JavaScript 第三天

循环和数组

学习目标

Learning Objectives

1. 掌握循环语句，让程序具备重复执行能力
2. 掌握数组声明及访问的语法

目录

Contents

循环-for

- for循环基本使用
- 退出循环
- 循环嵌套

1.1 for 循环-基本使用

目标：掌握for循环重复执行某些代码

1. for循环语法

- 也是重复执行代码
- 好处：把声明起始值、循环条件、变化值写到一起，让人一目了然

```
for (声明记录循环次数的变量; 循环条件; 变化值) {

    循环体
}
```

练习

循环练习

1. 利用for循环输出1~100岁

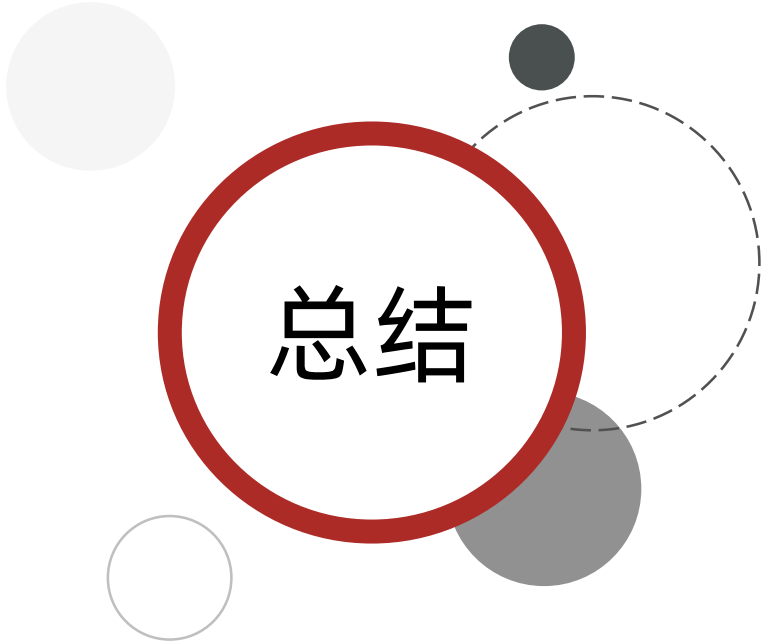
2. 求1-100之间所有的偶数和

3. 页面中打印5个小星星 ★★★★

4. for循环的最大价值： 循环数组

需求： 请将 数组 ['马超', '赵云', '张飞', '关羽', '黄忠'] 依次打印出来

循环小结



1. for循环和while循环有什么区别呢：
   - 当如果明确了循环的次数的时候推荐使用for循环
   - 当不明确循环的次数的时候推荐使用while循环

循环-for

- for循环基本使用
- 退出循环
- 循环嵌套

1.2 循环退出

目标： 能说出continue和break的区别

- 循环结束：
  - continue：结束本次循环，继续下次循环
  - break：跳出所在的循环

循环-for

- for循环基本使用
- 退出循环
- 循环嵌套

1.3 for 循环嵌套

目标：掌握for循环重复执行某些代码

for 循环嵌套

```
for (外部声明记录循环次数的变量; 循环条件; 变化值) {
    for (内部声明记录循环次数的变量; 循环条件; 变化值) {
        循环体
    }
}
```

- 一个循环里再套一个循环，一般用在for循环里

1.3 for 循环嵌套

目标：掌握for循环重复执行某些代码

for 循环嵌套-应用

计算： 假如每天记住5个单词，3天后一共能记住多少单词？

拆解：

- 第一天： 5个单词
- 第二天： 5个单词
- 第三天： 5个单词

```
第1天
记住第1个单词
记住第2个单词
记住第3个单词
记住第4个单词
记住第5个单词
第2天
记住第1个单词
记住第2个单词
记住第3个单词
记住第4个单词
记住第5个单词
第3天
记住第1个单词
记住第2个单词
记住第3个单词
记住第4个单词
记住第5个单词
```

