正课:

1. \*\*\*循环

2. \*\*\*数组

1. \*\*\*循环:

什么是: 让程序反复执行同一段代码

何时: 只要让程序反复执行同一段代码

如何: 三要素:

1. 循环条件: 让循环可以继续执行的条件

2. 循环变量: 在循环条件中用作比较和判断的变量

从几开始, 每次递增/递减几，到几结束

3. 循环体: 循环反复执行的代码段

几乎都要修改循环变量，向着不满足循环条件的趋势。

包括: 3种:

while: var 循环变量=初始值;

while(循环条件){

循环体;

修改循环变量

}

何时: 必须先验证循环条件，才能执行循环体时

do while:

var 循环变量=初始值;

do{

循环体;

修改循环变量

}while(循环条件);

何时: 希望即使不满足，也至少可以执行一次时

while vs do while

如果第一次循环都满足，两者的输出是完全一样的

while: 先验证后执行

do while: 先执行一次，再验证

如果第一次条件不满足，则while是一次都不执行，do while至少执行一次。

for:

for(var 循环变量=初始值;循环条件;修改循环变量){

循环体;

}

for循环和while循环的原理是完全一样的。

for vs while: 如果循环变量的变化有规律: 首先for

如果循环变量的变化没有规律: 首先while

强调: js中没有块级作用域:

放在if/else if/else/for/while中的变量，出了块，依然可用

vs java所有{}都认为是块级作用域，块中的变量出了块，不能使用。

解决: 让循环中的变量不要泄露到循环外

用let代替var: let声明的变量仅能在{}内使用

简写: 1. 第一部分: 可同时声明并初始化多个变量

2. 循环体:

如果for循环下只有一句话，可省略{}——不建议

如果for循环下只有一句话，且很简单:

for循环的第三部分可先后执行多个短小的操作，每个操作之间用逗号分隔。——不能改变原来的执行顺序

break vs continue:

break: 中断并退出当前循环

何时: 当判断条件非常复杂时，就可用死循环+break的方式灵活控制循环退出。——降低循环编写的难度

continue: 跳过本轮循环，继续执行下一轮

只要颠倒判断条件，就可避免使用continue

强调: break不能参与到任何简写中，必须独立一句。

100题: 1,9,11,12,14,17

嵌套循环: 循环内，又执行了另一个循环

如何: 1. 截取片段找规律

2. 用外层循环反复调用规律的公式

作业: 打印反三角和等腰三角

2. \*\*\*数组:

什么是: 内存中连续存储多个数据的一块存储空间，再起一个统一的名字。

为什么: 便于维护和查找

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

何时: 今后只要保存多个同一类型的数据时，都必须用数组

如何: 3件事:

创建: 3种:

1. 创建空数组: 2种: var arr=[];

new Array();

何时: 创建数组时，暂时不知道数组内容

2. 创建数组同时初始化数组内容:

var arr=[值1, 值2, ...];

new Array(值1,值2,...)

何时: 创建数组时已知数组的内容

3. 创建n个空元素的数组:

var arr=new Array(n);

何时: 创建数组时仅知道数组元素个数，暂时不知道内容。

下标: 数组中唯一标识一个元素存储位置的序号

从0开始，连续不重复

赋值: 将数据保存到数组的某个元素中

arr[i]=新值;

取值: 数组中每个元素的用法和普通的变量完全一样

数组也称为一组变量的集合，起一个统一的变量名

js的数组三个不限制:

1. 不限制元素的数据类型

2. 不限制下标越界:

赋值: 不会报错！自动在指定的新位置添加新元素。

如果添加后，数组的元素下标不连续——稀疏数组

取值: 不会报错! 仅返回undefined

3. 不限制元素个数:

在任何时候，在任何位置添加新元素

.length属性: 记录理论上数组中的元素个数

.length属性永远等于最后一个数字下标+1

.length属性永远指向最后一个元素的下一个新位置

.length可修改数组大小——缩容

固定套路: 1. 获取数组最后一个元素:

arr[arr.length-1]

2. 获得倒数第n个元素

arr[arr.length-n]

3. 末尾追加一个新元素:

arr[arr.length]=新元素

4. 删除数组最后一个元素

arr.length-=1;

5. 删除数组末尾倒数n个元素

arr.length-=n;

\*\*\*数组是引用类型的对象:

按值传递:

原始类型的值: 修改新变量，不影响原变量

原因: 复制的是值本身

引用类型的对象: 通过新变量修改对象，等效于直接修改原对象。新旧变量都受影响。

原因: 仅复制的是地址值，原对象始终只有一个

新旧变量使用相同的地址值，引用同一个对象

任何一方修改对象，另一方都同时受影响。

遍历:

什么是: 对数组中每个元素执行相同的操作

何时: 只要对数组中每个元素执行相同的操作

如何: for(var i=0;i<arr.length;i++){

arr[i] //当前元素

}