一. Node.js 基础介绍

1. 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

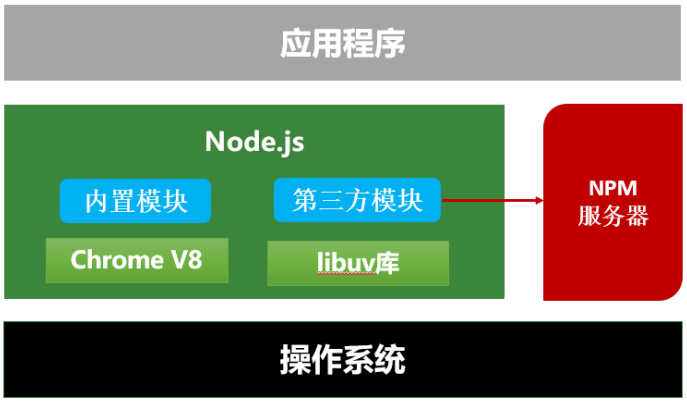
官网：https://nodejs.org/en

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

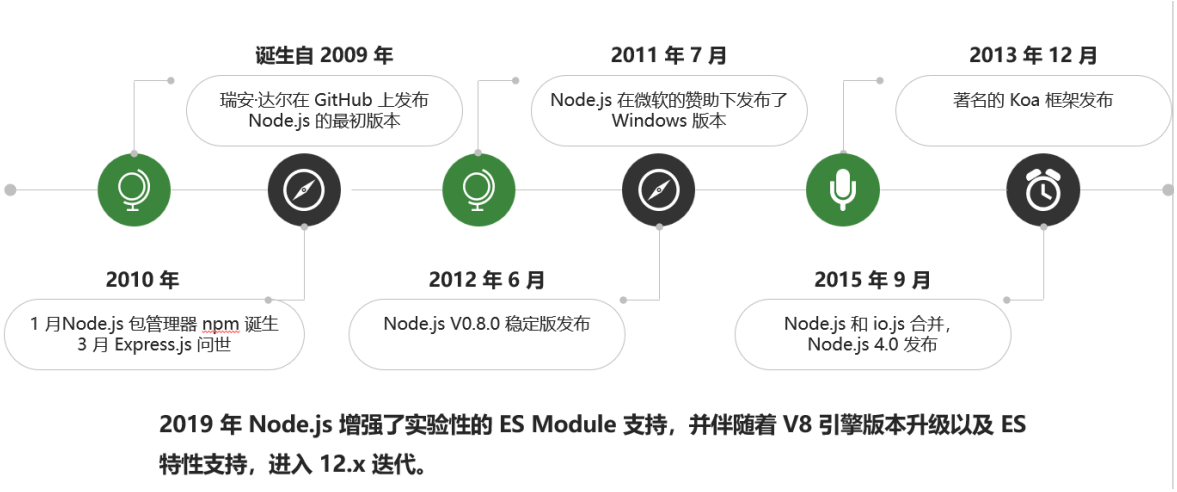
程序设计模式：

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：



2019 年 Node.js 增强了实验性的 ES Module 支持，并伴随着 V8 引擎版本升级以及 ES 特性支持，进入 12.x 迭代。

1.1 Node.js 中的 JavaScript 运行环境

