# 初识数组

在开发中，经常需要使用变量保存一批相关联的数据，若使用前面学习的方式，则需要定义很多个变量分别保存这些数据，这种明显的弊端会给开发和管理带来困难。此时可以使用数组类型的变量保存这样的数据。

1. 数组是存储一系列值的变量集合，它是由一个或多个数组元素组成的，各元素之间使用逗号“，”分隔。每个数组元素由“下标”和“值”构成。其中，“下标”也可称为“索引”，以数字表示，默认情况下从0开始依次递增，用于识别元素；“值”为元素的内容，可以是任意类型的数据（因为JavaScript中的数组中可以存放不同数据类型的数据），如数值型、字符型、数组、对象等。
2. 数组还可以根据维数划分为一维数组、二维数组、三维数组等多维数组。

# 创建数组

JavaScript中的数组中可以存放不同类型的数据。在JavaScript中创建数组有两 种方式。

1. 使用new Array( )关键字创建数组。其语法格式为：

**var 数组名=new Array(1,2,3) ;**

例如：



1. 使用[ ]创建数组。其语法格式为：

**var 数组名=[1,2,3] ;**

例如：



# 数组的基本操作

## 获取数组长度

1）数组对象可以调用length属性获取数组的长度。其语法格式为：

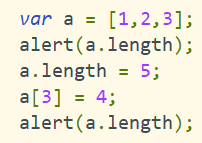
**数组名.length ;**

例如：

2）并且还可以利用length属性修改数组的长度，其语法格式为：

**数组名.length=长度 ;**

例如：



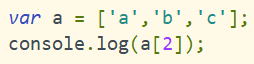
修改数组的length属性后，若length值大于数组中原来的元素个数，则没有值的数组元素会占用空存储位置。若length值等于数组中原来的元素个数，数组长度不变。若length值小于数组的原来元素个数，则多余的数组元素会被舍弃。

## 数组的访问和遍历

1. 访问数组元素

·如果想要看到数组中某个具体的元素，可以通过“数组名[下标]”的方式获取指

定下标的值。例如：



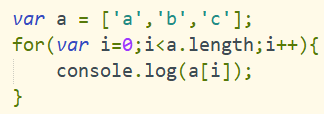
·如果直接输出数组（名称），则可以看到数组中所有的元素。例如：



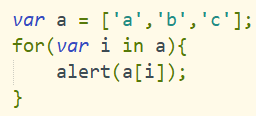
1. 遍历数组元素

遍历数组指的是访问数组中所有元素的操作，可以使用for循环语句、for…in语句for…of语句对数组进行遍历。

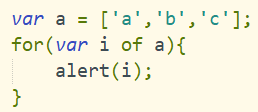
·使用for循环语句遍历数组：



·使用for…in语句遍历数组：



·使用for…of语句遍历数组：



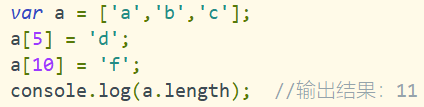
建议在遍历数组时，应对数组元素是否为空进行判断，做相应的处理。

## 数组元素的增删改

JavaScript中通过“数组名[下标]”的方式不仅可以访问数组中的元素，还可以完成数组元素的添加与修改。

1. **添加元素**

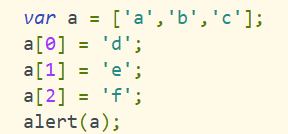
可以根据实际需要，通过自定义数组元素下标的方式添加元素。如下：



上述代码中，通过“数组名[下标]=值”的方式添加数组元素时，允许下标不按照数字顺序连续添加，其中未设置具体值的元素，会以空存储位置的形式存在。

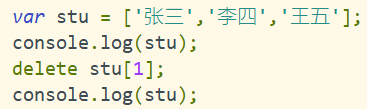
1. **修改元素**

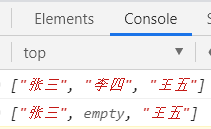
修改元素与添加元素的使用方式相同，不同的是修改元素是为已含有值的元素重新赋值。如下：



1. **删除元素**

根据实际情况，有时需要删除数组中某个元素值。此时，可以利用delete关键字删除该数组元素的值。如下：





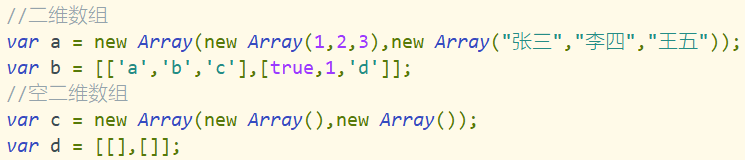
delete关键字只能删除数组中指定下标的元素值，删除后该元素依然会占用一个空存储位置。

