(n * m) }
499      });
500      server.listen(5050);
501
502      <client.js>
503      var dnode = require('dnode');
504
505      dnode.connect(5050, function (remote) {
506        remote.mul(10, 20, function (n) {
507          console.log('10 * 20 = ' + n);
508        });
509      });
510      ---------------------------------
511      $ node client.js
512      200
513
514    -Lab2
515      <server.js>
516
517      var dnode = require('dnode');
518
519      var server = dnode(function (client) {
520        this.calculate = function (n, m, cb) {
521          client.div(n*m, function (res) {
522            cb(res+1)
523          });
```

```
524       }
525     });
526     server.listen(5050);
527
528     <client.js>
529     var dnode = require('dnode');
530
531     var client = dnode({
532       div : function (n, cb) {
533         cb(n/5);
534       }
535     });
536
537     client.connect(5050, function (remote) {
538       remote.calculate(10, 20, function (n) {
539         console.log('the result is ' + n);
540       });
541     });
542
543
544   6) Everyauth
545     -connect 모듈의 미들웨어로 사용되며 페이스북, 트위터, 구글 등과 같은 서비스의 인증을 처리하기 위한 모듈
546
547     -설치
548       $ npm install connect
549       $ npm install everyauth
550
551     -모듈을 사용하기 위해서는 다음과 같이 connect에서의 미들웨어 설정을 한다.
552       var everyauth = require('everyauth');
553       var connect = require('connect');
554
555       var app = connect(everyauth.middleware());
556
557     -인증 처리를 할 서비스들의 비밀키를 다음과 같이 config.json에 구성한다.
558         module.exports = {
559           fb: {
560             appId: '111565172259433'
561           , appSecret: '85f7e0a0cc804886180b887c1f04a3c1'
562           },
563           twit: {
564             consumerKey: 'JLCGyLzuOK1BjnKPKGyQ'
565           , consumerSecret: 'GNqKfPqtzOcsCtFbGTMqinoATHvBcy1nzCTimeA9M0'
566           },
567           github: {
568             appId: '11932f2b6d05d2a5fa18'
569           , appSecret: '2603d1bc663b74d6732500c1e9ad05b0f4013593'
570           },
571           // ...
572         };
573
574     -사용하려는 서비스의 로그인을 수행하는 코드를 작성한다. (아래 코드는 페이스북 로그인 예제다)
575       var conf = require('./config.json');
576       var usersByFbId = {};
577       everyauth
```

```
        .facebook
        .appId(conf.fb.appId)
        .appSecret(conf.fb.appSecret)
        .findOrCreateUser(function (session, accessToken, accessTokenExtra, fbUserMetadata) {
         return usersByFbId[fbUserMetadata.id] ||
           (usersByFbId[fbUserMetadata.id] = addUser('facebook', fbUserMetadata));
        })
        .redirectPath('/');


7)Twitter
  Refer to https://www.npmjs.com/package/twitter
  Refer to https://github.com/desmondmorris/node-twitter/tree/master/examples
  -트위터를 사용하기 위한 모듈

  -설치
    $ npm install twitter

  -Lab : Posting
     var Twitter = require('twitter');

     var client = new Twitter({
       consumer_key: '',
       consumer_secret: '',
       access_token_key: '',
       access_token_secret: ''
     });

     client.post('statuses/update', {status : 'Hello, World'}, function(error, tweet, response){
        if(error){
           console.log("Error : ", error);
        }
        console.log(tweet);
     });

  -Lab : Posting
     var Twitter = require('twitter');
     var readline = require('readline');
     var r = readline.createInterface({
        input:process.stdin,
        output:process.stdout
     });

     var client = new Twitter({
       consumer_key: '',
       consumer_secret: '',
       access_token_key: '',
       access_token_secret: ''
     });

     r.question("Twitter에 보내실 메시지를 입력해 주세요 : ", function(answer) {
        client.post('statuses/update', {status : answer }, function(error, tweet, response){
           if(!error)console.log(tweet);
        });
```