练习

打印5行5列的星星

需求： 页面中打印出5行5列的星星

分析：

①：利用双重for循环来做

②：外层循环控制打印行，内层循环控制每行打印几个（列）

升级版本：

用户输入行数和列数，打印对应的星星！

练习

打印倒三角形星星

需求：如图所示

分析：

①：利用双重for循环来做

②：外层循环控制打印行，内层循环控制每行打印几个（列）

③：内层循环的个数跟第几行是一一对应的

练习

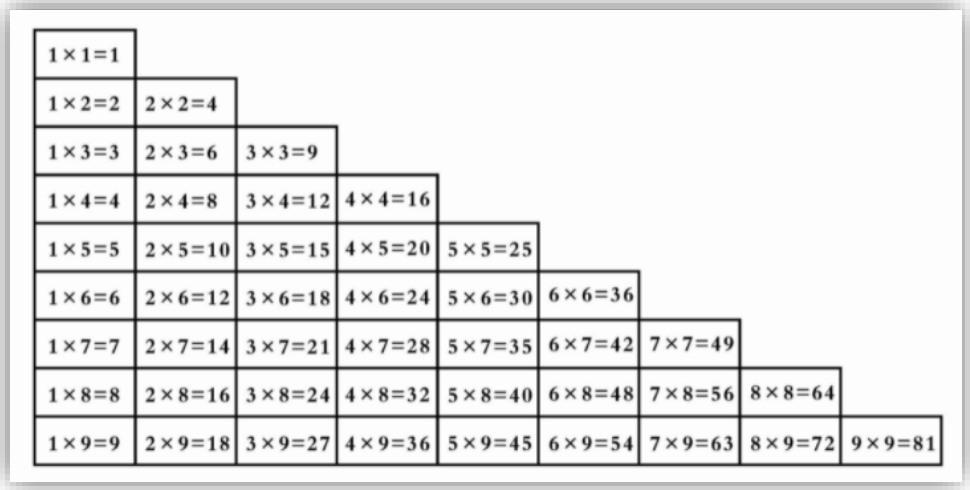
九九乘法表

需求：如图所示

分析：

①：只需要把刚才倒三角形星星做改动

②： ★ 换成 1 x 1 = 2 格式



1×1=1								
1×2=2	2×2=4							
1×3=3	2×3=6	3×3=9						
1×4=4	2×4=8	3×4=12	4×4=16					
1×5=5	2×5=10	3×5=15	4×5=20	5×5=25				
1×6=6	2×6=12	3×6=18	4×6=24	5×6=30	6×6=36			
1×7=7	2×7=14	3×7=21	4×7=28	5×7=35	6×7=42	7×7=49		
1×8=8	2×8=16	3×8=24	4×8=32	5×8=40	6×8=48	7×8=56	8×8=64	
1×9=9	2×9=18	3×9=27	4×9=36	5×9=45	6×9=54	7×9=63	8×9=72	9×9=81

目录

Contents

数组

- 数组是什么
- 数组的基本使用
- 操作数组
- 数组案例

2.1 数组是什么

目标：能说出数组是什么

- 数组(Array)是一种可以按顺序保存数据的数据类型
- 为什么要数组？
  - 思考：如果我想保存一个班里5个人的姓名怎么办？
  - 如果有多个数据可以用数组保存起来

数组

- 数组是什么
- 数组的基本使用
- 操作数组
- 数组案例

2.2 数组的基本使用

目标：能够声明数组并且能够获取里面的数据

1. 声明语法

```
let 数组名 = [数据1, 数据2, ..., 数据n]
```

- 例

```
let names = ['小明', '小刚', '小红', '小丽', '小米']
```

- 数组是按顺序保存，所以每个数据都有自己的编号
- 计算机中的编号从0开始，所以小明的编号为0，小刚编号为1，以此类推
- 在数组中，数据的编号也叫**索引或下标**
- 数组可以存储任意类型的数据

2.2 数组的基本使用

目标：能够声明数组并且能够获取里面的数据

2. 取值语法

```
数组名[下标]
```

- 例

```
let names = ['小明', '小刚', '小红', '小丽', '小米']
names[0]  // 小明
names[1]  // 小刚
```

- 通过下标取数据
- 取出来是什么类型的，就根据这种类型特点来访问

2.2 数组的基本使用

目标：掌握数组，把一堆数据有序管理起来

3. 一些术语：

- 元素：数组中保存的每个数据都叫数组元素
- 下标：数组中数据的编号
- 长度：数组中数据的个数，通过数组的length属性获得

```
let names = ['小明', '小刚', '小红', '小丽', '小米']
console.log(names[0])  // 小明
console.log(names[1]) // 小刚
console.log(names.length) //  5
```

