# Node.js基础介绍

## Node.js简介

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境

采用Google的V8引擎作为JavaScript语言解释器，通过自行开发的libuv库来调用操作系统资源

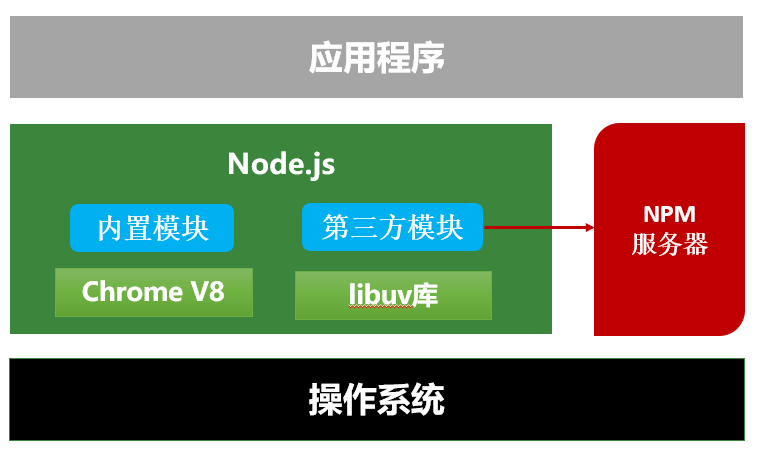
官网：<https://nodejs.org/en>

**实时 Web 应用开发平台：**用于方便地搭建响应速度快、易于扩展的网络应用

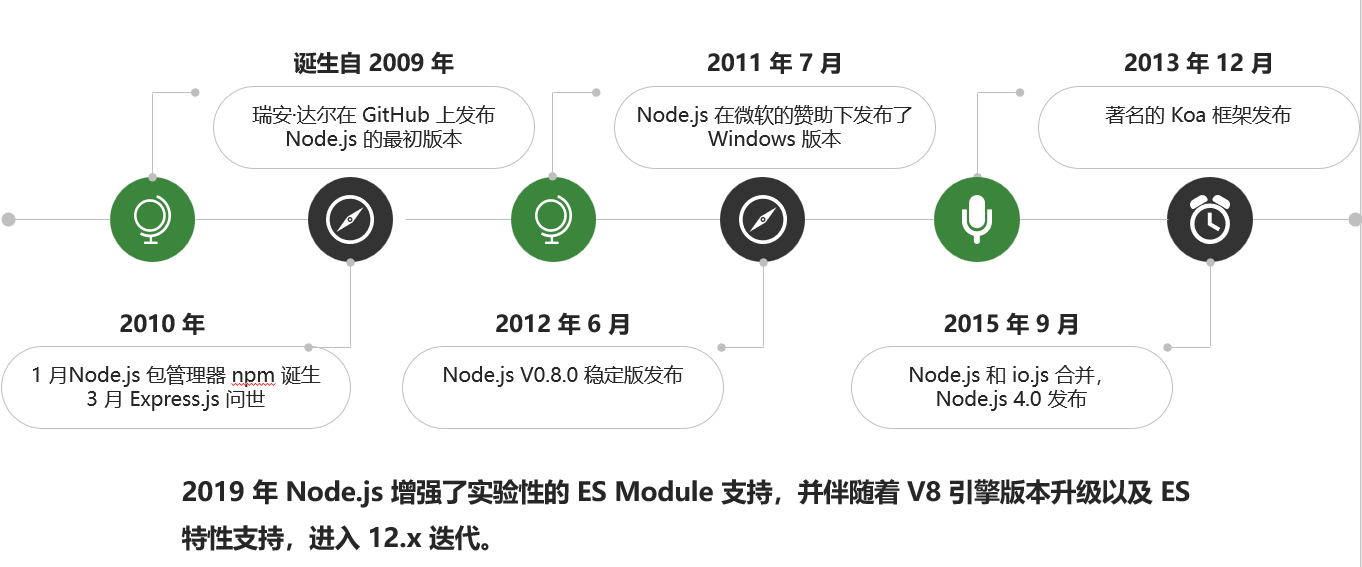
**程序设计模式：**

摒弃了传统平台依靠多线程来实现高并发的设计思路，采用了单线程、异步 I/O、事件驱动式

**架构可扩展性：**实时响应、超大规模数据要求



**发展历史：**



### 1.1 Node.js 中的 JavaScript 运行环境

浏览器是 JavaScript 的前端运行环境

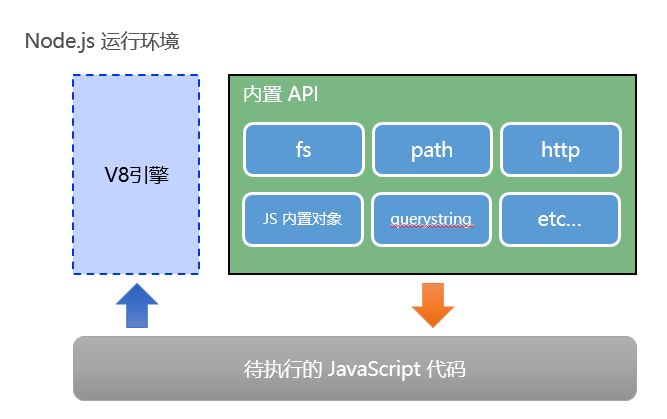
Node.js 是 JavaScript 的后端运行环境

Node.js 中无法调用 DOM 和 BOM 等浏览器内置 API

**区别**：Javascript是一种Web前端语言，Node.js 是JavaScript 语言的服务器运行环境

Javascript是一种Web前端语言，由浏览器解析执行，受制于浏览器提供的接口。浏览器为了安全考虑，对文件操作、网络操作、操作系统交互等功能有严格的限制，所以在浏览器端的 JavaScript 功能受限。

Node.js 事实上既是一个运行环境，同时又是一个库。 Node.js 完全没有浏览器端的限制，让 JavaScript 拥有了文件操作、网络操作、进程操作等功能，允许脱离浏览器环境运行 JavaScript 代码。



### 1.2 Node.js 特点及应用场景

Node.js 作为一个 JavaScript 的运行环境，仅仅提供了基础的功能和 API。

然而，基于 Node.js 提供的这些基础能，出现了很多强大的工具和框架

* 基于 Express 框架（<http://www.expressjs.com.cn/>），可以快速构建 Web 应用
* 基于 Electron 框架（<https://electronjs.org/>），可以构建跨平台的桌面应用
* 基于 restify 框架（<http://restify.com/>），可以快速构建 API 接口项目
* 读写和操作数据库、创建实用的命令行工具辅助前端开发、etc…

