

# 5. Node.js®

## 중요 내장 모듈

영진전문대학  
컴퓨터정보계열  
김종율

# Global Variables

- Node 어플리케이션에서 사용할 수 있는 전역 변수

- 코드의 실행(runtime)

- function으로 5개의 파라미터를 전달하여 run함
- Debug Perspective 실행

```
1 (function (exports, require, module, __filename, __dirname) { /**
2  * New node file
3  */
4  var http= require('http');
5  http.createServer(function(req,res){
6      res.writeHead(200,{ 'Content-Type': 'text/plain' });
7      res.write('Hello World\n');
8  }).listen(3000);
9  console.log('Server running at http://127.0.0.1:3000/');
10
11  });
```

Node.js 내부 객체: export, require, module  
실행 중인 경로(파일포함): \_\_filename  
실행 중인 경로: \_\_dirname

# Global Variables

```
console.log("__filename : ", __filename);  
console.log("__dirname : ", __dirname);
```

```
__filename: C:\Node.js_IDE\eclipse\workspace\HelloWorld\debuging-sample.js  
__dirname: C:\Node.js_IDE\eclipse\workspace\HelloWorld
```

## Global Objects

- global
- process
- console
- Class: Buffer
- require()
  - require.resolve()
  - require.cache
  - require.extensions
- \_\_filename
- \_\_dirname
- module
- exports
- setTimeout(cb, ms)
- clearTimeout(t)
- setInterval(cb, ms)
- clearInterval(t)

# process object

- node.js에서만 지원되는 객체
- node.js의 여러 기본 정보와 기본 기능들을 제공
- <https://nodejs.org/api/process.html>
  - argv: node 어플리케이션 실행시의 파라미터를 저장
  - execPath: 현재 node 실행파일의 경로
  - cwd(): 현재 node어플리케이션의 경로
  - version: node.js의 버전정보
  - memoryUsage(): 서버 메모리 상태를 반환(byte단위)
  - env: 여러가지 환경설정 정보를 저장

# process object

- process
  - Exit Codes
  - Event: 'exit'
  - Event: 'beforeExit'
  - Event: 'uncaughtException'
  - Signal Events
  - process.stdout
  - process.stderr
  - process.stdin
  - process.argv
  - process.execPath
  - process.execArgv
  - process.abort()
  - process.chdir(directory)
  - process.cwd()
  - process.env
  - process.exit([code])
  - process.exitCode
  - process.getgid()
  - process.setgid(id)
  - process.getuid()
  - process.setuid(id)
  - process.getgroups()
  - process.setgroups(groups)
  - process.initgroups(user, extra\_group)
  - process.version
  - process.versions
  - process.config
  - process.kill(pid[, signal])
  - process.pid
  - process.title
  - process.arch
  - process.platform
  - process.memoryUsage()
  - process.nextTick(callback)
  - process.umask([mask])
  - process.uptime()
  - process.hrtime()
  - process.mainModule

# process object

```
console.log("process: ",process);
```

```
process: { title: 'C:\\Program Files\\nodejs\\node.exe',
  version: 'v0.12.7',
  moduleLoadList:
    [ 'Binding contextify',
      'Binding natives',
      'NativeModule events',
      'NativeModule util',
      'NativeModule buffer',
      'Binding buffer',
      'Binding smalloc',
      'NativeModule path',
      'NativeModule module',
      'NativeModule vm',
      'NativeModule assert',
      'NativeModule fs',
      'Binding fs',
      'Binding constants',
      'NativeModule stream',
      'NativeModule _stream_readable',
      'NativeModule _stream_writable',
      'NativeModule _stream_duplex',
      'NativeModule _stream_transform',
      'NativeModule _stream_passthrough',
      'NativeModule smalloc',
      'NativeModule console',
      'Binding tty_wrap',
      'NativeModule net',
      'NativeModule timers',
      'Binding timer_wrap',
      'NativeModule _linklist',
      'Binding cares_wrap',
      'Binding uv',
      'Binding pipe_wrap',
      'Binding tcp_wrap',
      'Binding stream_wrap' ],
  versions:
    { http_parser: '2.3',
      node: '0.12.7',
```