2.2 数组的基本使用

目标：能够遍历输出数组里面的元素

4. 遍历数组：

用循环把数组中每个元素都访问到,一般会用for循环遍历

- 语法：

```
for (let i = 0; i < 数组名.length; i++) {
    数组名[i]
}
```

- 例

```
let nums = [10, 20, 30, 40, 50]
for (let i = 0; i < nums.length; i++) {
    document.write(nums[i])
}
```

案例

数组求和

需求：求数组 [2,6,1,7, 4] 里面所有元素的和以及平均值

分析：

①：声明一个求和变量 sum。

②：遍历这个数组，把里面每个数组元素加到 sum 里面。

③：用求和变量 sum 除以数组的长度就可以得到数组的平均值。

案例

数组求最大值和最小值

需求：求数组 [2,6,1,77,52,25,7] 中的最大值

分析：

①：声明一个保存最大元素的变量 max。

②：默认最大值可以取数组中的第一个元素。

③：遍历这个数组，把里面每个数组元素和 max 相比较。

④：如果这个数组元素大于max 就把这个数组元素存到 max 里面，否则继续下一轮比较。

⑤：最后输出这个 max

拓展：

自己求一下最小值

数组

- 数组是什么
- 数组的基本使用
- 操作数组
- 数组案例

2.3 操作数组

数组本质是数据集合，操作数据无非就是 增 删 改 查 语法：

查询数组数据

数组[下标]

或者我们称为访问数组数据

查

改

重新赋值

数组[下标] = 新值

数组添加新的数据

arr.push(新增的内容)

arr.unshift(新增的内容)

增

删

删除数组中数据

arr.pop()

arr.shift()

arr.splice(操作的下标,删除的个数)

目标：掌握利用push向数组添加元素(数据)

1. 数组增加新的数据

数组.push() 方法将一个或多个元素添加到数组的末尾，并返回该数组的新长度 (重点)

语法：

```
arr.push(元素1, ..., 元素n)
```

例如：

```
let arr = ['red', 'green']
arr.push('pink')
console.log(arr)  //  ['red', 'green', 'pink']
```

目标：掌握利用push向数组添加元素(数据)

1. 数组增加新的数据

数组.push() 方法将一个或多个元素添加到数组的末尾，并返回该数组的新长度

语法：

```
arr.push(元素1, ..., 元素n)
```

例如：

```
let arr = ['red', 'green']
arr.push('pink', 'hotpink')
console.log(arr)  //  ['red', 'green', 'pink', 'hotpink']
```

目标：掌握利用push向数组添加元素(数据)

1. 数组增加新的数据

arr.unshift(新增的内容) 方法将一个或多个元素添加到数组的**开头**，并返回该数组的新长度

语法：

```
arr.unshift(元素1, ..., 元素n)
```

例如：

```
let arr = ['red', 'green']
arr.unshift('pink')
console.log(arr)  //  ['pink', 'red', 'green']
```

目标：掌握利用push向数组添加元素(数据)

1. 数组增加新的数据

arr.unshift(新增的内容) 方法将一个或多个元素添加到数组的**开头**，并返回该数组的新长度

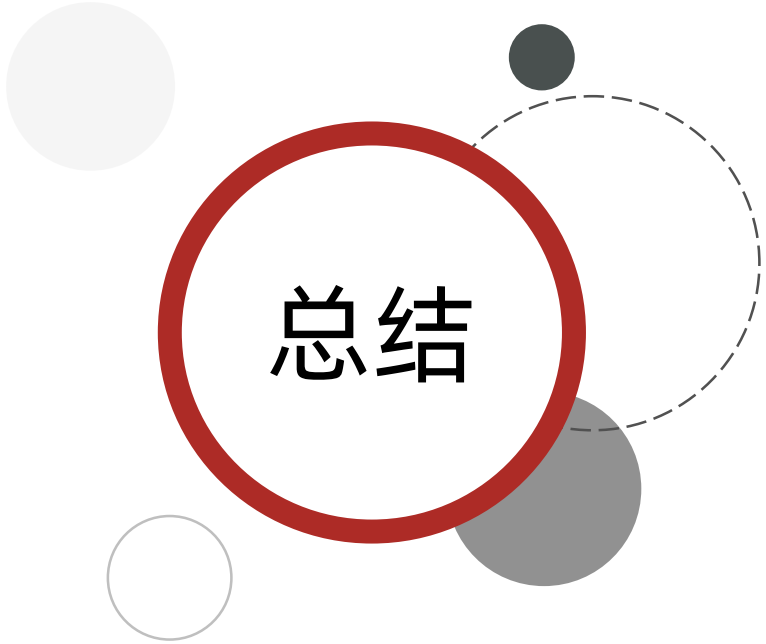
语法：

```
arr.unshift(元素1, ..., 元素n)
```

例如：

```
let arr = ['red', 'green']
arr.unshift('pink', 'hotpink')
console.log(arr)  //  ['pink', 'hotpink','red', 'green' ]
```

