[全局对象和属性 2](#_Toc391203101)

[process 2](#_Toc391203102)

[全局属性\_\_filename和\_\_dirname 2](#_Toc391203103)

[获得请求内容 3](#_Toc391203104)

[调试Node.js 4](#_Toc391203105)

[模块学习 5](#_Toc391203106)

[Path 5](#_Toc391203107)

[FileSystem 6](#_Toc391203108)

# 全局对象和属性

## process

process模块允许你获得或者修改当前node进程的设置，不想其他的模块，process是一个全局进程，你可以直接通过process变量直接访问它。

process.exit(); 退出当前node进程

## 全局属性\_\_filename和\_\_dirname

**\_\_filename**

The filename of the code being executed. This is the resolved absolute path of this code file. For a main program this is not necessarily the same filename used in the command line. The value inside a module is the path to that module file.

Example: running node example.js from /Users/mjr

console.log(\_\_filename);

// /Users/mjr/example.js

\_\_filename isn't actually a global but rather local to each module.

**\_\_dirname**

The name of the directory that the currently executing script resides in.

Example: running node example.js from /Users/mjr

console.log(\_\_dirname);

// /Users/mjr

\_\_dirname isn't actually a global but rather local to each module.

这大概的意思就是，\_\_filename指的是当前解析文件的实际物理地址，而\_\_dirname指的是此文件在被执行时所在的地址

　　如test.js文件在C:/目录中：

1 var text = require("./test2");

2 console.log("the test.js's filename is: %s",\_\_filename);

3 console.log("the test.js's dirname is %s",\_\_dirname);

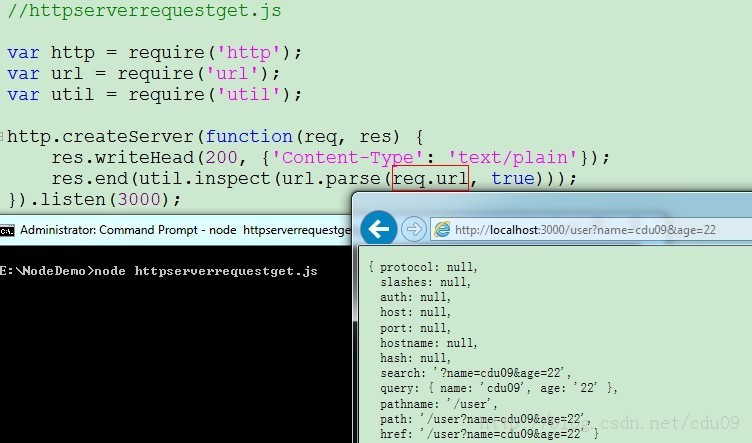
4 text.start();

则test.js中的\_\_filename为C:/test.js，\_\_dirname为C:/

# 获得请求内容

1. 获取GET请求内容

可以手动解析后面的内容作为GET请求的参数。Node.js的url模块中的parse函数提供了这个功能，如下图：通过url.parse，原始的path被解析为一个对象。



1. 获取POST请求内容

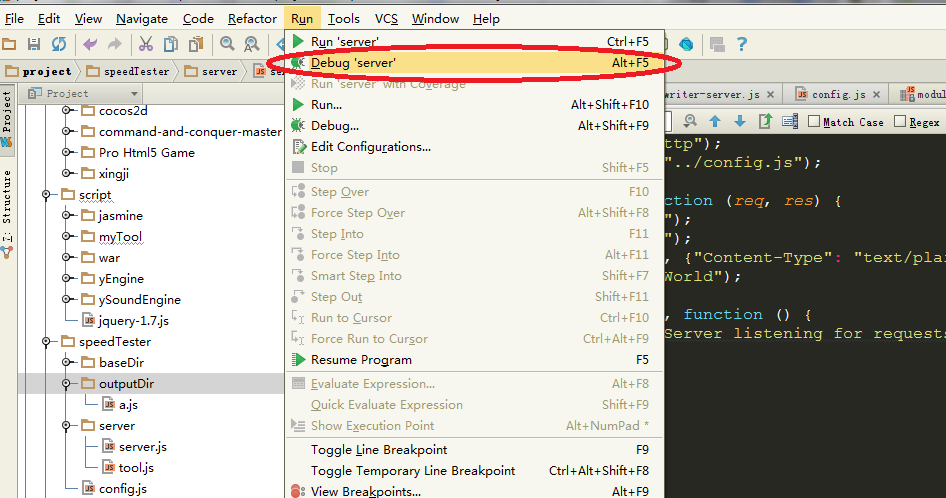
POST请求的内容全部都在请求体中，Node.js默认是不会解析请求体的，实现方法如下：并没有在请求响应函数中向客户端返回信息，而是定义了一个post变量，用于在闭包中暂存请求体的信息。通过req的data事件监听函数，每当接受到请求体的数据，就累加到post变量中。在end事件触发后，通过querystring.parse将post解析为真正的POST请求格式，然后向客户端返回。注意：不要在真正的生产应用中使用上面这种简单的方法来获取POST请求，因为它有严重的效率问题和安全问题。

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/cdu09/article/details/11006361)