# process object

- process.argv

```
process.argv.forEach(function(val,index,array){  
    console.log(index+" : "+val);  
});
```

- Project Explorer>소스코드파일  
명>MouseRClick>[Run AS]><Run Config...]
- Arguments Tab>파라미터 입력

- one two three four

```
0 : node  
1 : C:\Node.js_IDE\eclipse\workspace\HelloWorld\args.js  
2 : one  
3 : two  
4 : three  
5 : four
```

# process object

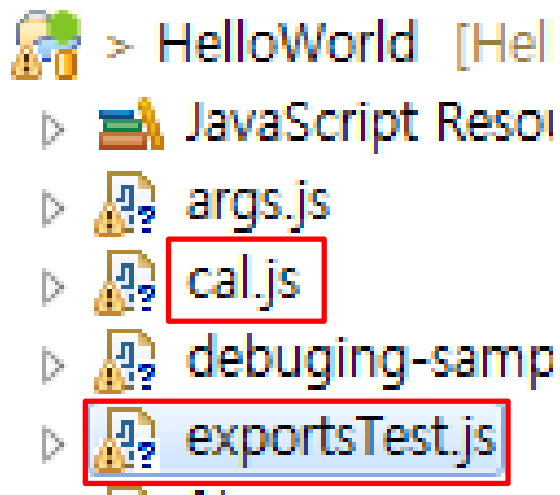
```
console.log("process.execPath      : ",process.execPath);
console.log("process.cwd()         : ",process.cwd());
console.log("process.version       : ",process.version);
console.log("process.memoryUsage(): ",process.memoryUsage());
console.log("process.env                : ",process.env);
```

```
process.execPath      : C:\Program Files\nodejs\node.exe
process.cwd()         : C:\Node.js_IDE\eclipse\workspace\HelloWorld
process.version       : v0.12.7
process.memoryUsage(): { rss: 14241792, heapTotal: 7458432, heapUsed: 2902064 }
process.env           : { PATH: 'C:\\ProgramData\\Oracle\\Java\\javapath;C:\\Windows\\system32;C:\\Windows\\
  SystemDrive: 'C:',
  SystemRoot: 'C:\\Windows',
  TEMP: 'C:\\Users\\LecPC\\AppData\\Local\\Temp',
  TMP: 'C:\\Users\\LecPC\\AppData\\Local\\Temp' }
```



# exports object

- node에서 사용하는 모듈 로딩 시스템
- 모든 모듈은 exports객체로 구현
- 사용자 정의 객체 구현 및 재사용



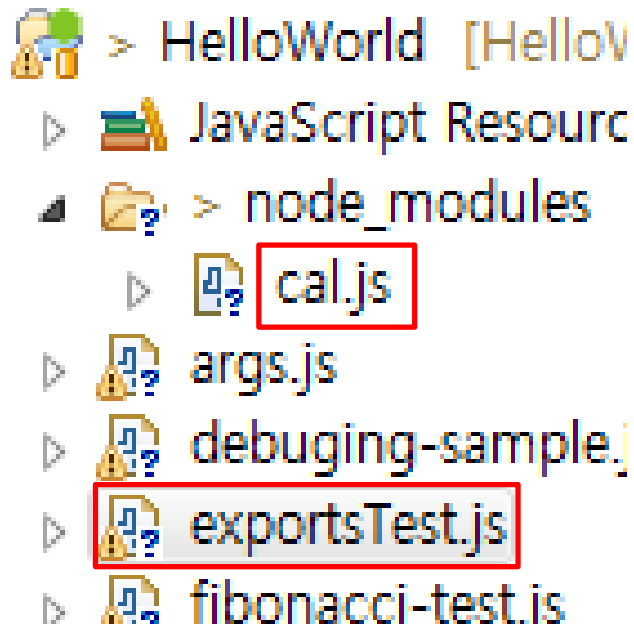
cal.js

```
exports.sum=function(x,y){  
  x = parseInt(x);  
  y = parseInt(y);  
  
  return x+y;  
}
```

exportsTest.js

```
var cal = require('./cal.js');  
console.log(cal.sum(5,6));  
  
11
```

