1. es6 ECMAScript 6

宿主是网页运行环境

1. 块级作用域

使用let关键字声明变量，阻止变量提升，添加到块级作用域中，成为局部变量。let 不能重复声明，后面直接赋值就行。

块级作用域：{ } if else while for switch...

1. 参数增强

|  |
| --- |
| function add(a,b, c=0 ){  console.log(a+b+c);  } add(10,20) |

给参数设默认值。若不将C设为默认参数，会打印NaN，仅可形参最后一位设默认。

1. 箭头函数（回调函数）

回调函数的另一种写法，可用箭头代替function，但不等价于匿名

|  |  |
| --- | --- |
| var arr = [23,9,45,78,6];  console.log(arr.soft( function(a,b){  return a-b;  } )); | var arr = [23,9,45,78,6];  console.log(arr.soft((a,b) => {  return a-b;  } )); |
| 若函数体中只有一行代码且是return形式，可省略为console.log(arr.soft( (a,b)=> a-b )); a-b后没分号 | |

1. 模板字符串（` .... `）

|  |
| --- |
| 在模板字符串之间可以放任何形式字符，全认为是字符串，若要放表达式则用 ${表达式} 拼接。{ }内部可进行简单运算或三目  console.log(`你好骚啊${a}${b+3}`); |

1. Node JS
2. NodeJS概述

基于谷歌V8引擎（JS解释器），运行在服务器端的环境

对比JS和NodeJS：

1. JS运行在浏览器端，存在多种浏览器解释器，容易产生兼容性的问题；nodeJS运行在服务器端，只有V8引擎一种解释器，不存在兼容问题。
2. 两者都有内置对象、自定义对象，宿主对象（不全相同）
3. JS用于浏览器端的交互效果，nodeJS用于服务器端操作，例如创建web服务器、操作数据库、文件操作
4. NodeJS执行方式

脚本模式 node C:/..../sha.js 回车

交互模式 node 回车

退出按 Ctrl+C两次 或 直接输入 .exit 回车

1. NodeJS的特点

简单、避免过度设计

单线程逻辑处理

非阻塞的异步I/O处理（输入输出数据速度很快）

事件驱动编程

无锁机制，不会产生锁死项

支持数万个并发连接

适用场景：

基于社交网络的大规模web应用（属于I/O密集型--输入输出比较多）

NodeJS对象

1. global全局对象

在交互模式下，声明的变量和创建的函数都属于全局对象下的，可以用global来访问 global.a global.fn()

在脚本模式下，声明的变量和创建函数都不属于全局对象下，不能用global访问

JS：window

在JS脚本中声明的变量和创建的函数都属于全局对象下的，可以用window访问 window.a

1. process 进程对象

进程：执行程序时，操作系统分配内存空间；

process.env 查看当前的环境变量（把软件加入环境变量可通过命令启动）

process.arch 查看当前CPU架构

process.platform 查看当前的操作系统

process.version 查看当前NodeJS版本号

process.pid 查看当前进程的编号

process.kill( ) 结束某个编号的进程

1. console (控制台)对象

console.log( ) 打印日志 ≈ console.info( ) 打印消息

console.warn( ) 打印警告 浏览器报黄

console.err( ) 打印错误 浏览器报红

console.time('自定义字符串'); 开始计时

console.timeEnd('自定义字符串'); 结束计时

注：前后 字符串 要保持一致

1. Buffer 缓冲对象

缓冲区：在内存中存储数据的区域，常用于网络传输时的资源

创建Buffer

var buf = Buffer.alloc(5,'abcde');

将buffer数据转为字符串

buf.toString()

注：一个中文占3个字符