正课:

1. \*\*循环

2. \*\*\*数组

1. \*\*循环:

什么是: 让程序反复执行同一代码段

何时: 只要一段代码需要反复执行时

三要素:

1. 循环条件: 判断循环能否继续执行的条件

只有满足条件，才可继续执行循环体

一旦条件不满足，就自动退出循环

2. 循环变量: 循环条件中用作比较和判断的变量

初始值:

向着不满足循环条件的趋势不断变化, 每次递增/递减n

3. 循环体: 循环反复执行的代码段

死循环: 循环条件永远满足，导致循环无法自动退出

如何: 3种:

1. while循环:

声明并初始化循环变量;

while(循环条件){//只要/当...的时候

循环体

修改循环变量

}

何时: 只要必须先判断循环条件才能执行循环

2. do...while循环:

声明并初始化循环变量;

do{

循环体

修改循环变量

}while(循环条件);

何时: 即使循环条件不满足，也至少能执行(试)一次时

while vs do while

当第一次循环条件都满足时，两者完全相同

只有第一次循环条件不满足时,差别:

while是一次都不执行

do...while是至少可以执行一次

3. for循环: 简化版的while循环:

for(声明并初始化循环变量;循环条件; 修改循环变量){

循环体

}

局限: 只能在循环体执行后，才修改循环变量的值

何时: 只有循环变量的变化有规律时，才用for循环

如果循环变量的变化没有规律，就用while循环

更简化:

1. for循环第一部分,可同时声明并初始化多个变量

2. for循环第三部分,可执行多个短小的语句，用逗号分隔

强调: 绝不能改变源程序的执行顺序

强调: 如果for循环，简化到没有循环体，则结尾必须加分号

break和continue:

break: 中断并退出当前结构的执行

何时: 只要希望不再执行后续循环时

continue: 跳过本轮循环，继续下一轮

何时: 后续循环依然希望执行时  
 其实，颠倒判断条件，就可避免使用continue

嵌套循环:

什么是: 在外层循环内，又执行了另一个内层循环

2. \*\*\*数组:

什么是: 内存中连续存储多个数据的存储空间，再起一个名字

何时: 只要连续存储多个数据，都用数组

为什么: 连续存储的多个数据，非常便于统一查找和使用

程序=数据结构+算法

数据结构: 数据在内存中的存储结构

算法: 解决问题的步骤

好的数据结构可极大提高程序的执行效率

如何:

1. 创建: 3种:

1. 创建一个空数组:

var 数组名=new Array()

可简化: var 数组名=[];

何时: 只要创建数组时，暂时不知道数组的内容

2. 创建数组同时，就保存数据

var 数组名=new Array(值1,值2,...);

可简化: var 数组名=[值1,值2,...];

何时: 只要创建数组时，已经知道数组的内容

3. 创建n个空元素的数组:

var 数组名=new Array(n);

何时: 创建数组时，如果仅知道将来的数据个数

.length属性: 记录数组中理论上的元素个数！

元素: 保存在数组中的每个数据，称为一个元素

下标(i,index): 序号，数组中唯一标识每个元素存储位置的房间号

默认: 下标从0开始，连续递增，不重复

2. 赋值: 将数据保存到数组中指定位置的元素中

arr[i]=值

特殊: 如果给不存在位置强行赋值，不会报错！

会在指定位置自动创建新元素保存数据。

3. 取值: 取出数组中指定位置的元素内容使用

arr[i]

特殊: 强行访问数组中不存在的位置的值，不会报错，而是返回undefined

总结: 数组中每个元素的用法和普通变量完全一样！

数组的另一个称呼: 一组变量的集合，再起一个名字.

.length的固定用法:

.length永远等于最大数字下标+1——自动维护

固定套路:

1. 获得最后一个元素: arr[arr.length-1]

获得倒数第n个元素: arr[arr.length-n]

2. 在数组末尾追加一个新元素: arr[arr.length]=新值

3. 其实length属性可修改: ——缩容

比如: 只删除数组末尾的一个元素: arr.length--

删除末尾的n个元素: arr.length-=n;

清空数组: arr.length=0;

js数组三个不限制:

1. 不限制元素的数据类型

2. 不限制元素个数

3. 不限制下标越界

\*\*\*数组是引用类型的对象:

什么是对象: 内存中存储多个数据的一块独立存储空间

什么是引用类型: 值无法保存在变量本地的数据类型

实际数据都存储在window之外的独立对象(存储空间)中

每个对象都有一个唯一的地址值，标识其存储位置

变量中其实仅保存地址值，通过地址引用对象

回顾按值传递:

对原始类型的值: 修改新变量，不影响原变量

对引用类型的对象: 修改新变量，等效于直接修改原对象！

因为按值传递，将原变量中的地址值，复制给新变量

新旧变量用相同的地址值，引用了同一个对象

用新变量修改对象，等效于直接修改原对象。

垃圾回收:

什么是: 引擎会自动释放不再使用的对象的内存空间

为什么: 内存空间的总量是有限的！

何时: 只要一个对象不再被使用，就会被垃圾回收

如何: 不用自己做！

垃圾回收器: 专门监控内存中的对象，并释放不再被使用的对象的小程序

随程序启动而启动，并伴随主程序在后台同步执行.

一个对象，只要还被任意一个变量引用，就不会被回收

建议: 当一个对象不再使用时，应主动释放对对象的引用:

如何: var 变量=null;

当一个对象不再被任何变量引用时，就会被垃圾回收！

遍历:

什么是: 对数组中每个元素依次执行相同的操作

何时: 只要想对数组中每个元素执行相同的操作时

如何:

for(var i=0;i<arr.length;i++){

arr[i]//当前正在遍历的元素内容

}

作业: 100题: 1,8,9,11,12,14,17