

**Nodejs基础**

目录

[目录 2](#_Toc476205112)

[第1章 介绍 3](#_Toc476205113)

[1.1 宿主环境 3](#_Toc476205114)

[1.2 安装 4](#_Toc476205115)

[1.3 环境变量 5](#_Toc476205116)

[1.4 体验Nodejs 9](#_Toc476205117)

[第2章 基础 9](#_Toc476205118)

[2.1 基础语法 9](#_Toc476205119)

[2.2 ES6 10](#_Toc476205120)

[2.2.1 块级作用域 11](#_Toc476205121)

[2.2.2 let 关建字 11](#_Toc476205122)

[2.2.3 const关键字 11](#_Toc476205123)

[2.2.4 变量解构赋值 11](#_Toc476205124)

[2.2.5 字符串扩展 12](#_Toc476205125)

[2.2.6 数组扩展 12](#_Toc476205126)

[2.2.7 函数扩展 12](#_Toc476205127)

[第3章 模块 12](#_Toc476205128)

[3.1 系统模块 13](#_Toc476205129)

[3.2 文件模块 13](#_Toc476205130)

[3.2.1 第三方模块 13](#_Toc476205131)

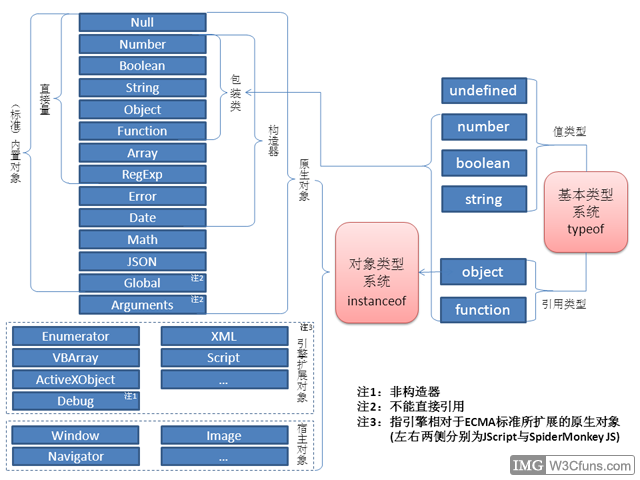
[3.2.2 用户模块 15](#_Toc476205133)

# 介绍

Node.js是一个基于Chrome V8引擎的JavaScript运行环境，其使用了一个事件驱动、非阻塞式 I/O的模型，使其轻量又高效。

## 宿主环境

以网景的Netscape Navigator内置的JavaScript 1.1为蓝本，由ECMA-262定义的 ECMAScript 是一种脚本语言标准，它对该脚本语言的语法、类型、语句、操作符等做出明确规定。



Javascript是由ECMAScript、DOM、BOM三部分组成，浏览器实现了ECMAScript标准，使其能够运行于浏览器，我们称浏览器为ECMAScript的宿主环境。

除了可以运行于浏览器环境外，ECMAScript还可以运行于Adobe Flash环境中（称为Action Script），这时Adobe Flash也是ECMAScript的宿主环境。

我们即将学习的Nodejs就是另一种可以运行ECMAScript的宿主环境。

## 安装

Nodejs是开源、跨平台（即可以运行在不同的操作系统上）的。

Windows平台，Nodejs就是一个以.msi结尾的软件，由于Windows操作系统分成32位和64位，所以在Nodejs在Windows上也分为[32位](https://nodejs.org/dist/v6.10.0/node-v6.10.0-x86.msi)和[64位](https://nodejs.org/dist/v6.10.0/node-v6.10.0-x64.msi)两个版本。

Linux平台，根据其发行版不同，其安装方式也不同，但都可以使用源码编译安装。

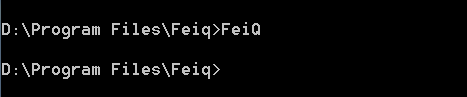
Mac平台，Nodejs是一个.pkg结尾的包，双击根据引导即可完成安装。

不论何种平台，检测Nodejs是否安装都可以通过命令 node -v，这个命令是执行结果会返回当前Nodejs的版本，以Windows平台为例，如下图：



## 环境变量

执行命令node -v时，其实是运行了一个名叫node.exe的可执行程序，操作系统自动查找并调用node.exe这个可执行程序，同样的我们也可以使用命令来执行其它可执行程序，以FeiQ.exe为例，如下图所示：