# 常见二维数组操作

1. 在项目开发中，经常需要对多维数组进行操作。其中，二维数组是最常见的多维数组。
2. 虽然JavaScript没有限制数组的维数，但是在实际应用中，为了便于代码阅读、调试和维护，推荐使用三维及三维以下的数组保存数据。

## 创建二维数组

二维数组的创建只需将数组元素设置为数组即可。

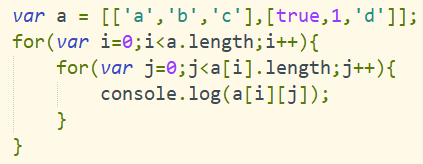


上述代码演示了使用Array对象和“[]”的方式创建二维数组。

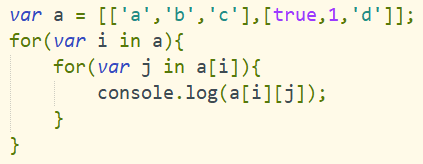
## 遍历二维数组

一维数组可以使用for、for…in、for…of语句进行遍历，二维数组也可以使用这些语句进行遍历，只需在遍历数组后，再次遍历数组的元素即可获取到二维数组的元素值。

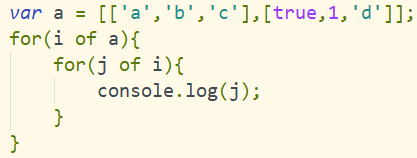
·使用for循环语句遍历二维数组



·使用for…in遍历二维数组



·使用for…of遍历二维数组



# 数组排序

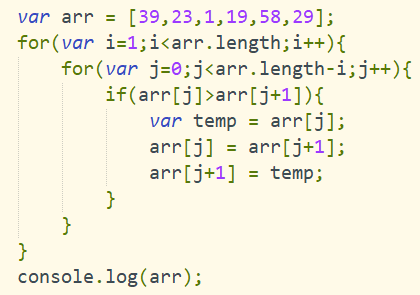
1. 开发中为了避免很多不必要的麻烦，在数组中进行定位时，经常需要将数组元素的值转为一个有序的排列。例如，使用二分查找法查找数据时，就必须是对一个有序排列的数组进行操作。

2）排序算法有很多，这里仅介绍冒泡排序和插入排序。

## 冒泡排序

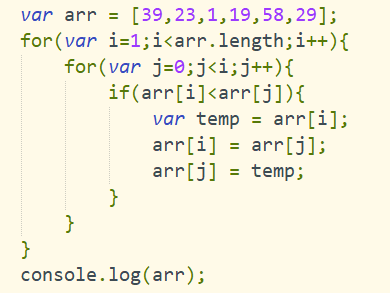
冒泡排序是较简单的排序算法，在冒泡排序中的过程中，按照要求从小到大排序或从大到小排序，不断比较数组中相邻两个元素的值，较小或较大的元素前移。

冒泡排序比较的轮数是数组长度减1，每轮比较的对数等于数组的长度减去当前的轮数。示例代码如下：



## 插入排序

插入排序时冒泡排序的优化，是一种直观的简单排序算法。它的实现原理是，通过构件有序数组元素的存储，对未排序的数组元素，在已排序的数组中从最后一个元素向第一个元素遍历，找到相应位置并插入。其中，待排序数组的第1个元素会被看作是一个有序数组，从第2个至最后一个元素会被看作是一个无序数组。



插入排序比较的次数与无序数组的长度相等，每次无序数组元素与有序数组元素中的所有元素进行比较，比较后找到对应位置插入，最后即可得到一个有序数组。

# 常用数组方法

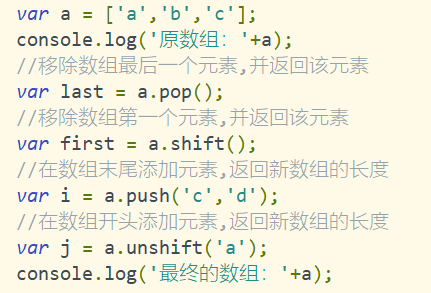
数组是JavaScript中最常用的数据类型之一，为此Array对象中提供了许多内置方法，如栈方法、检索方法、数组转字符串的方法等。

## 栈和队列方法

除了前面学习的添加与删除数组元素的方式外，还可以利用Array对象提供的方法，实现在数组的末尾或开头添加数组的新元素，或在数组的末尾或开头移出数组元素。利用这些方法可以模拟栈和队列的操作。

|  |  |
| --- | --- |
| 方法名称 | 功能描述 |
| push( ) | 在数组的末尾添加一个或多个元素，并返回数组的新长度 |
| unshift( ) | 在数组的开头添加一个或多个元素，并返回数组的新长度 |
| pop( ) | 从数组的末尾移出并返回一个元素，若是空数组则返回undefined |
| shift( ) | 从数组的开头移出并返回一个元素，若是空数组则返回undefined |