```
        r.close();
      });


6. 확장 모듈을 글로벌로 설치하기 : nodemon
   1)소개
     -node명령어로 노드 어플리케이션을 실행하면 소스가 메모리에 올라가기 때문에 소스를 수정해도 어플리케이션이 실행되는 중에는 적용되지 않는다.
     -소스를 수정하고 적용된 결과를 확인하기 위해 매번 재시작하는 것은 간단한 명령어임에도 귀찮은 작업니다.
     -nodemon은 매번 재시작해야 하는 반복 작업을 자동화하는 도구이다.
     -node source.js하는 대신 nodemon source.js로 실행하면 현재 폴더 하위에 있는 자바스크립트 파일을 감시하다가 파일이 수정되면 자동으로 재시작한다.
     -소스를 hot-deployed하는 방식은 아니지만, 노드 어플리케이션은 실행되는 데 시간이 거의 걸리지 않으므로 nodemon을 이용하면 상당한 생산성을 얻을 수 있다.

   2)수정후 자동 재시작
     $ nodemond  source.js

   3)설치
     -글로벌로 설치
     -관리자 권한 필요
      [sudo] npm install -g nodemon

   4)실행
     $ nodemon  app.js

   5)Lab
      $ npm install -g nodemon
      In Linux,
      /usr/bin/nodemon -> /usr/lib/node_modules/nodemon/bin/nodemon.js
      /usr/lib
        |------nodemon@1.11.0
           |-------chokidar@1.6.1
           ...
      npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^1.0.0
      (node_modules\nodemon\node_modules\chokidar\node_modules\fsevents):
        npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for
        fsevents@1.1.1: wanted {"os":"darwin","arch":"any"} (current:
        {"os":"win32","arch":"x64"})

      In Windows,
        C:\Users\Instructor\AppData\Roaming\npm\nodemon ->
        C:\Users\Instructor\AppData\Roaming\npm\node_modules\nodemon\bin\nodemon.js
        C:\Users\Instructor\AppData\Roaming\npm
          `-- nodemon@1.11.0
            +-- chokidar@1.6.1
            | +-- anymatch@1.3.0


        npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^1.0.0
        (node_modules\nodemon\node_modules\chokidar\node_modules\fsevents):
        npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for
        fsevents@1.1.1: wanted {"os":"darwin","arch":"any"} (current:
        {"os":"win32","arch":"x64"})
```

```
676
677       $ npm list nodemon
678
679    6)Lab
680      <nodemondemo.js>
681      console.log('Hello World');
682
683      $nodemon nodemondemo
684      [nodemon] 1.11.0
685      [nodemon] to restart at any time, enter `rs`
686      [nodemon] watching: *.*
687      [nodemon] starting `node nodemondemo chat.js`
688      Hello World
689      [nodemon] clean exit - waiting for changes before restart
690
691      console.log('Hello Node.js');  <--변경 후 저장
692
693      아래 내용 추가됨
694      [nodemon] restarting due to changes...
695      [nodemon] starting `node nodemondemo chat.js`
696      [nodemon] restarting due to changes...
697      Hello Node.js
698      [nodemon] starting `node nodemondemo chat.js`
699      Hello Node.js
700      [nodemon] clean exit - waiting for changes before restart
701
702
703  7. 패키지 정보
704    1)패키지 설정 파일
705      -package.json
706      -npm init으로 생성
707      -패키지에 대한 정보 입력
708      -반드시 프로젝트의 루트에 위치해야 한다.
709
710    2)패키지 정보 파일 : package.json
711      {
712        "name": "npm-sample",
713        "version": "1.0.0",
714        "description": "npm sample",
715        "main": "chat.js",
716        "dependencies": {
717          "async": "^2.3.0",
718          "jade": "^1.11.0",
719          "request": "^2.81.0"
720        },
721        "devDependencies": {},
722        "scripts": {
723          "test": "echo \"Error: no test specified\" && exit 1"
724        },
725        "author": "",
726        "license": "ISC"
727      }
728      -name
729        --프로젝트 이름
```

