

# NodeJs HTTP 模块、URL 模块、supervisor 工具

主讲教师：（大地）

合作网站：[www.itying.com](http://www.itying.com)

## 目录

一、Node.js 创建第一个应用 .....	1
二、HTTP 模块、URL 模块 .....	2
三、Nodejs 自启动工具 supervisor .....	6

## 一、Node.js 创建第一个应用

如果我们使用 PHP 来编写后端的代码时，需要 Apache 或者 Nginx 的 HTTP 服务器，来处理客户端的请求相应。不过对 Node.js 来说，概念完全不一样了。使用 Node.js 时，我们不仅仅在实现一个应用，同时还实现了整个 HTTP 服务器。

### 1、引入 http 模块

```
var http = require("http");
```

### 2、创建服务器

接下来我们使用 `http.createServer()` 方法创建服务器，并使用 `listen` 方法绑定 8888 端口。函数通过 `request, response` 参数来接收和响应数据。

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

http.createServer(function (request, response) {

    // 发送 HTTP 头部
    // HTTP 状态值: 200 : OK
    // 设置 HTTP 头部, 状态码是 200, 文件类型是 html, 字符集是 utf8
    response.writeHead(200,{ 'Content-Type': 'text/html;charset=UTF-8' });

    // 发送响应数据 "Hello World"
    res.end("哈哈哈哈哈, 我买了一个 iPhone" + (1+2+3) + "s");

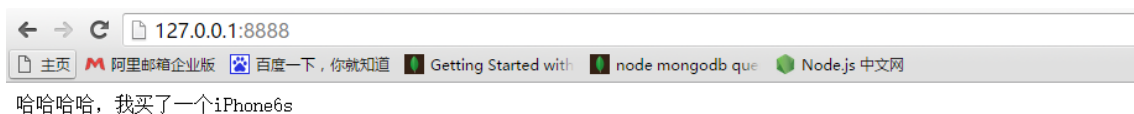
}).listen(8888);
// 终端打印如下信息
console.log('Server running at http://127.0.0.1:8888/');
```

### 3. 运行程序

用命令行切换到程序对应目录。通过 `node` 命令运行程序。

```
C:\Users\Administrator>cd C:\nodejs
C:\nodejs>node server.js
```

浏览器运行



哈哈哈哈哈, 我买了一个iPhone6s

你会发现, 我们本地写一个 `js`, 打死都不能直接拖入浏览器运行, 但是有了 `node`, 我们任何一个 `js` 文件, 都可以通过 `node` 来运行。也就是说, `node` 就是一个 `js` 的执行环境。

## 二、HTTP 模块、URL 模块

`Node.js` 中, 将很多的功能, 划分为了一个个 `module` (模块)。 `Node.js` 中的很多功能都是通过模块实现。

## 2.1、HTTP 模块的使用

```
//引用模块
var http = require("http");

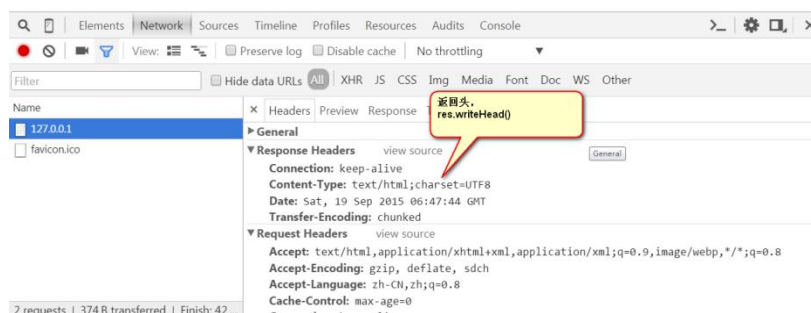
//创建一个服务器，回调函数表示接收到请求之后做的事情
var server = http.createServer(function(req,res){
    //req 参数表示请求，res 表示响应
    console.log("服务器接收到了请求" + req.url);
    res.end(); // End 方法使 Web 服务器停止处理脚本并返回当前结果
});
//监听端口
server.listen(3000,"127.0.0.1");
```

设置一个响应头：

