Node基础编程

1. 基于Chrome V8引擎，可处理高并发
2. 单线程
3. 支持js语法开发后端代码
4. 非阻塞的IO：基于事件驱动

# Common规范

定义：一个文件代表一个模块，导出，引用

node.js模块：内置模块，第三方、自定义模块

内置模块载入：https://nodejs.org/dist/latest-v12.x/docs/api/

const os = require('os');

console.log(os.hostname())

第三方模块：例request模块发送http请求

需要先安装(npm)：npm instrall –save request

发起get请求



发送post请求



form格式的post请求：



暴露接口：自定义模块

创建demo.js文件

写个方法:demo1

引出：export demo1 = demo1 or module.exports = hello

引入：const demo = require(./demo.js)

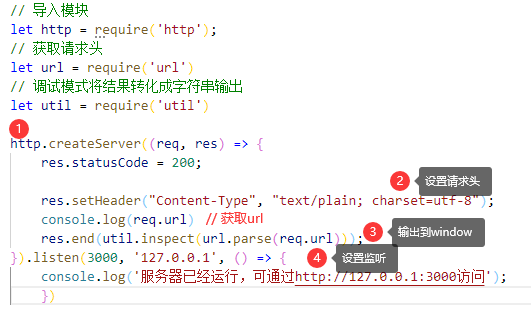
console.log(demo.demo1())

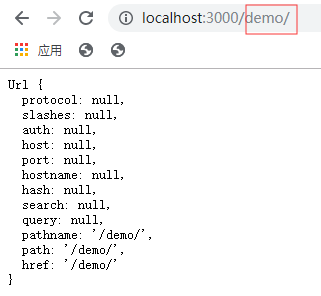
**exports 和 module.exports 的使用**

如果要对外暴露属性或方法，就用 ***exports*** 就行，要暴露对象(类似class，包含了很多属性和方法)，就用 ***module.exports***。

引用：

# 创建http server





# 创建一个web容器，可以访问到HTML内容

# Node.js常用模块介绍

## url

### url.parse(urlStr, [parseQueryString], [slashesDenoteHost])

url 模块中的 parse 函数可以用于解析url中的参数。

参数详解：

**urlStr** - 需要接收的url字符串。

**parseQueryString** - 为true时将使用查询模块分析查询字符串，默认为false。

主要针对query中的内容分析

**shashesDenoteHost**

         -默认为false，//foo/bar 形式的字符串将被解释成 { pathname: ‘//foo/bar' }

         -如果设置成true，//foo/bar 形式的字符串将被解释成  { host: ‘foo', pathname: ‘/bar' }

例：解析博客地址https://www.cnblogs.com/1314-/p/6517740.html

const url = require('url')

const myURL =  url.parse('https://www.baidu.com/s?tn=80035161\_2\_dg&wd=%E6%98%9F%E7%A9%BA')

console.log(myURL)

**parseQueryString** 为false时：

query: 'tn=80035161\_2\_dg&wd=%E6%98%9F%E7%A9%BA',



## querystring（查询字符串）

**语法：querystring.stringify(obj[, sep[, eq[, options]]])**

提供用于解析和格式化 URL 查询字符串的实用工具，一般是对**http请求所带的数据**进行解析

调用：const querystring = require('querystring');

方法：

### querystring.parse(str,separator,eq,options)

parse这个方法是将一个字符串反序列化为一个**对象**

separator：默认分隔符为&

eq: 默认分割=，用于划分键值对

options： 默认该参数是对象



**[Object: null prototype] { name: 'whitemu', sex: [ 'man', 'women' ] }**

### querystring.stringify(obj,separator,eq,options)

将一个对象序列化成一个**字符串**，与querystring.parse相对。

var b = querystring.stringify({ name: 'whitemu', sex: [ 'man', 'women' ]})

console.log(b)

**name=whitemu&sex=man&sex=women**

### querystring.escape(str)

escape可使传入的字符串进行**编码**

var c = querystring.escape('name=唐高峰')

console.log(c)

name%3D%E5%94%90%E9%AB%98%E5%B3%B0

### querystring.unescape(str)

unescape方法可将**含有%的字符串**进行**解码**

demoUrl = 'https://www.baidu.com:8080/s?tn=80035161\_2\_dg&wd=%E6%98%9F%E7%A9%BA';

var d = querystring.unescape(demoUrl)

console.log(d)

https://www.baidu.com:8080/s?tn=80035161\_2\_dg&wd=星空

## HTTP模块

调用：const http = require('http')



createServer方法本质上是为http.Server对象添加了一个request事件监听

var server=new http.Server();

server.on("request", (req,res) => {}).listen(3000);

req是请求消息，res是响应消息

req是http发送请求后返回的一个对象，req.query里面有你传递的数据

res.on等同于监听事件addListener

res.end; res 是个writable的流，当writable流接收到 null 的时候会触发 end 事件这里的确切含义就是给客户端的东西写完了，可以发送给客户端了

### http.request(GET & POST)

## Events

## fs

## Stream

Stream 是一个抽象接口，Node 中有很多对象实现了这个接口。例如，对http 服务器发起请求的request 对象就是一个 Stream，还有stdout（标准输出）。

Stream 有四种流类型：

* **Readable** - 可读操作。
* **Writable** - 可写操作。
* **Duplex** - 可读可写操作.
* **Transform** - 操作被写入数据，然后读出结果。

所有的 Stream 对象都是 EventEmitter 的实例。常用的事件有：

* **data** - 当有数据可读时触发。
* **end** - 没有更多的数据可读时触发。
* **error** - 在接收和写入过程中发生错误时触发。
* **finish** - 所有数据已被写入到底层系统时触发。

## 后端路由

## 请求参数

## Events

## 同步异步编码async

## socket

## MongoDB

## express