```
  --version과 함께 필수 항목
  --배포할 때 모듈의 유일한 키 값으로 사용
  --URL로 사용되고 설치시 디렉토리명이 되기 때문에 URL이나 디렉토리에서 쓸 수 없는 이름은 안된다.
  --js나 node 같은 문자를 포함하지 않기를 권장
-version
  --프로젝트 버전
  --필수 항목
  --버전을 기반으로 설치되기 때문에 프로젝트 내에서 유일한 값이어야 한다.
  --v로 시작할 수 있으며 3단계 버전 사용하는데, 뒤에 하이픈으로 태그명을 적을 수 있다.
  --예를 들어, 0.1.2, 0.1.2-beta 등이 가능하다.
-description
  --프로젝트 설명
  --npm search를 사용할 때 검색에 포함
-keywords
  --프로젝트 키워드를 문자열의 배열로 작성하고 npm search를 할 때 keywords에 적은 문자열도 같이 찾아준다.
  --예를 들어 ['twitter', 'streaming', 'oauth']
-homepage
  --프로젝트 홈페이지 주소
-author
  --작성자 정보
  --Outsider<outsider+nodejs@gmail.com>(http://blog.outsider.ne.kr)과 같이 한줄로 작성하거나
  JSON 형식으로 작성한다.
  --email과 url은 옵션 필드이다.
  {
    "name" : "outsider",
    "email" : "outsider@gmail.com",
    "url" : "http://..."
  }
-contributors
  --프로젝트에 참여한 공헌자 정보
  --보통 노드 확장 모듈이 오픈소스로 개발하기 때문에 공헌자의 항목이 따로 있다.
  --authors에서 사용한 형식을 배열로 적는다.
-repository
  --소스를 확인할 수 있는 저장소의 주소
  "repository" :
  {
    "type" : "git",
    "url" : "http://git-repository-url"
  }
  "repository" :
  {
    "type" : "svn",
    "url" : "http://subversion-repository-url"
  }
-scripts
  --프로젝트에서 자주 실행해야 하는 명령어를 scripts로 작성하면 npm 명령어로 실행할 수 있다.
  {
    "scripts" :
      {
         "start" : "node app.js",
         "install" : "make & make install",
         "test" : "node ./test/*"
      }
  }
```

783 --이렇게 작성하면 프로젝트를 실행할 때, node app.js를 입력하는 대신 npm start 라고 입력해서 프로젝트를 실행할 수 있다.
784 --더 자세한 정보는 npm help scripts로 확인가능
785 -config
786 --위에서 scripts를 사용해 npm start등으로 실행하면 소스에서 config 필드에 있는 값을 환경 변수처럼 접근할 수 있다.
787 {"config" : {"port" : "8080"}}
788 --이처럼 config 필드를 작성하면 소스에서 process.env.npm_package_config_prot 로 접근해서 8080값을 가져올 수 있다.
789 -private
790 --이 값을 true로 지정하면 이 프로젝트를 npm 중앙저장소로 배포하지 않는다.
791 --배포할 목적이 아니라면 실수로 배포되는 것을 방지하기 위해 true로 설정하는 편이 좋다.
792
793 3)패키지 정보 중 의존성 정보
794 "dependencies": {
795 "async": "^2.3.0",
796 "jade": "^1.11.0",
797 "request": "^2.81.0"
798 },
799 "devDependencies": {},
800 "scripts": {
801 "test": "echo \"Error: no test specified\" && exit 1"
802 },
803
804 -dependencies
805 --npm은 dependencies 필드로 의존 확장 모듈을 관리한다.
806 --누구든지 이 필드덕분에 package.json만 공유한다면 의존성 모듈을 제대로 설치할 수 있다.
807 --이 필드를 명시하면 모듈 설치 시 모듈명을 지정하지 않고 프로젝트의 루트에서 npm install 만 입력하면 dependencies에 있는 모듈을 버전 규칙에 맞게 설치한다.
808 --npm update를 실행했을 때 최신 버전이 있더라도 dependencies의 규칙에 맞는 범위 내에서만 업데이트되기 때문에 의존성 확장 모듈이 버전으로 인한 문제를 최소화할 수 있다.
809 -devDependencies
810 --dependencies와 같지만 개발할 때만 필요한 모듈은 여기에 명시한다.
811 --여기에 명시도니 모듈은 config에 production이 true로 돼 있을 때는 설치하지 않는다.
812 --production 기본 값은 false이다.
813 --config값은 npm config list -l 명령어로 확인할 수 있으며, 값을 바꾸려면 다음 명령어로 해야 한다.
814 npm config set production true
815
816 4)기타
817 -engine
818 --dependencies에서 사용한 버전 표시 방식을 이용해 다음과 같이 작성한다.
819 { "engine" : { "node" : ">=0.4.0 < 0.4.11"}}
820 --기반이 되는 엔진 버전을 표시하는 데 npm은 당연히 노드 기반이므로 노드의 버전을 적어주며, 노드의 호환 버전을 표시하는 정보성 필드이다.
821
822 5)패키지 정보 기록하기
823 -JSON 은 간단한 포맷이지만 프로젝트를 만들 때마다 package.json을 만드는 것을 번거로운 일이고, 필드명을 기억하기도 쉽지 않다.
824 -아래 명령을 사용하여 기본적인 package.json을 만들 수 있다.
825 --npm init
826 -이처럼 이 명령을 입력하면 기본적인 사항을 물어보고 적절한 값을 입력하면 그에 맞는 package.json을 생성한다.
827 -추가로 다음처럼 npm install 모듈 설치 시 --save 옵션을 사용하면 존재하는 package.json의 dependencies 필드에 자동으로 모듈에 대한 정보를 추가한다.
828 -save 옵션을 추가하면 버전 정보는 '~현재 버전명' 방식으로 추가된다.