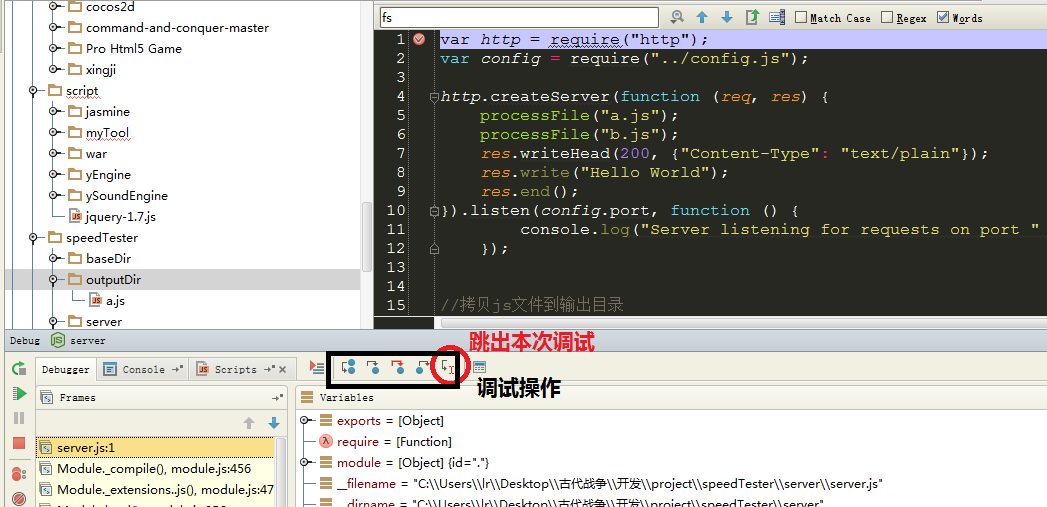
1. **var** http = require('http');
2. **var** querystring = require('querystring');
3. **var** util = require('util');
5. http.createServer(**function**(req, res) {
6. **var** post = '';
7. req.on('data', **function**(chunk) {
8. post += chunk;
9. });
10. req.on('end', **function**() {
11. post = querystring.parse(post);
12. res.end(util.inspect(post));
13. });
14. }).listen(3000);

# 调试Node.js

run->debug开始调试node.js文件（会run文件，因此需要先关闭已经run的文件）：



可直接在node.js文件中加入断点：



# 模块学习

Request

Http

Response

## Path

1. path.join([path1], [path2], [...])

Join all arguments together and normalize the resulting path.

Arguments must be strings. In v0.8, non-string arguments were silently ignored. In v0.10 and up, an exception is thrown.

Example:

path.join('/foo', 'bar', 'baz/asdf', 'quux', '..')

// returns

'/foo/bar/baz/asdf'

path.join('foo', {}, 'bar')

// throws exception

TypeError: Arguments to path.join must be strings

1. path.normalize(p)

Normalize a string path, taking care of '..' and '.' parts.

When multiple slashes are found, they're replaced by a single one; when the path contains a trailing slash, it is preserved. On Windows backslashes are used.

Example:

path.normalize('/foo/bar//baz/asdf/quux/..')

// returns

'/foo/bar/baz/asdf'

## FileSystem

1. Class: fs.Stats[#](http://nodejs.org/api/fs.html#fs_class_fs_stats)

Objects returned from fs.stat(), fs.lstat() and fs.fstat() and their synchronous counterparts are of this type.

* stats.isFile()
* stats.isDirectory()
* stats.isBlockDevice()
* stats.isCharacterDevice()
* stats.isSymbolicLink() (only valid with fs.lstat())
* stats.isFIFO()
* stats.isSocket()

For a regular file util.inspect(stats) would return a string very similar to this:

{ dev: 2114,

ino: 48064969,

mode: 33188,

nlink: 1,

uid: 85,

gid: 100,

rdev: 0,

size: 527,

blksize: 4096,

blocks: 8,

atime: Mon, 10 Oct 2011 23:24:11 GMT,

mtime: Mon, 10 Oct 2011 23:24:11 GMT,

ctime: Mon, 10 Oct 2011 23:24:11 GMT }

Please note that atime, mtime and ctime are instances of [Date](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Date) object and to compare the values of these objects you should use appropriate methods. For most general uses [getTime()](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Date/getTime) will return the number of milliseconds elapsed since *1 January 1970 00:00:00 UTC* and this integer should be sufficient for any comparison, however there additional methods which can be used for displaying fuzzy information. More details can be found in the [MDN JavaScript Reference](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Date) page.

1. 删除文件夹

rmdir只能删除空文件夹，不能删除非空文件夹。

删除文件——

fs.unlink(path[,callback])或者fs.unlinkSync(path)

删除文件夹——

方法1：使用递归

deleteFolderRecursive = function(path) {

    var files = [];

    if( fs.existsSync(path) ) {

        files = fs.readdirSync(path);

        files.forEach(function(file,index){

            var curPath = path + "/" + file;

            if(fs.statSync(curPath).isDirectory()) { // recurse

                deleteFolderRecursive(curPath);

            } else { // delete file

                fs.unlinkSync(curPath);

            }

        });

        fs.rmdirSync(path);

    }

};

方法2：使用系统的命令

var exec = require('child\_process').exec,child;

child = exec('rm -rf test',function(err,out) {

  console.log(out); err && console.log(err);

});