# exports object



exportsText.js

```
var cal = require('cal.js');  
console.log(cal.sum(5,6));
```

# events object

- Event의 등록
  - EventEmitter 클래스의 addListener(), on() 메서드 이용

## Events

### ◦ Class: events.EventEmitter

- emitter.addListener(event, listener)
- emitter.on(event, listener)
- emitter.once(event, listener)
- emitter.removeListener(event, listener)
- emitter.removeAllListeners([event])
- emitter.setMaxListeners(n)
- EventEmitter.defaultMaxListeners
- emitter.listeners(event)
- emitter.emit(event[, arg1][, arg2][, ...])
- Class Method: EventEmitter.listenerCount(emitter, event)
- Event: 'newListener'
- Event: 'removeListener'

# events object

- **process**
  - **Exit Codes**
  - **Event: 'exit'**
  - **Event: 'beforeExit'**
  - **Event: 'uncaughtException'**
  - **Signal Events**

```
var eventHandler = function(){  
  console.log('EXIT');  
};
```

```
process.on('exit',eventHandler); // 오탈자 방지를 위해 더 선호하는 메서드  
//process.addListener('exit',eventHandler);
```

Console Debug Markdown HTML Preview Problem

<terminated> HelloWorld-events.js [Node Application] Node.js Process

EXIT

# events object

- 동일한 이벤트 10개 초과 등록시 에러발생
  - setMaxListener(n)으로 조정가능
  - process.setMaxListeners(15);
    - 15개까지 이벤트 등록 가능. `process.setMaxListeners(0);`
    - '0'입력: 제한없이 등록 가능 `process.on('exit',function() {})`  
`process.on('exit',function() {})`

[illegible]

# events object

```
process.on('exit',function(){
    console.log("잘가!");
});

var errorHandler1 = function(e){
    console.log("첫번째 에러",e);
};

var errorHandler2 = function(e){
    console.log("두번째 에러",e);
};

process.on('uncaughtException',errorHandler1);
process.on('uncaughtException',errorHandler2);

errorTest1
```

첫번째 에러 [ReferenceError: errorTest1 is not defined]  
두번째 에러 [ReferenceError: errorTest1 is not defined]  
잘가!

# events object

- Event의 삭제

- removeListener(), removeAllListener()

```
process.on('exit',function(){
  console.log("잘가!");
});

var errorHandler1 = function(e){
  console.log("첫번째 에러",e);
};

var errorHandler2 = function(e){
  console.log("두번째 에러",e);
};

process.on('uncaughtException',errorHandler1);
process.on('uncaughtException',errorHandler2);

process.removeListener('uncaughtException',errorHandler2);
```

errorTest1|

첫번째 에러 [ReferenceError: errorTest1 is not defined]  
잘가!

**removeListener(eventName, handler)**

특정 이벤트의 이벤트 리스너 제거

```
process.on('exit',function(){
  console.log("잘가!");
});

var errorHandler1 = function(e){
  console.log("첫번째 에러",e);
};

var errorHandler2 = function(e){
  console.log("두번째 에러",e);
};

process.on('uncaughtException',errorHandler1);
process.on('uncaughtException',errorHandler2);

process.removeAllListeners('uncaughtException');
```

errorTest1

잘가!

C:\Node.js\_IDE\eclipse\workspace\HelloWorld\events.js:31

errorTest1

^

ReferenceError: errorTest1 is not defined

at Object.<anonymous> (C:\Node.js\_IDE\eclipse\workspace\HelloWorld\events.js:31:1)

at Module.\_compile (module.js:460:26)

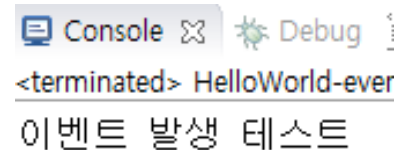
**removeAllListeners([eventName])**

모든 이벤트 리스너를 제거

# events object

- Event의 발생:
  - **emit(event, [arg1], [arg2], ...)**

```
process.on('test',function(){  
    console.log('이벤트 발생 테스트');  
});  
  
process.emit('test');
```