浏览器是 JavaScript 的前端运行环境

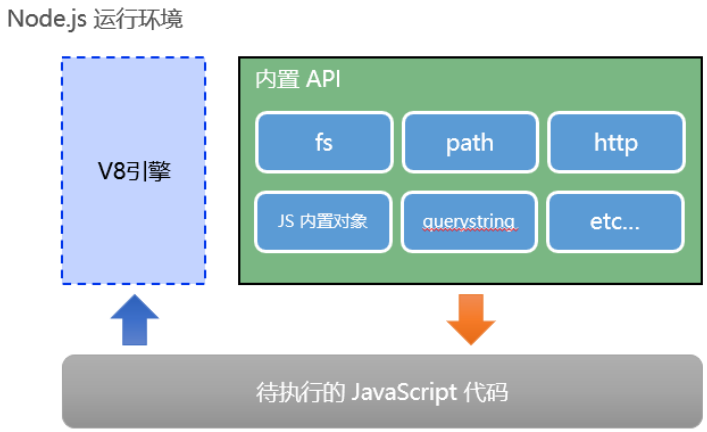
Node.js 是 JavaScript 的后端运行环境

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

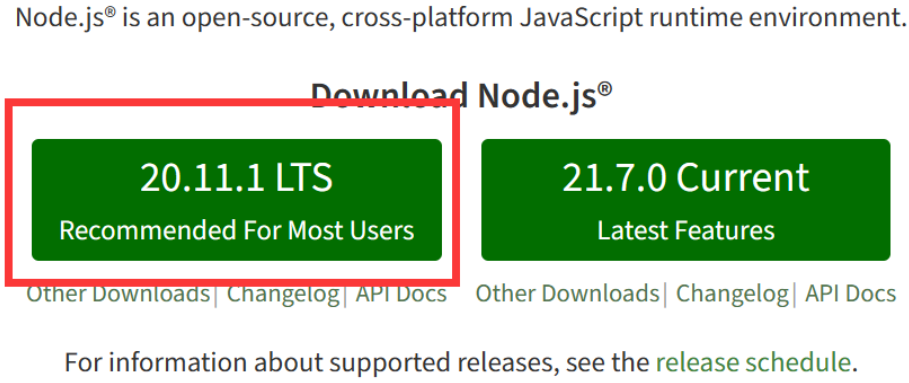
学习路径：

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。

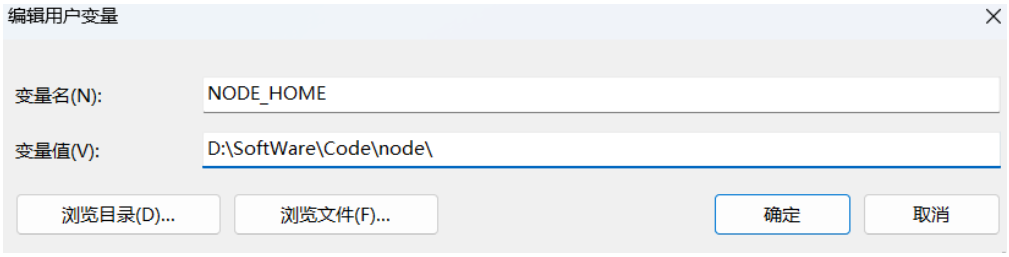
Current 为新特性尝鲜版，对热衷于尝试新特性的用户来说，推荐安装 Current 版本的 Node.js。但是,Current 版本中可能存在隐藏的 Bug 或安全性漏洞,因此不推荐在企业级项目中使用 Current 版本的 Node.js。