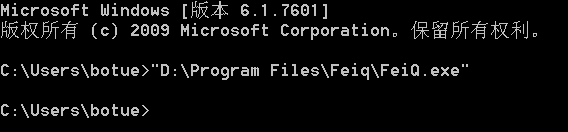


其实并不是在任意位置都可以执行feiq命令(windows平台不区分大小写)，如下图所示：



必须输入FeiQ.exe的绝对路径或将目录调整到FeiQ.exe所在目录才能被执行，

如下图所示：



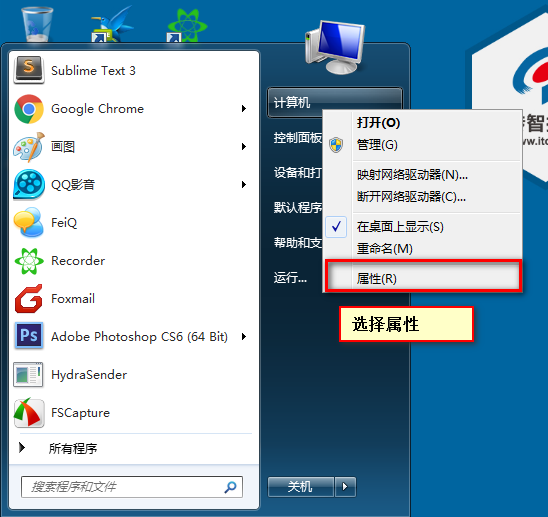
但是在执行node命令时却不需要，出现这种差别的原因在于环境变量。

环境[变量](http://baike.baidu.com/view/296689.htm)是在操作系统中一个具有特定名字的对象，它包含了一个或者多个应用[程序](http://baike.baidu.com/view/17674.htm)所将使用到的信息。

当在命令行窗口输入某个命令要求系统运行一个[程序](http://baike.baidu.com/view/17674.htm)而没有告诉它程序所在的完整路径时，操作系统会优先[当前目录](http://baike.baidu.com/view/396351.htm)下面查找此程序，找到则执行，找不到会到环境变量path中去找。

一般情况下安装Nodejs时，会自动设置path环境变量，所以安装完成立即执行node -v就能被执行，但是也有各别情况环境变量不能自动被添加，这时就需要手动添加。

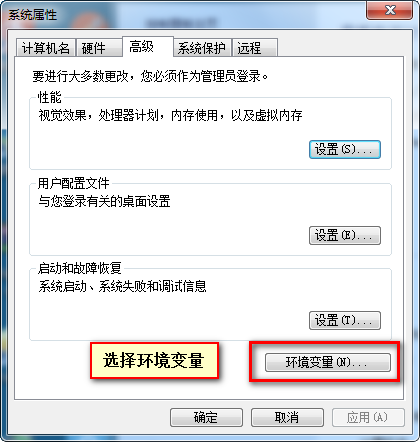
步骤1



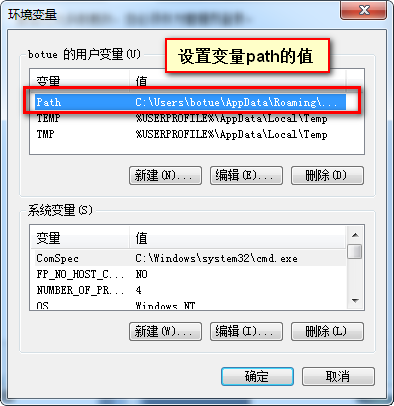
步骤2



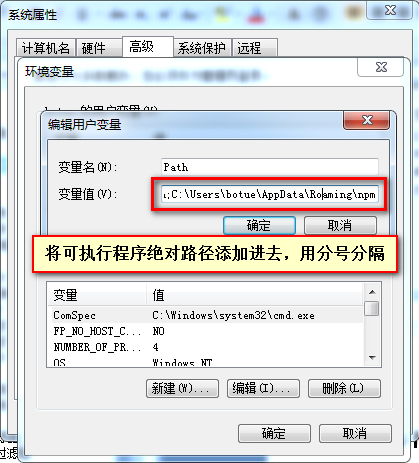
步骤3



步骤4

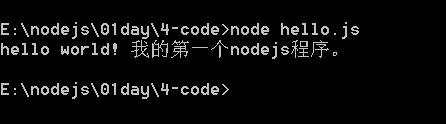


步骤5



## 体验Nodejs

宿主环境不同依据ECMAScript标准实现的脚本的执行方式也不同，例如Javascript必须通过script标签引入，然后在浏览器环境才能被执行，Nodejs则是需要通过命令行工具以命令的形式执行。



见示例代码 01 hello.js

# 基础

## 基础语法

Nodejs是ECMAScript的宿主环境，与浏览器中的Javascript一奶同胞，所以其基础语法是一致的，都遵循ECMAScript 的标准规范。

见示例代码 02 syntax.js

实践中不同的宿主环境又会在ECMAScript的基础上进行扩展，例如浏览器的javascript就扩展了window、location、navigator等对象，在Nodejs中则没有这些对象。

Nodejs也进行了扩展，主要体现在全局变量和全局方法两方面，如下：

1、require 用来加载并执行其它脚本文件，见示例代码 03 require.js

2、module用来导出脚本文件的属性或方法，详见模块章节

3、exports 用来导出脚本文件的属性或方法，详见模块章节

4、\_\_dirname 用来获取当前文件目录，见示例代码 04 \_\_dirname.js

5、\_\_filename 用来获取当前文件的绝对路径，见示例代码 05 \_\_filename.js

6、global 全局对象，类似于浏览器中的window。

a) setTimeout/clearTimeout

b) setInterval/clearInterval

c) console

见示例代码 06 global.js

## ES6

Javascript在设计之初只是用来做简单的数据较验，后来发展到可以做一些网页的动态效果用来增强用户体验，再后来ECMA组织负责组织制订javascript规范的制订，将其称为ECMAScript，伴随着互联网的迅速发展，对ECMAScript提出了更高的要求，为了适应开发的需要，ECMA组织也在不断的修改完善ECMAScript的标准（如ECMAScript6简称ES6）。