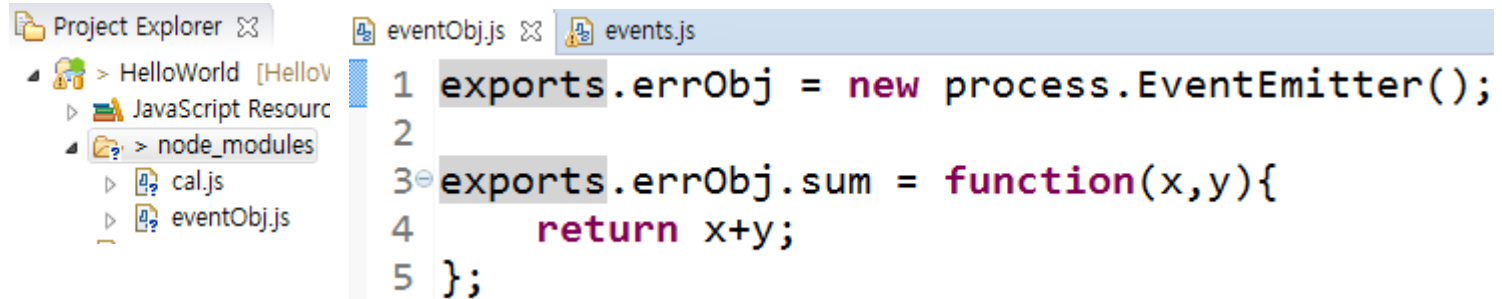


The screenshot shows a Node.js console window with two tabs: 'Console' and 'Debug'. The 'Console' tab is active, displaying the output of the code. The first line shows '<terminated> HelloWorld-ever' and the second line shows '이벤트 발생 테스트'.