1.3.2 安装及验证

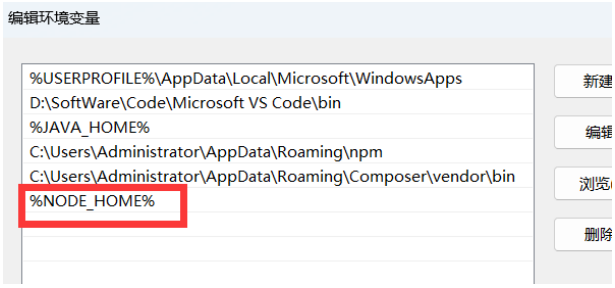
以安装路径 D:\SoftWare\Code\node 为例子

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



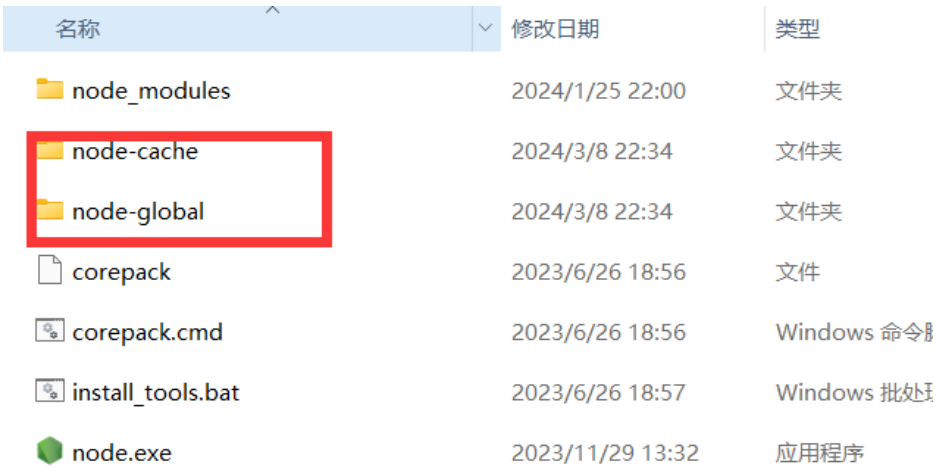
新增 PATH 变量，%NODE_HOME%



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"

```
C:\Users\Administrator>npm config set prefix "D:\SoftWare\Code\node\node-global"

C:\Users\Administrator>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）

```
C:\Users\Administrator>npm config set registry "https://registry.npm.taobao.org"
```

npm config get registry 验证现使用源

```
PS D:\SoftWare\Code\node\node_modules\npm> npm config get registry
https://registry.npm.taobao.org
```

验证安装

node -v　　npm -v

```
C:\Users\Administrator>node -v
v18.19.0

C:\Users\Administrator>npm -v
10.2.3
```

进行依赖更新

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

```
PS C:\Users\Administrator> cd D:\SoftWare\Code\node\node_modules\npm
PS 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

```
PS D:\SoftWare\Code\node\node_modules\npm> npm install
```

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

2. 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

```
2   console.log('Hello World!');
```

运行方式：

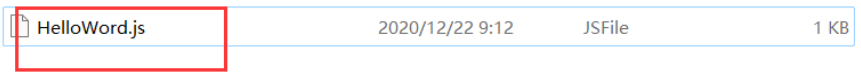
➢ 通过运行，以 node.js 程序进行运行

```
Hello World!
```

➢ 或通过终端，进入指定路径，以 node 文件名进行运行

```
D:\Code\node\Node.js应用开发-源代码\第1章Node.js认知\1.3第一个Node.js程序>node HelloWord
Hello world!
```

➢ 或通过 git bash 进行运行



4. 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。

```
2  //输出普通信息
3  console.log('Hello World!');
4  console.log('I', 'am', 'a', 'student.');
5  console.log('We are students\nWe are learning Node.js.');
6  console.log('%d + %d = %d', 1, 1, 2);
7
8  console.log("%s", "Hello", "World!");// 依次序输出所有字符串,%s支持字符输出
9
10 console.log("%s",  {school: "CCIT"});// 将对象转换为普通字符串后执行
11 console.log("%o",  {school: "CCIT"});//%o输出对象
12
13 console.log("今天是%d年%d月%d日",2021,01,26);// 将字符串作为数值进行转换,%d输出整数
14
15 console.log("圆周率是%f",3.1415926);//%f输出浮点数
16
17 console.log("%%");// 输出%,仅作为字符串输出
18 console.log("%%", "CCIT","%%");//% CCIT %%
```

```
Hello World!
I am a student.
We are students
We are learning Node.js.
1 + 1 = 2
Hello World!
[object Object]
> {school: 'CCIT'}
今天是2021年1月26日
圆周率是3.1415926
%%
% CCIT %%
```

第2行，隔开的参数在输出时以空格连接

第3 行中，“\n”表示输出一个换行符

%d、%s、%f、%o 分别以整数、字符串、浮点数和对象来输出参数

最后一行，在格式化输出时

“%%”表示在第二个参数输出前加一个“%”