然而新的标准所定义的规范在浏览器中的javascript中并不能得到很好的应用，原因在于浏览器种类较多，开发者并不能约束用户必须使用何种浏览器，导致浏览器高低版本共存是常态，为了考虑兼容，开发者不得不放弃使用一些高级特性。

Nodejs则不会出现上述情况，Nodejs由统一的团队组织维护，通过版本升级方式可以及时跟进ECMAScript的标准，而且多数情况Nodejs是运行于服务端的，开发者可以决定使用Nodejs的版本，很多高级特性可以在Nodejs中得到很好的运用。

### 块级作用域

任何一对花括号（｛｝）中的语句集都属于一个块，在块中声明的变量在代码块外都是不可访问的，称之为块级作用域，ES5以前没有块级作用域，见示例代码 07 scope.js

### let 关建字

let用于声明变量（类似于var），其声明的变量只在当前代码块内有效，实际上是创建了块级作用域，见示例代码 08 let.js

### const关键字

const 用于声明常量，常量于变量不同，变量声明后其值可以任意更改，而常量一旦赋值就不允许被更改，见示例代码 09 const.js

### 变量解构赋值

按照某种模式，从数组和对象中提取值，对变量进行赋值这被称为解构。

a) 数组解构赋值，见示例代码 10 array-structure.js

b) 对象解构赋值，见代码代码 11 object-structure.js

以上介绍了解构赋值的语法规则，可以在现实开发中带来很大的便利，提升开发效率，见代码示例 12 case.js

### 字符串扩展

a) 对象方法，新增一些用于处理字符串的便捷方法，见示例代码 13 string-extends-1.js

b) 模板字符串，可以使用反引号（`）定义字符串，见示例代码 14 string-extends-2.js

### 数组扩展

见示例代码 15 array-extends.js

### 函数扩展

a) 默认参数，允许为函数指定默认参数，见示例代码 16 function-extends-1.js

b) reset参数，用于获取多余的参数，将多余的参数装到一个数组中，见示例代码 17 function-extends-2.js

c) 扩展运算符，它好比 rest 参数的逆运算，将一个数组转为用逗号分隔的参数序列，见示例代码 18 function-extends-3.js

d) 箭头函数，使用“箭头”(=>) 定义函数，见示例代码 19 function-extends.js

# 模块

Nodejs在ECMAScript的基础上扩展并封装了许多高级特性，如文件访问、网络访问等，使得Nodejs成为一个很好的Web开发平台。

基于Nodejs这个平台将Web开发常用的一些功能进行封装，称为模块。

## 系统模块

在安装Nodejs时一些模块自动被安装，这些模块大多由Nodejs官方维护，称这类模块为系统（核心）模块，如fs、path、os、http等。

模块被加载后才能使用，Nodejs提供了全局的函数require加载模块。

1、os模块 获取当前操作系统信息，见代码示例 20 os.js

2、fs模块 文件读写操作，见代码示例 21 fs.js

3、path模块 路径处理，见代码示例 22 path.js

## 文件模块

除了系统模块外，还有以文件形式存在的模块，这些模块由一些组织或个人根据不同的业务需求开发。

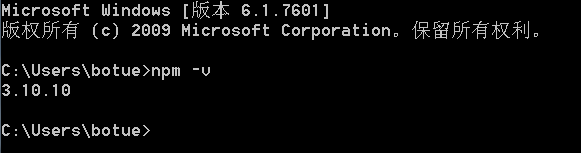
注：模块也被称为包。

### 第三方模块

到目前为止，第三方模块数量已经相当庞大，为了能够更好的管理这些模块，Nodejs官方提供了一个工具（npm）专门用于管理这些第三方模块。



npm (Node Package Manager) 这个工具随Nodejs一起被安装，可以使用 npm -v 检测是否安装，如下图



使用npm可以安装、卸载、发布模块。

1、安装

a) 本地安装，安将到当前目录下node\_modules中

npm install 模块名称

b) 全局安装，安装到默认c盘

npm install 模块名称 -g

2、卸载

a) 本地卸载

npm uninstall 模块名称

b) 全局卸载

npm uninstall 模块名称 -g

3、发布模块

a) 注册用户，注册[npm账号](https://www.npmjs.com/signup)

b) 编写模块

新建helloitcast.js，编写简单逻辑。

c) package.json，是一个用于描述当前模块信息的文件，例如模块名称、版本号、作者等，使用npm init进行根据提示创建。

c) 发布模块

在模块目录执行 npm login，根据提示输入用户名、密码、邮箱进行登录，然后执行npm publish完成发布。

### 用户模块