**特点：**

* 采用异步式 I/O 与事件驱动架构设计

单线程模型，在执行过程中只启动一个线程来运行代码

CPU 和内存在同一时间集中处理一件事，同时尽可能让耗时的 I/O 操作并行执行

* 拥有强大而灵活的包管理器

有上万个第三方模块

网站开发框架、数据库接口，模板语言解析、CSS 生成工具、图形用户界面和操作系统 API 工具等

* 内置 HTTP 服务器

作为服务器向用户提供服务，它跳过了 HTTP 服务器，直接面向前端开发。

* 具有强大的标准类库

二进制类库、核心模块

二进制类库包括 libuv，为网络以及文件系统提供了快速的时间轮循以及非阻塞 I/O

HTTP 类库，快速构建 HTTP 客户端和服务器

**应用场景：**

Web 服务 API 实时多人游戏 后端 Web 服务 基于 Web 的应用 多客户端的通信

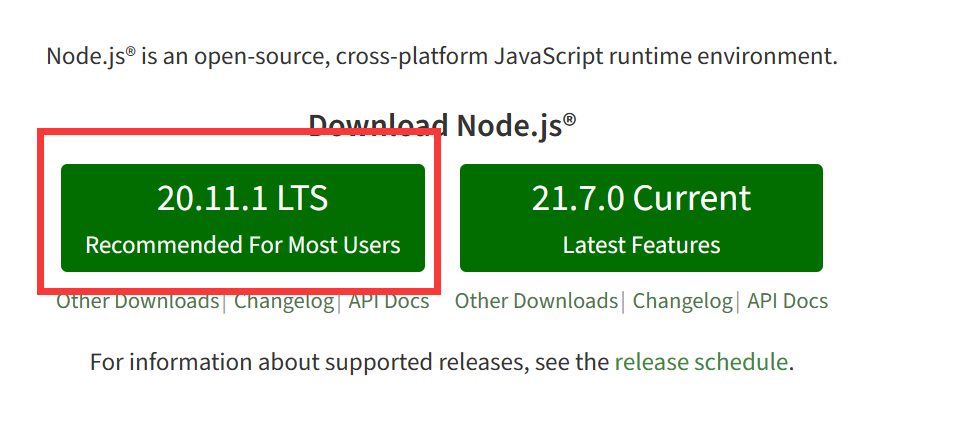
**学习路径：**

JavaScript 基础语法 + Node.js 内置 API 模块（fs、path、http等）

+ 第三方 API 模块（express、mysql 等）

### 1.3 Node.js 环境的安装

官网：<https://nodejs.org/en>



#### 1.3.1 区分 LTS 版本和 Current 版本的不同

LTS 为长期稳定版，对于追求稳定性的企业级项目来说，推荐安装 LTS 版本的 Node.js。

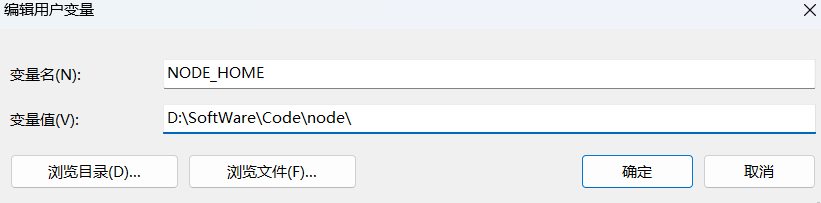
Current 为新特性尝鲜版，对热衷于尝试新特性的用户来说，推荐安装 Current 版本的 Node.js。但是，Current 版本中可能存在隐藏的 Bug 或安全性漏洞，因此不推荐在企业级项目中使用 Current 版本的 Node.js。

#### 1.3.2 安装及验证

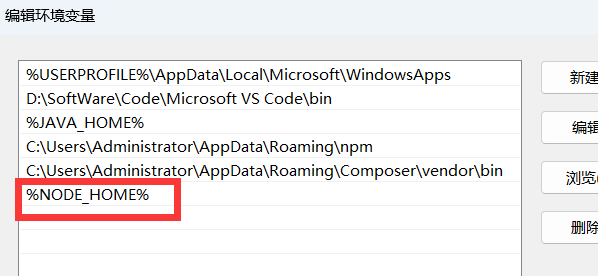
以安装路径D:\SoftWare\Code\node 为例子

我的电脑右键属性---高级系统设置---环境变量---新建系统变量

**新建系统变量为NODE\_HOME,值为D:\SoftWare\Code\node**



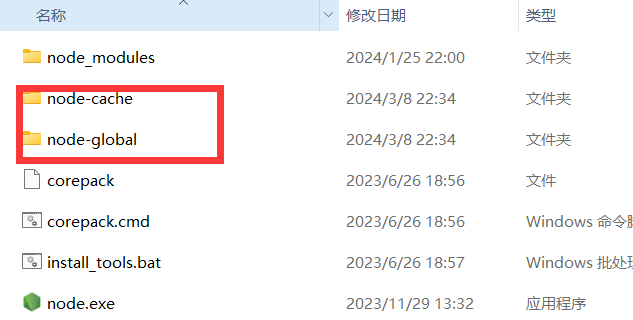
**新增PATH变量，%NODE\_HOME%**



**在D:\SoftWare\Code\node目录下新建两个文件夹**

node-global(说明：npm install -g XXX 安装全局模块的目录)

