模块

2020年8月2日

21:36

**模块形式**

(function(exports, require, module, \_\_filename, \_\_dirname) {

// Module code actually lives in here

'use strict'

function hello() {

console.log('Hello from ' + \_\_filename + '!');

}

module.exports.hello = hello;

});

**\_\_filename**:模块的JavaSrcipt源文件的标准名称，此名称由Node随其他几个参数传递到函数包装程序

**exports**:module.exports的简短表示形式

require:包含其他模块的函数

module:对当前模块的引用

\_\_dirname:模块的JavaScript源文件所在目录的完整路径

**常用模块**

**文件处理模块: file-processor**

-processFileSync()

-processFile() 函数

**消息日志记录模块: simple-logger**

* 四个内部组件

-日志级别: 由Level对象实现

-当前日志级别: 默认情况下设置为INFO,但可由程序进行修改

-log()函数: 选择性记录到控制台的内部实现

-助手函数: 如记录参考消息的info(),对调试消息进行更精细调整的debug(),用于警告的warn()

* 修改当前日志阈值: setLogLevel(newLevel)
* log(messageLogLevel,message,startTime)
* Trace(message,startTime),debug(message,startTime),info(message,startTime),warn(message,startTime),error(message,startTime)

**V8概要分析器 :**

生成日志文件:　node --prof --no-logfile-per-isolate --log --logfile=example3.log example3

分析日志文件:　node --prof-process example3.log > example3.log.txt

**http模块**

const http = require('http');

const server = http.createServer((request,response)=>{}).listen(PORT);

**负载测试分析**

Apache Bench

loadtest

* 安装: npm install loadtest -g
* 测试: loadtest -n 100 -c 10 -m POST <http://localhost:8080/sync>
  + n:总共请求数量
  + c:同时请求数量
  + m:请求方法

**nodemon**: nodemon可以监听源文件的变化,从而重新启动源文件

安装: npm install nodemon -g

使用:nodemon example.js

金字塔厄运