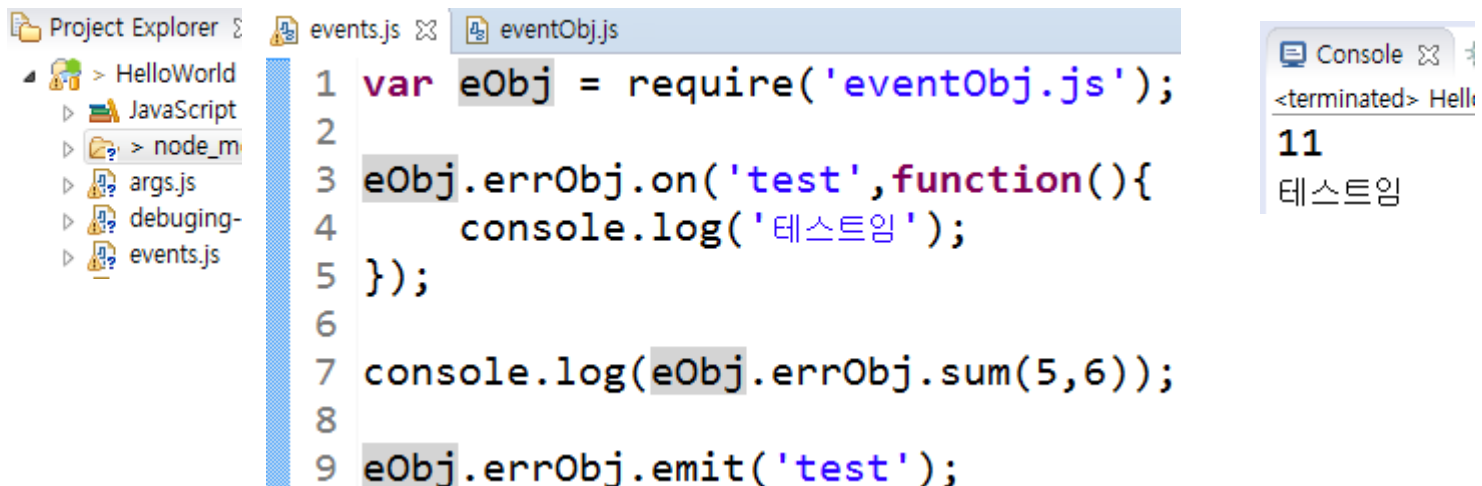
# events object

- 사용자 정의 event 객체
  - EventEmitter 상속



This screenshot shows the VS Code interface with the Project Explorer on the left and the editor on the right. The Project Explorer shows a project named 'HelloWorld' with a folder 'JavaScript Resources' containing a sub-folder 'node\_modules' with files 'cal.js' and 'eventObj.js'. The editor has two tabs: 'eventObj.js' and 'events.js', with 'eventObj.js' active. The code in 'eventObj.js' is as follows:

```
1 exports.errObj = new process.EventEmitter();
2
3 exports.errObj.sum = function(x,y){
4     return x+y;
5 };
```



This screenshot shows the VS Code interface with the Project Explorer on the left and the editor on the right. The Project Explorer shows the same project structure as the previous screenshot, but with an additional file 'events.js' in the 'node\_modules' folder. The editor has three tabs: 'events.js', 'eventObj.js', and 'eventObj.js', with 'events.js' active. The code in 'events.js' is as follows:

```
1 var eObj = require('eventObj.js');
2
3 eObj.errObj.on('test', function(){
4     console.log('테스트임');
5 });
6
7 console.log(eObj.errObj.sum(5,6));
8
9 eObj.errObj.emit('test');
```

On the right side of the editor, there is a 'Console' panel showing the output of the code execution:

```
<terminated> Hello
11
테스트임
```

# os module

- 서버의 기본적인 하드웨어 정보 확인

```
• os
  ◦ os.tmpdir()
  ◦ os.endianness()
  ◦ os.hostname()
  ◦ os.type()
  ◦ os.platform()
  ◦ os.arch()
  ◦ os.release()
  ◦ os.uptime()
  ◦ os.loadavg()
  ◦ os.totalmem()
  ◦ os.freemem()
  ◦ os.cpus()
  ◦ os.networkInterfaces()
  ◦ os.EOL

var os = require('os');

console.log(os.tmpdir());
console.log(os.endianness());
console.log(os.hostname());
console.log(os.type());
console.log(os.platform());
console.log(os.arch());
console.log(os.release());
console.log(os.uptime());
console.log(os.loadavg());
console.log(os.totalmem());
console.log(os.freemem());
console.log(os.cpus());
console.log(os.networkInterfaces());
console.log(os.EOL);

//서버의 temp 디렉토리 (임시 저장폴더 위치)
//endian type (BE or LE)
//서버의 host name
//서버의 OS type
//서버의 platform
//서버 CPU architecture
//서버 OS의 version
//서버 OS의 기동했던 때의 시간
//load average에 담긴 정보
//시스템 메모리
//사용가능한 메모리
//CPU정보
//네트워크 정보
//OS에 따른 End of Line 기호
```

<https://ko.wikipedia.org/wiki/엔디언>  
<https://nodejs.org/api/os.html>

```

C:\Users\LecPC\AppData\Local\Temp
LE
LecPC-PC
Windows_NT
win32
ia32
6.1.7601
25142.6218543
[ 0, 0, 0 ]
3757629440
2539438080
[ { model: 'Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz',
  speed: 3392,
  times: { user: 35250, nice: 0, sys: 68625, idle: 24925625, irq: 28656 } },
  { model: 'Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz',
  speed: 3393,
  times: { user: 75875, nice: 0, sys: 41015, idle: 24912218, irq: 5625 } },
  { model: 'Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz',
  speed: 3393,
  times: { user: 55218, nice: 0, sys: 87515, idle: 24886343, irq: 10312 } },
  { model: 'Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz',
  speed: 3393,
  times: { user: 53421, nice: 0, sys: 54718, idle: 24920828, irq: 8437 } } ]
{ 'ë|i»- ìì- ì°ê²°':
  [ { address: 'fe80::f101:790c:dbc8:52d8',
    netmask: 'ffff:ffff:ffff:ffff::',
    family: 'IPv6',
    mac: '08:00:27:19:42:07',
    scopeid: 11,
    internal: false },
    { address: '10.0.2.15',
    netmask: '255.255.255.0',
    family: 'IPv4',
    mac: '08:00:27:19:42:07',
    internal: false } ],
  'Loopback Pseudo-Interface 1':
  [ { address: '::1',
    netmask: 'ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff',
    family: 'IPv6',
    mac: '00:00:00:00:00:00',
    scopeid: 0,
    internal: true },
    { address: '127.0.0.1',
    netmask: '255.0.0.0',
    family: 'IPv4',
    mac: '00:00:00:00:00:00',
    internal: true } ] }