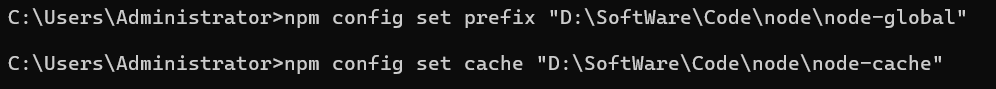
node-cache



**win+r打开cmd命令行，设置全局/缓存目录**

npm config set prefix "D:\SoftWare\Code\node\node-global"

npm config set cache "D:\SoftWare\Code\node\node-cache"



**设置npm源（可选，提高开发效率）**

npm config set registry <https://registry.npm.taobao.org>（淘宝源）

<https://npm.aliyun.com/>（阿里源）<http://r.cnpmjs.org/>（cnpm源）<https://registry.npmjs.org/>（官方）

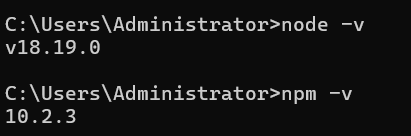


npm config get registry 验证现使用源



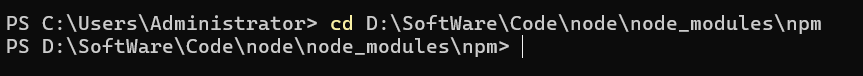
**验证安装**

node -v npm -v



**进行依赖更新**

切换到安装目录cd D:\SoftWare\Code\node\node\_modules\npm



输入命令：npm install进行依赖更新 ，更新完毕即可



能够正常显示版本信息，则安装正确

安装失败问题可以尝试：

npm cache clean --force # 先清除一下缓存

npm config set registry https://registry.npmjs.org/ #恢复默认镜像

有使用代理的话，手动去删除代理

npm config rm https-proxy npm config delete proxy

## 终端

**快捷键**：

在 Windows 的 powershell 或 cmd 终端中，我们可以通过如下快捷键，来提高终端的操作效率：

1. 使用 ↑ 键，可以快速定位到上一次执行的命令
2. 使用 tab 键，能够快速补全路径
3. 使用 esc 键，能够快速清空当前已输入的命令
4. 输入 cls 命令，可以清空终端