数组增加新的数据小结



1. 想要数组末尾增加数据元素利用那个方法？
   - arr.push()
   - 可以添加一个或者多个数组元素
   - 返回的是数组长度
2. 想要数组开头增加数据元素利用那个方法？
   - arr.unshift()
   - 可以添加一个或者多个数组元素
   - 返回的是数组长度
3. 重点记住那个？
   - arr.push()

案例

数组筛选

需求：将数组 [2, 0, 6, 1, 77, 0, 52, 0, 25, 7] 中大于等于 10 的元素选出来，放入新数组

分析：

①：声明一个新的数组用于存放新数据newArr

②：遍历原来的旧数组， 找出大于等于 10 的元素

③：依次追加给新数组 newArr

案例

数组去0案例

需求：将数组 [2, 0, 6, 1, 77, 0, 52, 0, 25, 7] 中的 0 去掉后，形成一个不包含 0 的新数组

分析：

①：声明一个新的数组用于存放新数据newArr

②：遍历原来的旧数组， 找出不等于0的元素

③：依次追加给新数组 newArr

2.3 操作数组

数组本质是数据集合，操作数据无非就是 增 删 改 查 语法：

查询数组数据

数组[下标]

或者我们称为访问数组数据

查

改

重新赋值

数组[下标] = 新值

数组添加新的数据

arr.push(新增的内容)

arr.unshift(新增的内容)

增

删

删除数组中数据

arr.pop()

arr.shift()

arr.splice(操作的下标,删除的个数)

目标：能够利用splice删除数组元素(数据)

2. 数组删除元素

数组. pop() 方法从数组中删除最后一个元素，并返回该元素的值

语法：

```
arr.pop()
```

例如：

```
let arr = ['red', 'green']
arr.pop()
console.log(arr)  //  ['red']
```

目标：能够利用splice删除数组元素(数据)

2. 数组删除元素

数组. shift() 方法从数组中删除第一个元素，并返回该元素的值

语法：

```
arr.shift()
```

例如：

```
let arr = ['red', 'green']
arr.shift()
console.log(arr)  //  ['green']
```

目标：能够利用splice删除数组元素(数据)

2. 数组删除元素

数组. splice() 方法 删除指定元素

语法：

```
arr.splice(start, deleteCount)

arr.splice(起始位置, 删除几个元素)
```

解释：

- start 起始位置：
  - 指定修改的开始位置（从0计数）
- deleteCount:
  - 表示要移除的数组元素的个数
  - 可选的。 如果省略则默认从指定的起始位置删除到最后

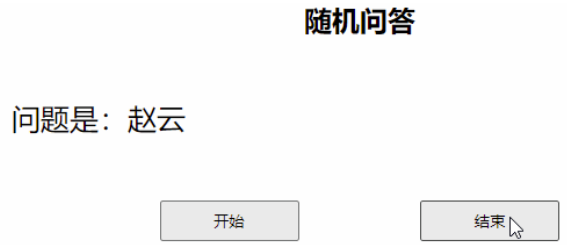
目标：能够利用splice删除数组元素(数据)

