1.递归

是一个函数的内部调用自身这个函数，本身是一个死循环

递归一直运行会产生内存泄漏，造成系统崩溃

如何使用：

要有边界条件，结合着return使用

JS属于单线程运行逻辑，无法充分利用CPU内核，递归对CPU消耗比较大，JS不适合深层次的递归嵌套。

2.匿名函数

function(){ }

没有名称的函数

(1)创建函数

|  |
| --- |
| 函数声明  function fn(){  } |
| 函数表达式  var fun=function(){  }  变量名称就是函数名称  调用 fun() |

对比**函数名称**和**函数名称()**

函数名称()：调用函数，得到函数的返回值

函数名称：本质上就是一个变量，保存了一个函数

对比**函数声明**和**函数表达式**创建函数的区别

函数声明存在函数提升，可以先写调用再写创建；

函数表达式只是变量声明提升，必须先写创建再写调用

练习：使用函数表达式创建函数getSum，传递任意两个数字，返回两个数字之间所有整数的和。

(2)匿名函数自调用

目的是为了防止全局污染

|  |
| --- |
| (function(){  //函数作用域下，里边的变量都是局部变量  })(); |

(3)回调函数

将函数作为参数来传递到另一个函数

|  |
| --- |
| function tao(madai){  madai() //调用传递的回调函数  }  function dong(){  }  tao(dong) //dong是回调函数  tao(function(){ }) //匿名函数是回调函数 |

3.系统函数

isNaN() 判断一个值是否为NaN，会隐式转换为数值型，

是NaN -> true 不是NaN -> false

isFinite() 判断一个值是否为有限值，只有Infinity是无限值，其它都是有限值

2/0 -> Infinity

eval() 执行字符串表达式

4.对象

是一组属性和方法的集合

一部手机，属性有颜色、品牌、CPU、内存... 方法有打电话、发短信、玩游戏、看视频、办公...

涛哥，属性有发色、身高、体重... 方法有摊煎饼、养兔子、跑接力赛...

万物皆对象

(1)JS中的对象

自定义对象：用户自己创建的对象

内置对象：JS提供的对象

宿主对象：根据不同的执行环境划分

(2)自定义对象创建方式

对象字面量

内置构造函数

自定义构造函数

(3)对象字面量

{ 属性名1: 属性值1, 属性名2: 属性值2.... }

属性名的引号可以省略，如果含有特殊字符必须添加

练习：创建一个商品对象，包含的属性有编号，标题，价格，是否在售

(4)访问属性

对象.属性名

对象['属性名']

如果访问没有的属性，则返回undefined

练习：创建一个图书对象，包含的属性有编号，书名、作者、价格；修改图书的价格，添加图书的出版社，出版时间；最后打印对象

(5)内置构造函数

new Object() 创建一个空对象，需要单独的添加每个属性

练习：创建一个汽车对象，包含有汽车的品牌，颜色，长度，宽度

(6)遍历属性

|  |
| --- |
| for(var k in 对象){  k代表每个属性名  对象[k] 代表属性名对应的属性值  } |

练习：使用对象字面量创建一个学生对象，包含有学号，姓名，性别，分数；遍历对象得到每个属性

课后任务

(1)复习今天内容，整理思维导图

(2)练习：创建对象包含一组成绩，遍历对象属性，计算出总成绩和平均成绩

(3)预习js第7天 数组