**在Node.js 环境中执行 JavaScript 代码：**

打开终端---输入 node 要执行的js文件的路径

## 3. 第一个node.js小程序

使用编辑器vscode新建文件helloword.js文件

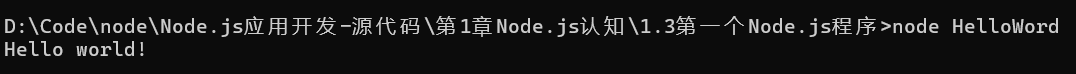


**运行方式：**

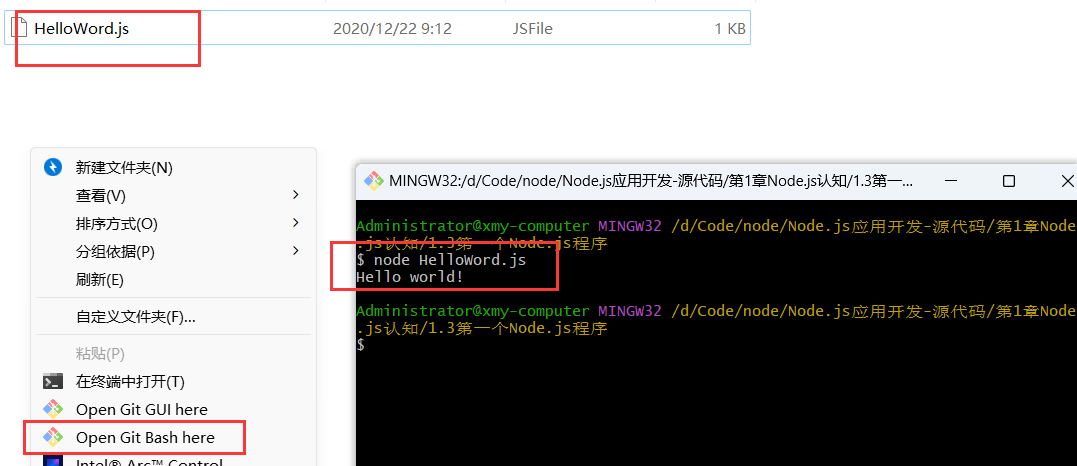
* 通过运行，以node.js程序进行运行



* 或通过终端，进入指定路径，以node 文件名进行运行



* 或通过git bash进行运行



## node.js的控制台console

Console 对象提供了一个简单的调试控制台，类似于Web 浏览器提供的JavaScript 控制台。

使用Console 对象的一系列方法可以将调试模式的信息输出到控制台。

|  |  |
| --- | --- |
| 方法 | 功能 |
| log() | 控制台输出一条信息 |
| info() | 控制台输出一条提示信息 |
| error() | 输出错误信息到控制台 |
| warn() | 输出警示信息 |
| table() | 以表格形式显示数据 |
| time() | 计时器，开始计时间，与timeEnd()联合使用，用于计算一个操作花费的准确时间 |
| timeEnd() | 计时结束 |
| assert() | 如果断言为false，则在信息到达控制台时输出错误信息 |

### 4.1 console. log()输出普通信息

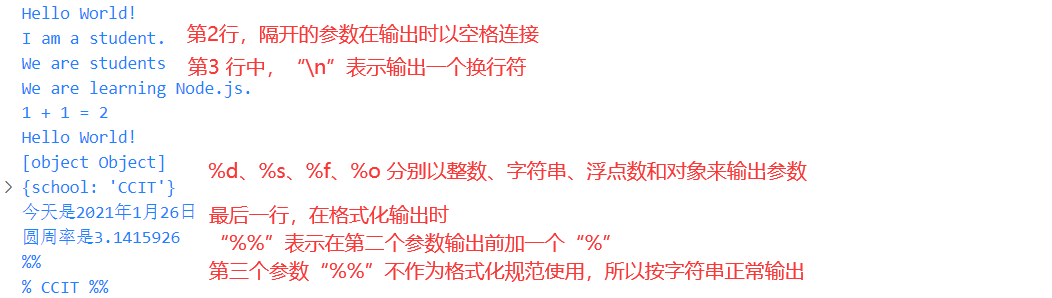
console.log()方法可用于在控制台输出普通信息，如单个变量（表达式）、多个变量、换行以

及格式化输出，类似于C 语言中的printf()。

格式化输出时可使用类似printf()风格的占位符

支持字符（%s）、整数（%d）、浮点数（%f）和对象（%o）4 种占位符。





**占位符：**

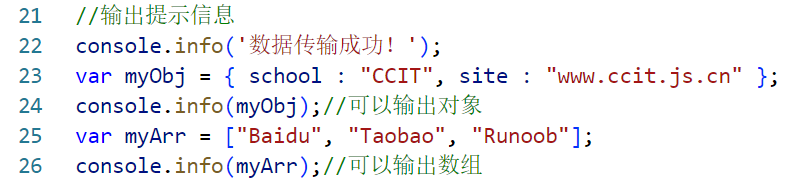
|  |  |
| --- | --- |
| 占位符 | 占位符 含义 |
| %s | 字符串输出 |
| %d | 整数输出 |
| %f | 浮点数输出 |
| %o | 打印JavaScript 对象  可以是整数、字符串以及JS 对象简谱（JavaScript Object Notation，JSON）数据 |
| %% | 百分比输出 |
| %c | 补充：特许占位符，用于修改控制台文字样式 |