```

# File System module

- 파일읽기 , 파일확인 ,파일쓰기등을 하기 위한 모듈
  - 비동기 방식: 파일을 읽는 동안 다른 작업가능
  - 동기 방식: 파일을 읽는 동안 다른 작업 불가
    - 비동기방식메서드명+Sync

```
var fs = require('fs');
```

<https://nodejs.org/api/fs.html>

# File System module

- 파일 읽기

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

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

```
var fs = require('fs');
```

```
// 비동기방식
```

```
fs.readFile('test.txt', 'utf8', function(err, data){  
    if(err) throw err;  
    console.log(data);  
});
```

내 마음 별과 같이  
떠 올라 잠이 온다.

```
var fs = require('fs');
```

```
// 동기 방식
```

```
var data = fs.readFileSync('test.txt', 'utf8');  
console.log(data);
```

# File System module

- 파일 확인
  - fs.exists(path, callback);
  - fs.existsSync(path);

```
var fs = require('fs');

//비동기 방식
fs.exists('test1.txt',function(isFlag){
    console.log('fs.exists : ',isFlag);
});

//동기방식
var isFlag = fs.existsSync('test.txt');
console.log('fs.existsSync : ',isFlag);
```

Console ✕ Debug Markdown HTI  
<terminated> HelloWorld-fsTest.js [Node Applic  
fs.existsSync : true  
fs.exists : false

# File System module

- File Write

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





```
var fs = require('fs');
```

```
//비동기 방식
```

```
fs.writeFile('msg.txt',"안녕 노드야!","utf8",function(err){  
    if (err) throw err;  
    console.log("저장되었음");  
});
```

```
//동기방식
```

```
fs.writeFileSync('msg1.txt',"안녕 노드야!","utf8');
```