```
  -의존성 정보 기록
    --npm install 모듈이름 --save

6)패키지 의존성 정보 사용하기
  -패키지 정보 파일에 기록된 모듈 설치/업데이트
    npm install
    npm update

7)Lab1
  $ npm init
  C:\NodeHome>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: (NodeHome) npm-sample
    version: (1.0.0)
    description: npm sample
    entry point: (chat.js)
    test command:
    git repository:
    keywords:
    author:
    license: (ISC)
    About to write to C:\NodeHome\package.json:

    {
      "name": "npm-sample",
      "version": "1.0.0",
      "description": "npm sample",
      "main": "chat.js",
      "dependencies": {
        "async": "^2.3.0",
        "request": "^2.81.0"
      },
      "devDependencies": {},
      "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
      },
      "author": "",
      "license": "ISC"
    }


    Is this ok? (yes) yes

    package.json 에 모두 기록돼있다.
  $ npm install jade --save
```

```
883       C:\NodeHome>npm install jade --save
884       npm WARN deprecated jade@1.11.0: Jade has been renamed to pug, please install the latest
          version of pug instead of jade
885       npm WARN deprecated transformers@2.1.0: Deprecated, use jstransformer
886       npm-sample@1.0.0 C:\NodeHome
887       `-- jade@1.11.0
888
889       npm WARN npm-sample@1.0.0 No repository field.
890
891     8)Lab2
892       -Visual Studio Code 의 목록에서 node_modules 폴더를 삭제한다.
893       -$ npm install
894       -그러면 package.json의 내용을 보고 자동으로 새로 설치한다.
895
896       $ npm list
897
898
899   8. 사용자 정의 모듈 만들기
900     1)모듈만들기 이유
901       -소스 코드 분리
902       -클라이언트 JavaScript처럼 재사용 가능한 JavaScript를 라이브러리로 분리하고 싶을 때
903
904     2)모듈작성 방법
905       module.exports
906
907     3)모듈 사용하기
908       -모듈 로딩 : require
909       -require('./greeting.js');
910
911     4)모듈 로딩 에러
912       require('greeting.js');  <--경로를 넣지 않으면 node_modules 에서 찾는다.
913       Error : Cannot find module 'greeting.js'
914
915     5)함수 exports
916       -모듈 greeting.js
917           module.exports.goodMorning = function() {
918             //모듈 함수 기능 작성
919           }
920           exports.goodNight = function(arg, callback) {
921             //module 생략 가능
922           }
923       -사용하기
924         var greeting = require('./greeting.js');
925         greeting.goodMorning();
926
927     6)exports 하지 않은 함수는 사용 불가
928       function goodAfternoon(){
929         console.log('goodAfternoon);
930       }
931       -사용하기
932         var greeting = require('./greeting.js')(;
933         greeting.goodAfternoon();
934       -에러
935         TypeError : greeting.goodAfternoon is not a function
```