### 4.2 console. info()输出提示信息

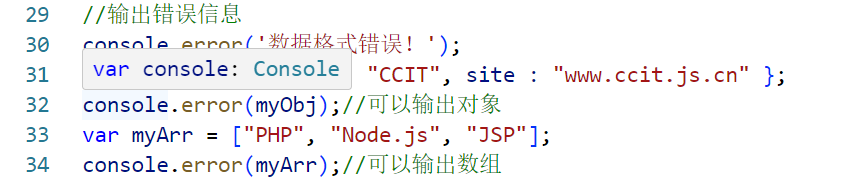
console.info()方法可以用于在控制台输出提示信息





### 4.3 console. error()输出错误信息

console.error()方法用于输出错误信息到控制台





### 4.4 console. warn()输出警示信息

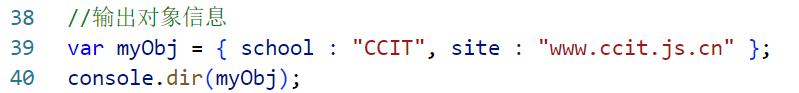
console.warn()方法用于输出警示信息到控制台,在Node.js 中可以使用console.warn()方法来代替console.error()方法，两个方法的使用方法完全相同。





### 4.5 console. dir()输出对象信息

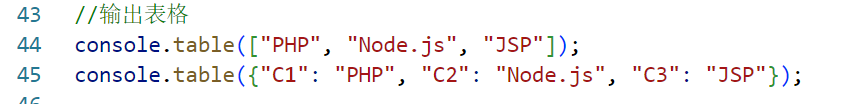
console.dir()方法可以显示一个对象的所有属性和方法

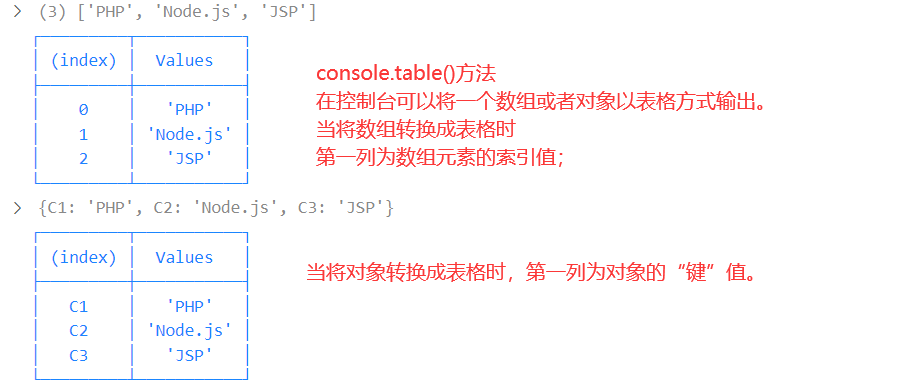




### 4.6 console. table()输出表格

console.table()方法用来在控制台输出一个表格

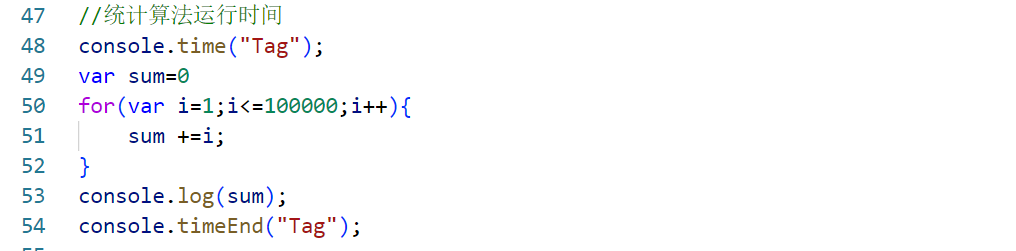


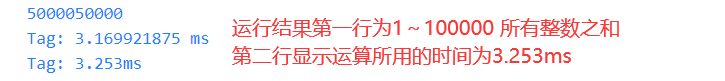


### 4.7 console. time()和console. timeEnd()

若需要统计某个算法的运行时间，可以使用console.time()方法和console.timeEnd()方法

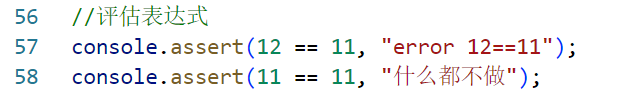
这两个方法都要接受一个字符串作为参数，两个方法的参数要相同，这样才能正确计算出算法从开始到结束运行的时间。