第三个参数“%%”不作为格式化规范使用，所以按字符串正常输出

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

```
console.info("%c数据传输\n成功", "color:green;"); //%c特许占位符，用于修改控制台文字颜色,换行符（\n）
```

```
数据传输
成功
```

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

```
21  //输出提示信息
22  console.info('数据传输成功！');
23  var myObj = { school : "CCIT", site : "www.ccit.js.cn" };
24  console.info(myObj);//可以输出对象
25  var myArr = ["Baidu", "Taobao", "Runoob"];
26  console.info(myArr);//可以输出数组
```

```
> {school: 'CCIT', site: 'www.ccit.js.cn'}
> (3) ['Baidu', 'Taobao', 'Runoob']
```

console.info()方法可以输出一个字符串，也可以输出对象和数组

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

```
29  //输出错误信息
30  console.error('数据格式错误！');
31  var console: Console  "CCIT", site : "www.ccit.js.cn" };
32  console.error(myObj);//可以输出对象
33  var myArr = ["PHP", "Node.js", "JSP"];
34  console.error(myArr);//可以输出数组
```

```
  数据格式错误！
> {school: 'CCIT', site: 'www.ccit.js.cn'}
> (3) ['PHP', 'Node.js', 'JSP']
```

console.error()方法在控制台以红色文字打印以上字符串、对象和数组信息。

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。

```
27  //输出警示信息
28  console.warn("数据传输成功警告信息");
```

```
(3) [ 'www' , 'baidu' , 'com' ]
数据传输成功警告信息
```

console.warn()方法在控制台以红色文字打印以上字符串信息

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

```
38  //输出对象信息
39  var myObj = { school : "CCIT", site : "www.ccit.js.cn" };
40  console.dir(myObj);
```

```
> {school: 'CCIT', site: 'www.ccit.js.cn'}
```

4.6 console. table()输出表格

console.table()方法用来在控制台输出一个表格

```
43  //输出表格
44  console.table(["PHP", "Node.js", "JSP"]);
45  console.table({"C1": "PHP", "C2": "Node.js", "C3": "JSP"});
```

```
> (3) ['PHP', 'Node.js', 'JSP']
```

(index)	Values
0	'PHP'
1	'Node.js'
2	'JSP'

console.table()方法
在控制台可以将一个数组或者对象以表格方式输出。
当将数组转换成表格时
第一列为数组元素的索引值;

```
> {C1: 'PHP', C2: 'Node.js', C3: 'JSP'}
```

(index)	Values
C1	'PHP'
C2	'Node.js'
C3	'JSP'

当将对象转换成表格时，第一列为对象的“键”值。

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。

```
47  //统计算法运行时间
48  console.time("Tag");
49  var sum=0
50  for(var i=1;i<=100000;i++){
51      sum +=i;
52  }
53  console.log(sum);
54  console.timeEnd("Tag");
```

```
5000050000
Tag: 3.169921875 ms
Tag: 3.253ms
```

运行结果第一行为1～100000 所有整数之和
第二行显示运算所用的时间为3.253ms

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

```
56  //评估表达式
57  console.assert(12 == 11, "error 12==11");
58  console.assert(11 == 11, "什么都不做");
```

```
error 12==11
Assertion failed: error 12==11
```

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

```
> console.log('数据格式错误！');
  console.error('数据格式错误！');
  console.info('数据格式错误！');
  console.warn('数据格式错误！');
```

数据格式错误！ VM180:1

▶ 数据格式错误！ VM180:2

数据格式错误！ VM180:3

▶ 数据格式错误！ VM180:4

4.10 实验及补充

4.10.1 调式输出：console.debug

```
// 问题2：调试输出。
// 此函数作用与console.log作用相同
//均为调试输出，目前谷歌浏览器和opera不支持console.debug()，在控制台中看不到效果。我们在IE浏览器中看一下效果。
console.debug("这是一行文字");
```