```
936
937   7)클래스 정의 exports
938     function BusDef(){
939       this.take = function(){}
940     }
941     function MetroDef(){
942       this.ride = function(){}
943     }
944     module.exports.Bus = BusDef;
945     exports.Metro = MetroDef;   //module 생략 가능
946     -사용하기
947       var Bus = require('./transport').Bus;
948       var bus = new Bus();
949       bus.take();
950
951   8)클래스 exports
952     function Exercise(){
953       this.pushup = function(){}
954     }
955     Exercise.prototype.run = function(){}
956
957     module.exports = Exercise;  //module 생략불가
958     -사용하기
959       var Exercise = require('./exercise');
960       var exercise = new Exercise();
961       exercise.pushup();
962
963   9)객체 exports
964     var student = {
965       hour : 0,
966       study : function(){
967         this.hour++;
968         console.log(this.hour + '시간째 공부중');
969       }
970     };
971     module.exports = student;
972     -사용하기
973       var you = require('./student.js');
974       you.study();
975
976   10)모듈은 캐쉬됨
977     var you = require('./student.js');
978     you.study();   //1시간
979     you.study();   //2시간
980
981     //모듈 로딩
982     var him = require('./student.js');
983     him.study();   //3시간
984
985   11)Lab
986     단순한 JavaScript로 만들어진 함수가 있다.
987     function concatArray(str, array){
988       return array.map(function(element){
989         return str + ' ' + element;
```

```
  }
 }
 이 함수를 Node에서 사용하고자 한다.
 exports 개체를 이용하여 노출된 함수를 내보내야 한다.
 <arrayfunction.js>로 파일 저장
 exports.concatArray = function(str, array){
  return array.map(function(element){
   return str + ' ' + element;
  }
 };
 Node 어플리케이션에서 이 함수를 사용하려면, require를 사용하여 라이브러리를 로딩한다.
 var newArray = require('./arrayfunction.js');
 console.log(newArray.concatArray('hello', ['test1', 'test2']));
 ------------------------------------------------
 [ 'hello test1', 'hello test2' ]


9. 디렉토리 전체 패키징하기
  1)node_modules 내에 inputcheck라는 하위 디렉토리를 만든다.
  2)inputChecker.js 코드 파일을 이곳으로 이동시키고 파일 이름을 index.js로 변경한다.
   <inputChecker.js>
    var util = require('util');
    var eventEmitter = require('events').EventEmitter;
    var fs = require('fs');

    function inputChecker(name, file){
       this.name = name;
       this.writeStream = fs.createWriteStream('./' + file + '.txt',
       {'flags' : 'a',
       'encoding' : 'utf8',
       'mode' : 0666});
    }

    util.inherits(inputChecker, eventEmitter);
    inputChecker.prototype.check = function check(input){
       var self = this;
       var command = input.toString().trim().substr(0,3);
       if(command == 'wr:'){
          self.emit('write', input.substr(3, input.length));
       }else if(command == 'en:'){
          self.emit('end');
       }else{
          self.emit('echo', input);
       }
    }
    var ic = new inputChecker('Shelley', 'output');

    ic.on('write', function(data){
       this.writeStream.write(data, 'utf8');
    });

    ic.on('echo', function(data){
       console.log(this.name + ' wrote ' + data);
    });
```