### 4.8 console. assert()评估表达式

console.assert()在第一个参数值为false 的情况下会在控制台输出信息



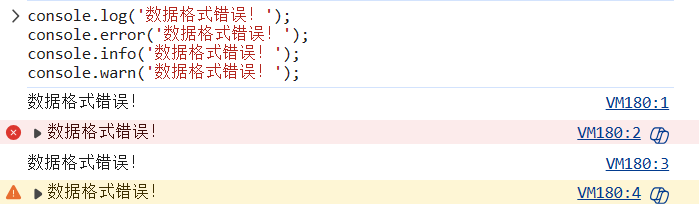


console.assert()对表达式结果进行评估

如果该表达式的执行结果为false，则输出一个消息字符串并抛出AssertionError 异常。

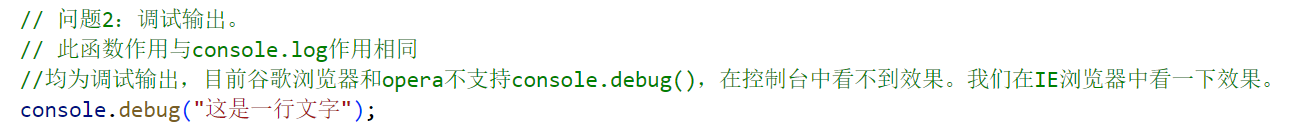
若参数表达式返回true，则该语句什么都不做。

### 4.9 浏览器的控制台输出信息

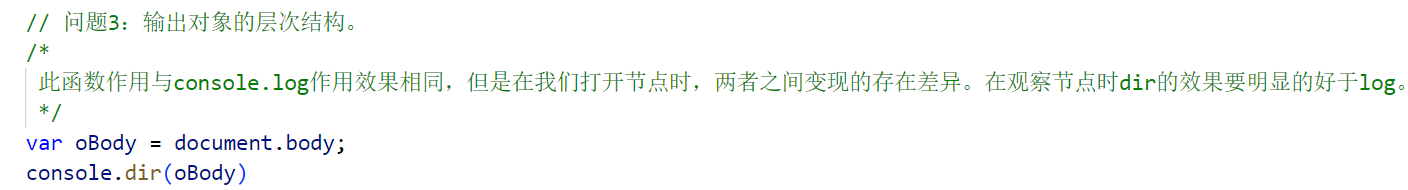


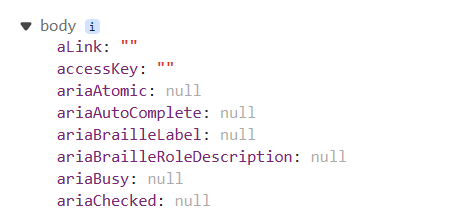
### 实验及补充

#### 4.10.1 调式输出：console.debug



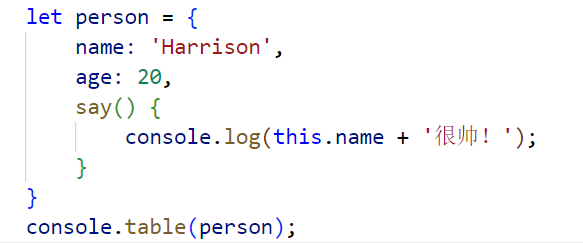
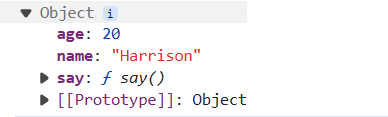
#### 4.10.2 输出对象层级结构：console.dir