注意，push( )和unshift( )方法的返回值是新数组的长度，而pop( )和shift( )方法返回的是移出的数组元素。

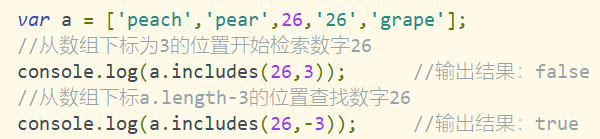


## 检索方法

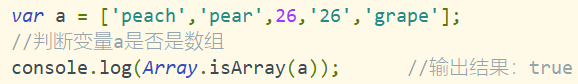
在开发中，若要检测给定的值是否是数组，或是查找指定的元素在数组中的位置，则可以利用Array对象提供的检索方法。

|  |  |
| --- | --- |
| 方法名称 | 功能描述 |
| includes( ) | 判断数组中是否包含某元素，有则返回true，否则返回false |
| Array.isArray( ) | 判断数据是否是数组，是返回true，否则返回false |
| indexOf( ) | 返回在数组中找到给定值的第一个索引，如果不存在则返回-1 |
| lastIndexOf( ) | 返回指定元素在数组中的最后一个的索引，如果不存在则返回-1 |

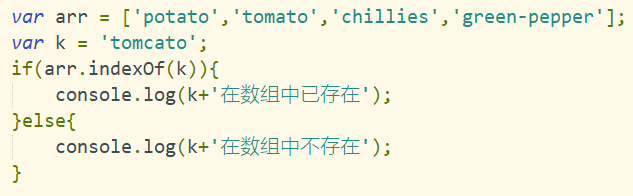
·includes( )方法的使用，示例如下：



·Array.isArray( )方法的使用，示例如下：



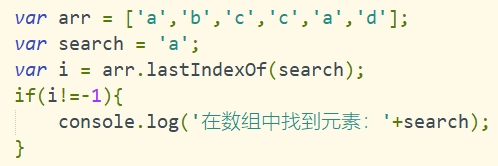
·indexOf( )方法的使用，示例如下：



indexOf( )方法的第二个参数用于指定开始查找的下标，当其值大于或等于数组长度

时，程序不会在数组中查找，直接返回-1；当其值为负数时，查找的下标位置等于数组长度加上这个负数，若结果仍是小于0的数，则检索整个数组。

·lastIndexOf( )方法的使用，示例如下：



与indexOf( )检索方式不同的是，lastIndexOf( )方法默认逆向检索，即从数组的末尾向数组的开头检索。

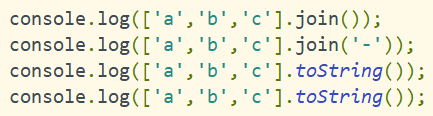
lastIndexOf( )方法的第二个参数用于指定查找的下标，且由于其采用逆向的方式检索，因此当其值大于或等于数组长度时，则整个数组都会被查找；当其值为负数时，则索引位置等于数组程度加上给定的负数，若其值仍为负数，则直接返回-1。

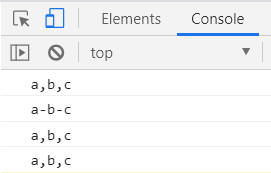
## 数组转字符串

在项目开发中，若需要将数组转换为字符串，则可以利用JavaScript提供的join( )和toString( )方法实现。

|  |  |
| --- | --- |
| 方法名称 | 功能描述 |
| join( ) | 将数组的所有元素连接到一个字符串中，默认使用“，”连接，也可以指定连接符号。 |
| toString( ) | 返回一个字符串，表示指定的数组及其元素，只能使用“，”连接 |

示例如下：





注意：当数组元素为undefined、null或空数组时，对应的元素会被转换为空字符串。

## 其他方法

除了前面讲解的几个常用方法外，JavaScript还提供了很多其他常用的数组方法。例如数组合并、数组浅拷贝、颠倒数组元素的顺序等。

|  |  |
| --- | --- |
| 方法名称 | 功能描述 |
| sort( ) | 对数组的元素进行排序，并返回数组 |
| fill( ) | 用一个固定值填充数组中指定下标范围内的全部元素 |
| reverse( ) | 颠倒数组中元素的位置 |
| splice( ) | 对一个数组在指定下标范围内删除或添加元素 |
| slice( ) | 从一个数组的指定下标范围内拷贝数组到一个新数组中 |
| concat( ) | 返回一个合并两个或多个数组后的新数组 |

上述方法中，slice( )和concat( )方法在执行后会返回一个新的数组，不会对原数组产生影响，剩余的方法在执行后皆会对原数组产生影响（以上方法的使用还请参考教材书，如果教材书上没有，则查找相关资料）。