```
res.writeHead(200,{ "Content-Type":"text/html;charset=UTF8"});
```



我是一个主标题



我们现在来看一下 req 里面能够使用的东西。

最关键的就是 req.url 属性,表示用户的请求 URL 地址。所有的路由设计,都是通过 req.url 来实现的。

我们比较关心的不是拿到 URL,而是识别这个 URL。

识别 URL,用到了下面的 url 模块

## 2.2、URL 模块的使用

url.parse() 解析 URL

url.format(urlObject) //是上面 url.parse() 操作的逆向操作

url.resolve(from, to) 添加或者替换地址

### 1、url.parse()

```
> url.parse('http://www.baidu.com')
Url {
  protocol: 'http:',
  slashes: true,
  auth: null,
  host: 'www.baidu.com',
  port: null,
  hostname: 'www.baidu.com',
  hash: null,
  search: null,
  query: null,
  pathname: '/',
  path: '/',
  href: 'http://www.baidu.com/' }
```

```
> url.parse('http://www.baidu.com?name=zhangsan&age=20')
Url {
  protocol: 'http:',
  slashes: true,
  auth: null,
  host: 'www.baidu.com',
  port: null,
  hostname: 'www.baidu.com',
  hash: null,
  search: '?name=zhangsan&age=20',
  query: 'name=zhangsan&age=20',
  pathname: '/',
  path: '/?name=zhangsan&age=20',
  href: 'http://www.baidu.com/?name=zhangsan&age=20' }
```

```
> url.parse('http://www.baidu.com?name=zhangsan&age=20', true)
Url {
  protocol: 'http:',
  slashes: true,
  auth: null,
  host: 'www.baidu.com',
  port: null,
  hostname: 'www.baidu.com',
  hash: null,
  search: '?name=zhangsan&age=20',
  query: { name: 'zhangsan', age: '20' },
  pathname: '/',
  path: '/?name=zhangsan&age=20',
  href: 'http://www.baidu.com/?name=zhangsan&age=20' }
```

```
href: 'http://www.baidu.com/?name=zhangsan&age=20' }
> url.parse('//foo/bar', true, true)
Url {
  protocol: null,
  slashes: true,
  auth: null,
  host: 'foo',
  port: null,
  hostname: 'foo',
  hash: null,
  search: '',
  query: {},
  pathname: '/bar',
  path: '/bar',
  href: '//foo/bar' }
> url.parse('//foo/bar', true)
Url {
  protocol: null,
  slashes: null,
  auth: null,
  host: null,
  port: null,
  hostname: null,
  hash: null,
  search: '',
  query: {},
  pathname: '//foo/bar',
  path: '//foo/bar',
  href: '//foo/bar' }
>
```

## 2、url.format()

```
> url.format({
...   protocol: 'http:',
...   slashes: true,
...   auth: null,
...   host: 'www.baidu.com:8010',
...   port: '8010',
...   hostname: 'www.baidu.com',
...   hash: null,
...   search: '?name=zhangsan&age=20',
...   query: { name: 'zhangsan', age: '20' },
...   pathname: '/',
...   path: '/?name=zhangsan&age=20',
...   href: 'http://www.baidu.com:8010/?name=zhangsan&age=20' })
'http://www.baidu.com:8010/?name=zhangsan&age=20'
```

### 3、url.resolve()

```
> url.resolve('http://example.com/', '/one')
'http://example.com/one'
>
```

## 三、Nodejs 自启动工具 supervisor

supervisor 会不停的 watch 你应用下面的所有文件，发现有文件被修改，就重新载入程序文件这样就实现了部署，修改了程序文件后马上就能看到变更后的结果。麻麻再也不用担心我的重启 nodejs 了！

### 1. 首先安装 supervisor

```
npm install -g supervisor
```

### 2. 使用 supervisor 代替 node 命令启动应用

```
C:\nodejs>supervisor server.js

Running node-supervisor with
  program 'server.js'
  --watch '.'
  --extensions 'node,js'
  --exec 'node'
```