- ▷  hw-svr.js
- ▷  msg.txt
- ▷  msg1.txt
- ▷  osTest.js

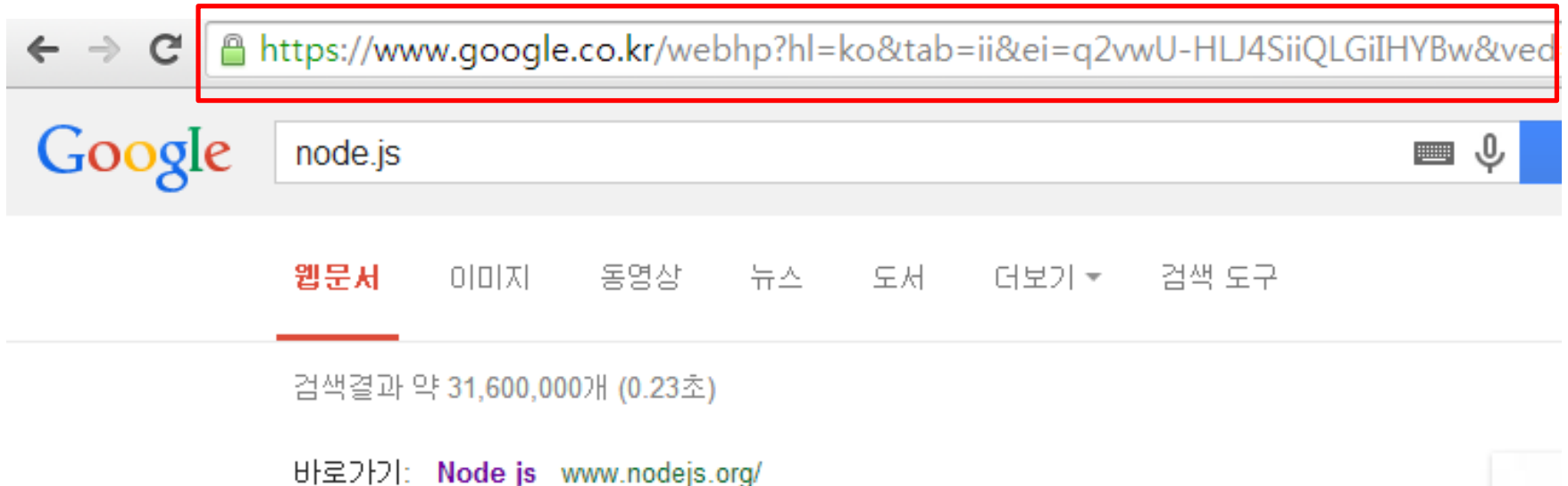
Console

<terminated> He

저장되었음

# url module

- url: uniform resource locator
  - network 상에서 resource가 어디 있는지를 알려주기 위한 규약
  - <https://nodejs.org/api/url.html>





# url module

- URL

- `url.parse(urlStr[, parseQueryString][, slashesDenoteHost])`
- `url.format(urlObj)`
- `url.resolve(from, to)`

- url string을 객체화: **parse( )**

- url 객체의 직렬화(string화): **format( )**

```
var url = require('url');

var urlObj = url.parse('https://www.google.co.kr/webhp'+
    '?sourceid=chrome-instant&ion=1&espv=2'+
    '&ie=UTF-8#newwindow=1&'+
    'q=node.js+%EA%B0%95%EC%A2%8C');
console.log('url string to Object: ', urlObj);
console.log('Object to url string: ', url.format(urlObj));
```

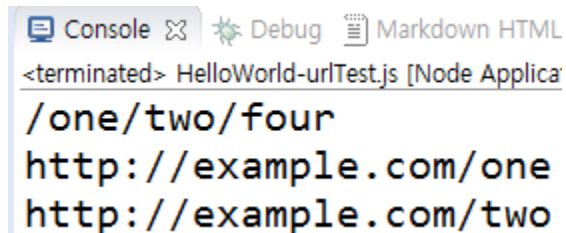
```
url string to Object: { protocol: 'https:',
  slashes: true,
  auth: null,
  host: 'www.google.co.kr',
  port: null,
  hostname: 'www.google.co.kr',
  hash: '#newwindow=1&q=node.js+%EA%B0%95%EC%A2%8C',
  search: '?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8',
  query: 'sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8',
  pathname: '/webhp',
  path: '/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8',
  href: 'https://www.google.co.kr/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#newwindow=1&q=node.js+%EA%B0%95%EC%A2%8C' }
Object to url string: https://www.google.co.kr/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#newwindow=1&q=node.js+%EA%B0%95%EC%A2%8C
```

# url module

- resolve( ): base url + path->동적 url 작성

```
var url = require('url');
```

```
console.log(url.resolve('/one/two/three', 'four'));  
console.log(url.resolve('http://example.com/', '/one'));  
console.log(url.resolve('http://example.com/one', '/two'));
```



The screenshot shows a web browser's developer console with three tabs: 'Console', 'Debug', and 'Markdown HTML'. The 'Console' tab is active, displaying the output of the code. The text in the console is as follows:

```
<terminated> HelloWorld-urlTest.js [Node Applica  
/one/two/four  
http://example.com/one  
http://example.com/two
```

# util module

- Utility software

<https://nodejs.org/api/util.html>

- 필수적이지 않지만, 일부 특정 작업을 수행

- util

- `util.debuglog(section)`
- `util.format(format[, ...])`
- `util.log(string)`
- `util.inspect(object[, options])`
  - Customizing `util.inspect` colors
  - Custom `inspect()` function on Objects
- `util.isArray(object)`
- `util.isRegExp(object)`
- `util.isDate(object)`
- `util.isError(object)`
- `util.inherits(constructor, superConstructor)`
- `util.deprecate(function, string)`
- `util.debug(string)`
- `util.error([...])`
- `util.puts([...])`
- `util.print([...])`
- `util.pump(readableStream, writableStream[, callback])`

# util module

```
var util = require('util');

var data = util.format('%s, %d, %j, %',
                        '문자열입력예', 0691,
                        {name: 'node.js예제'}, '테스트');

console.log(data);

util.log('로그가 시간과 함께 출력된다.');
```

Console | Debug | Markdown HTML Preview | Problems | Error Log

<terminated> HelloWorld-utilTest.js [Node Application] Node.js Process

문자열입력예, 691, {"name": "node.js예제"}, % 테스트  
14 Jul 20:36:15 - 로그가 시간과 함께 출력된다.