2. 数组删除元素

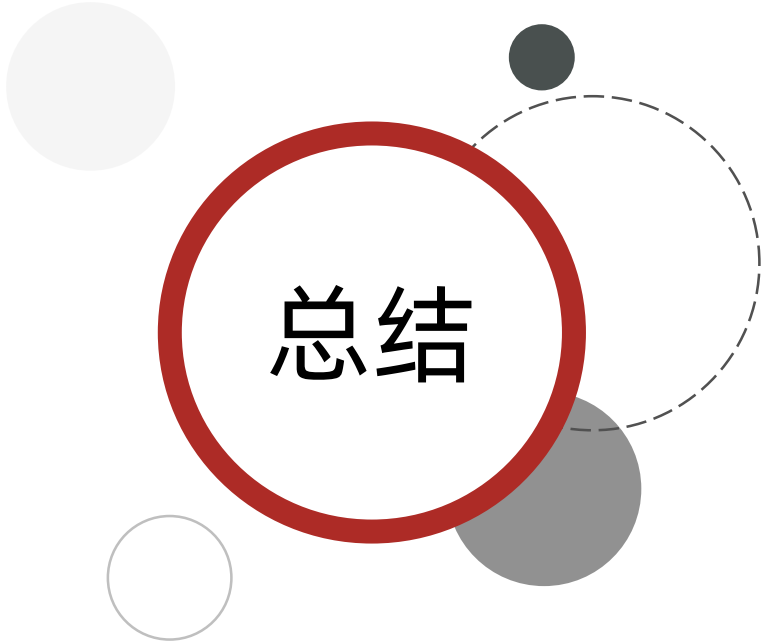
删除元素的使用场景：

1. 随机抽奖，中奖的用户就需要从数组里面删除，不允许重复抽奖
2. 点击删除按钮，相关的数据会从商品数据中删除

后期课程我们会用到删除操作，特别是**splice**



数组删除元素小结



1. 想要数组末尾删除1个数据元素利用那个方法？带参数吗？
   - arr.pop()
   - 不带参数
   - 返回值是删除的元素
2. 想要数组开头删除1个数据元素利用那个方法？带参数吗？
   - arr.shift()
   - 不带参数
   - 返回值是删除的元素
3. 想要指定删除数组元素用那个？开发常用吗？有那些使用场景？
   - arr.splice(起始位置, 删除的个数)
   - 开发很常用，比如随机抽奖，比如删除指定商品等等

案例

冒泡排序

冒泡排序是一种简单的排序算法。

它重复地走访过要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。

这个算法的名字由来是因为越小的元素会经由交换慢慢“浮”到数列的顶端。

比如数组 [2,3,1,4,5] 经过排序成为了 [1,2,3,4,5]　或者 [5,4,3,2,1]

案例

冒泡排序

分析：

1. 一共需要的趟数 我们用外层for 循环

5个数据我们一共需要走4趟

长度就是 数组长度 减去 1 arr.length - 1

2.每一趟交换次数 我们 用里层 for循环

第一趟 交换 4次

第二趟 交换 3次

第三趟 交换 2次

第四趟 交换 1次

长度就是 数组长度 减去 次数

但是我们次数 是从 0次开始的 所以 最终 arr.length - i - 1

3. 交换2个变量就好了

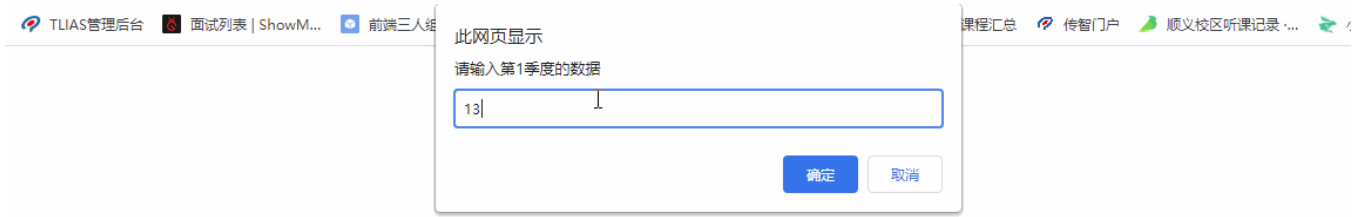
目录

Contents

案例

根据数据生成柱形图

需求： 用户输入四个季度的数据，可以生成柱形图



案例

根据数据生成柱形图

需求： 用户输入四个季度的数据，可以生成柱形图

分析：

①：需要输入4次，所以可以把4个数据放到一个数组里面

- 利用循环，弹出4次框，同时存到数组里面

②：遍历改数组，根据数据生成4个柱形图，渲染打印到页面中

- 柱形图就是div盒子，设置宽度固定，高度是用户输入的数据
- div里面包含显示的数字和 第n季度

今日复习路线

1. 晚自习回来每个同学先必须xmind梳理今日知识点 (md 笔记也行)
2. 需要把今天的所有案例，按照书写顺序写一遍。
3. 独立书写今日作业
4. 每日一句鼓励自己的话：

我命由我不由天

传智教育旗下高端IT教育品牌