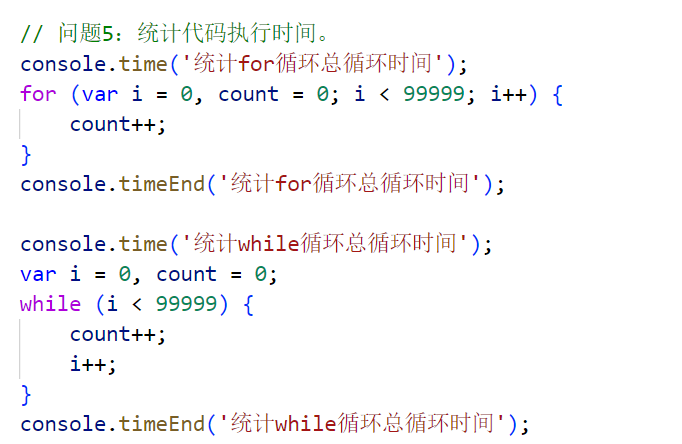


#### 4.10.3 输出对象表格化：console.table

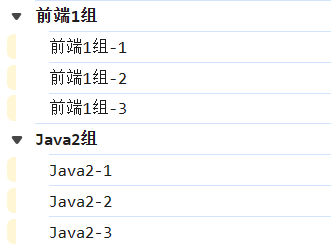
 

#### 4.10.4 统计代码执行时间：console.time和console.timeEnd

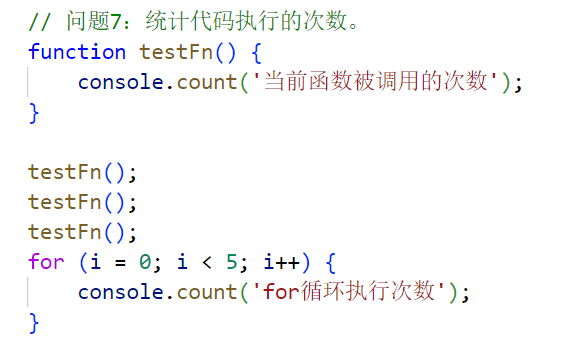
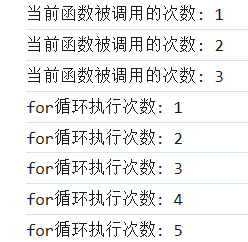




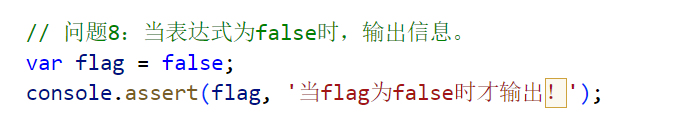
#### 4.10.5 分组输出信息：console.group和console.groupEnd

#### 4.10.6 统计代码执行的次数：testFn

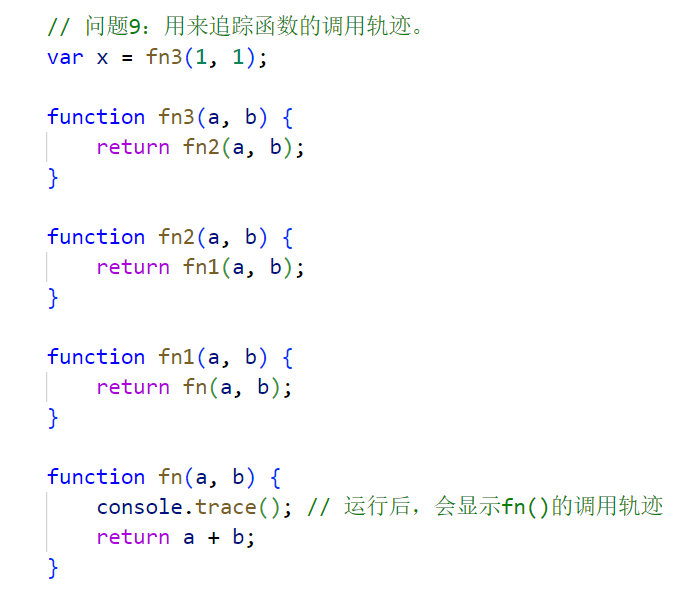
 

#### 4.10.7 当表达式为false时，输出信息console.assert





#### 4.10.8 用来追踪函数的调用轨迹：trace

#### 4.10.9 清除控制台所有内容

console.clear()