# net module

- InterNet

- 전세계 컴퓨터를 TCP/IP를 이용하여 정보를 주고 받을 수 있도록 한 컴퓨터 네트워크

5	응용 계층	DNS, TFTP, TLS/SSL, FTP, HTTP, IMAP, IRC, ECHO, 비트토렌트, RTP, PNRP, rlogin, ENRP,
4	전송 계층	TCP, UDP, DCCP, SCTP, IL, RUDP, ...
3	인터넷 계층	IP (IPv4, IPv6)
2.5	ARP	ARP, RARP
1,2	네트워크 인터페이스 계층	이더넷, Wi-Fi, 토큰링, PPP, SLIP, FDDI, ATM, .

# net module

- <https://nodejs.org/api/net.html>

- **net**

- `net.createServer([options][, connectionListener])`
- `net.connect(options[, connectionListener])`
- `net.createConnection(options[, connectionListener])`
- `net.connect(port[, host][, connectListener])`
- `net.createConnection(port[, host][, connectListener])`
- `net.connect(path[, connectListener])`
- `net.createConnection(path[, connectListener])`

- **Class: net.Server**

- `server.listen(port[, host][, backlog][, callback])`
- `server.listen(path[, callback])`
- `server.listen(handle[, callback])`
- `server.listen(options[, callback])`
- `server.close([callback])`
- `server.address()`
- `server.unref()`
- `server.ref()`
- `server.maxConnections`
- `server.connections`
- `server.getConnections(callback)`
- Event: 'listening'
- Event: 'connection'
- Event: 'close'
- Event: 'error'

# net module

- Class: `net.Socket`

- `new net.Socket([options])`
- `socket.connect(port[, host][, connectListener])`
- `socket.connect(path[, connectListener])`
- `socket.bufferSize`
- `socket.setEncoding([encoding])`
- `socket.write(data[, encoding][, callback])`
- `socket.end([data][, encoding])`
- `socket.destroy()`
- `socket.pause()`
- `socket.resume()`
- `socket.setTimeout(timeout[, callback])`
- `socket.setNoDelay([noDelay])`
- `socket.setKeepAlive([enable][, initialDelay])`
- `socket.address()`
- `socket.unref()`
- `socket.ref()`
- `socket.remoteAddress`
- `socket.remoteFamily`
- `socket.remotePort`
- `socket.localAddress`
- `socket.localPort`
- `socket.bytesRead`
- `socket.bytesWritten`
- Event: 'lookup'
- Event: 'connect'
- Event: 'data'
- Event: 'end'
- Event: 'timeout'
- Event: 'drain'
- Event: 'error'
- Event: 'close'

- `net.isIP(input)`
- `net.isIPv4(input)`
- `net.isIPv6(input)`

# net module

```
var net = require('net');

//서버의 생성
var server = net.createServer(function(con){
  console.log('클라이언트에서 서버로 연결됨');

  //서버 종료될 때 메시지
  con.on('end',function(){
    console.log('클라이언트와 서버 접속 종료됨');
  });

  //서버 접속시 클라이언트에 전송되어 표시되는 메시지
  con.write('Hi!\r\nThis my Telnet Service' +
    ' written by Node.js!\r\n');
  con.pipe(con);
});

//서버의 시작
server.listen(9630, function(){
  console.log('서버 시작됨');
});
```

Console ⌵ Debug Markdown I

HelloWorld-netTest.js [Node Application] No

서버 시작됨

클라이언트에서 서버로 연결됨

클라이언트와 서버 접속 종료됨



# net module

- 시작 > 제어판 > 프로그램 > Window 기능 사용/사용안함
  - 맨 아래 텔넷 클라이언트 체크