4.10.2 输出对象层级结构：console.dir

```
// 问题3：输出对象的层次结构。
/*
此函数作用与console.log作用效果相同，但是在我们打开节点时，两者之间变现的存在差异。在观察节点时dir的效果要明显的好于log。
*/
var oBody = document.body;
console.dir(oBody)
```

```
▼ body i
    aLink: ""
    accessKey: ""
    ariaAtomic: null
    ariaAutoComplete: null
    ariaBrailleLabel: null
    ariaBrailleRoleDescription: null
    ariaBusy: null
    ariaChecked: null
```

4.10.3 输出对象表格化：console.table

```
// 问题4：输出数据表格化。
var students = [{
        name: '张三',
        email: 'zhangsan@163.com',
        qq: 12345
    },
    {
        name: '李四',
        email: 'lisi@126.com',
        qq: 12346
    },
    {
        name: '王五',
        email: 'wangwu@sina.com',
        qq: 12347
    },
    {
        name: '赵六',
        email: 'zhaoliu@gmail.com',
        qq: 12348
    }
];
console.table(students);
```

```
▼ Array(4) i
  ▶ 0: {name: '张三', email: 'zhangsan@163.com', qq: 12345}
  ▶ 1: {name: '李四', email: 'lisi@126.com', qq: 12346}
  ▶ 2: {name: '王五', email: 'wangwu@sina.com', qq: 12347}
  ▶ 3: {name: '赵六', email: 'zhaoliu@gmail.com', qq: 12348}
    length: 4
  ▶ [[Prototype]]: Array(0)
```

```
let person = {
    name: 'Harrison',
    age: 20,
    say() {
        console.log(this.name + '很帅！');
    }
}
console.table(person);
```

```
▼ Object i
    age: 20
    name: "Harrison"
  ▶ say: ƒ say()
  ▶ [[Prototype]]: Object
```

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

```
// 问题5：统计代码执行时间。
console.time('统计for循环总循环时间');
for (var i = 0, count = 0; i < 99999; i++) {
    count++;
}
console.timeEnd('统计for循环总循环时间');

console.time('统计while循环总循环时间');
var i = 0, count = 0;
while (i < 99999) {
    count++;
    i++;
}
console.timeEnd('统计while循环总循环时间');
```

```
统计for循环总循环时间: 1.384033203125 ms
统计while循环总循环时间: 1.327880859375 ms
```

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

```
// 问题6：分组输出信息。
console.group('前端1组');
console.log('前端1组-1');
console.log('前端1组-2');
console.log('前端1组-3');
console.groupEnd();

console.group('Java2组');
console.log('Java2-1');
console.log('Java2-2');
console.log('Java2-3');
console.groupEnd();
```

```
▼ 前端1组
    前端1组-1
    前端1组-2
    前端1组-3
▼ Java2组
    Java2-1
    Java2-2
    Java2-3
```

4.10.6 统计代码执行的次数：testFn

```
// 问题7：统计代码执行的次数。
function testFn() {
    console.count('当前函数被调用的次数');
}

testFn();
testFn();
testFn();
for (i = 0; i < 5; i++) {
    console.count('for循环执行次数');
}
```

```
当前函数被调用的次数: 1
当前函数被调用的次数: 2
当前函数被调用的次数: 3
for循环执行次数: 1
for循环执行次数: 2
for循环执行次数: 3
for循环执行次数: 4
for循环执行次数: 5
```

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

```
// 问题8：当表达式为false时，输出信息。
var flag = false;
console.assert(flag, '当flag为false时才输出!');
```

```
声明失败：当flag为false时才输出!
```

4.10.8 用来追踪函数的调用轨迹：trace

```
// 问题9：用来追踪函数的调用轨迹。
var x = fn3(1, 1);

function fn3(a, b) {
    return fn2(a, b);
}

function fn2(a, b) {
    return fn1(a, b);
}

function fn1(a, b) {
    return fn(a, b);
}

function fn(a, b) {
    console.trace(); // 运行后，会显示fn()的调用轨迹
    return a + b;
}
```

```
▼ console.trace
  fn @ console的使用.html:121
```

4.10.9 清除控制台所有内容

console.clear()