```
    ic.on('end', function(){
        process.exit();
    });

    process.stdin.resume();
    process.stdin.setEncoding('utf8');
    process.stdin.on('data', function(input){
        ic.check(input);
    });
    ----------------------------------------------------
    $ node inputChecker
    wr:Hello, World
    wr:Good Night
    en:

    $ cat output.txt
    Hello, World
    Good Night
```

3)코드를 수정, exports 개체를 추가한다.

<index.js>

```
    var util = require('util');
    var eventEmitter = require('events').EventEmitter;
    var fs = require('fs');

    exports.inputChecker = inputChecker;  <--여기를 추가한다.

    function inputChecker(name, file){
          this.name = name;
          this.writeStream = fs.createWriteStream('./' + file + '.txt',
          {'flags' : 'a',
          'encoding' : 'utf8',
          'mode' : 0666});
    }

    util.inherits(inputChecker, eventEmitter);
    inputChecker.prototype.check = function check(input){
          var self = this;
          var command = input.toString().trim().substr(0,3);
          if(command == 'wr:'){
                self.emit('write', input.substr(3, input.length));
          }else if(command == 'en:'){
                self.emit('end');
          }else{
                self.emit('echo', input);
          }
    }
```

4)package.json을 생성하기 위해 npm init를 실행한 다음 각 구성 요소에 답을 넣는다.

```
  $ npm init

    <package.json>
    {
      "name": "inputcheck",
```

```
    "version": "0.0.1",
    "description": "Looks for commands within the string and implements the commands",
    "homepage" : "http://inputcheck.burningbird.net",
    "repository" : {
      "url" : ""
    },
    "main": "inputcheck.js",
    "engine" : { "node" : "~6.0.0" },
    "dependencies": {},
    "devDependencies": {},
    "scripts": {
      "test": "echo \"Error: no test specified\" && exit 1"
    },
    "author": "Shelly Powers <shelleyp@burningbird.net> (http://burningbird.net)",
    "license": "ISC"
  }
```

5)테스트 해본다.

<test.js>

```
var inputChecker = require('inputcheck').inputChecker;

var ic = new inputChecker('Shelley', 'output');

ic.on('write', function(data){
   this.writeStream.write(data, 'utf8');
});

ic.on('echo', function(data){
   console.log(this.name + ' wrote ' + data);
});
ic.on('end', function(){
   process.exit();
});

process.stdin.resume();
process.stdin.setEncoding('utf8');
process.stdin.on('data', function(input){
   ic.check(input);
});
```

```
$ node test
  wr:Hello, World
  wr:Good Night
  en:

$ cat output.txt
  Hello, World
  Good Night
```

6)모듈 게시

추가적으로 아래의 내용을 package.json에 넣는다.

```
"directories": {
    "doc": ".",
    "test": "test",
    "example": "examples"
```

```
    },

7)$ npm install . -g
  - acorn@2.7.0 node_modules\inputcheck\node_modules\acorn
  - acorn-globals@1.0.9 node_modules\inputcheck\node_modules\acorn-globals
  - amdefine@1.0.1 node_modules\inputcheck\node_modules\amdefine
  - asap@1.0.0 node_modules\inputcheck\node_modules\asap
  - asn1@0.2.3 node_modules\inputcheck\node_modules\asn1
  - assert-plus@0.2.0 node_modules\inputcheck\node_modules\assert-plus
  - asynckit@0.4.0 node_modules\inputcheck\node_modules\asynckit
  - aws-sign2@0.6.0 node_modules\inputcheck\node_modules\aws-sign2
  - aws4@1.6.0 node_modules\inputcheck\node_modules\aws4
  ....
  C:\Users\Instructor\AppData\Roaming\npm
     |-- inputcheck@0.0.1

```