数组

一日三省且温故知新

索引数组

Index Array

01

1. 索引数组

1.1、什么是：内存中连续存储多个数据的数据结构，再起一个统一的名字

1.2、为什么：普通的变量只能存储一个数据程序=数据结构+算法

算法: 解决问题的步骤

数据结构: 数据在内存中的存储结构

好的数据结构可以极大的提高程序的执行效率

1.3、何时：只要存储多个连续的数据

1. 索引数组

1.4、创建

1、创建空数组: 2种

1. 数组直接量: var arr=[];

2. 用new: var arr=new Array();

新建 数组

何时: 在创建数组时，还不知道数组中的元素内容时

1. 索引数组

1.4、创建-例

```
var arr1 = [ ];                    //定义一个不包含元素的数组
var arr2 = [97, 85, 79];           //定义一个包含3个元素的数组
var arr3 = new Array();            //定义一个不包含元素的数组
var arr4 = new Array("Tom", "Mary", "John");
//定义一个三个字符串元素的数组
```

1. 索引数组

1.4、创建-创建数组同时初始化

1. 数组直接量: var arr=[元素1,元素2,...];

2. 用new: var arr=new Array(元素1,元素2,...);

何时: 在创建数组时，已经知道数组的元素内容

```
var array = [4500, 5500, 5000];
var array = new Array('市场部', '研发部', '运营部');
```

1. 索引数组

1.4、创建-先声明空数组，再添加元素

```
var empArray = [ ];
empArray[0] = 'Scott';
empArray[1] = 'Smith';
```

```
var mArray = new Array();
mArray[0] = '三国志';
mArray[2] = 195;
mArray[5] = true;
```

混合元素类型数组

1. 索引数组

1.5、访问数组中的元素

元素: 数组中每个数据都是一个元素
如何访问: 下标: 数组中唯一标识每个元素存储位置的序号
特点: 从0开始,连续不重复
何时: 只要访问数组元素，只能用下标
如何: 数组名[i]——用法和单个变量完全一样!

1. 索引数组

1.6、数组GET操作与SET操作

设置数组元素的值——SET

```
var scores = [95, 88, 100];
scores[2] = 98;     //将值为100的元素重新赋值为98
scores[3] = 75;     //在数组尾部添加一个新的元素
```

下标从0开始，最大到length-1

1. 索引数组

获取数组元素的值——GET

```
var cities = new Array( ‘南京’ , ‘杭州’ , ‘青岛’ );
console.log( cities[1] );                    //杭州
console.log( cities[3] );                    //undefined
```

不会抛出数组下标越界异常

1. 索引数组

1.7、访问数组中的元素:

数组的length属性: 记录了数组中理论上的元素个数

length属性的值永远是最大下标+1

```
var arr4 = new Array(10);
console.log( arr4.length );          //长度为10
```

```
var arr5 = [ ];                      //长度为0
arr5[0] = 87;                        //长度变为1
arr5[3] = 98;                        //长度变为4
```

1. 索引数组

1.8、数组的遍历

遍历数组元素，通常选择for循环语句，元素的下标作循环变量

```
var nums = [50, 90, 20, 10];
for( var i=0; i< nums.length; i++){
        nums[ i ] += 10;
}
```

元素下标从0开始，到length-1结束

1. 索引数组

1.9、固定套路

1.获得数组最后一个元素: arr[arr.length-1]

2.获得倒数第n个元素的位置: arr[arr.length-n]

3.数组缩容: 减小arr.length的数值，会删除结尾的多余元素。

4.遍历数组: 依次访问数组中每个元素，对每个元素执行相同的操作

```
for(var i=0;i<arr.length;i++){
        arr[i]//当前正在遍历的元素
}
```

1. 索引数组

1.10、特殊: 三个不限制:

1. 不限制数组的元素个数:长度可变

2. 不限制下标越界:

获取元素值: 不报错！返回undefined

修改元素值: 不报错! 自动在指定位置创建新元素，并且自动修改length属性为最大下标+1

如果下标不连续的数组——稀疏数组

3. 不限制元素的数据类型

关联数组

Associative Array

02

2. 关联数组:

索引数组: 下标为数字的数组

2.1什么是关联数组: 可自定义下标名称的数组

2.2为什么: 索引数组中的数字下标没有明确的意义

2.3何时: 只要希望每个元素都有专门的名称时

2.4如何: 2步

1. 创建空数组
2. 向空数组中添加新元素，并自定义下标名称

2. 关联数组:

2.4、关联数组的创建方式

```
var bookInfo = [ ];
bookInfo['bookName'] = '西游记' ;
bookInfo['price'] = 35.5 ;
```

由于关联数组的 length 属性值无法获取其中元素的数量，所以遍历关联数组只能使用 for..in 循环

2. 关联数组:

2.5遍历关联数组: for in循环

```
for(var key in hash){
        key//只是元素的下标名
        hash[key]//当前元素值
  }
```

2. 关联数组:

2.6 索引数组与关联数组的对比

索引数组	VS	关联数组:
1. 以字符串输出		不能用字符串输出
2. 下标是数字		下标是自定义的字符串
3. length属性有效		length属性失效(=0)
4. 访问元素,都用数组名["下标"]		
5. 可用for循环遍历		不能用for循环遍历——for in

2. 关联数组:

2.6 索引数组与关联数组的对比

查找:	索引	hash数组
	遍历	不用遍历
	受存储位置影响	和存储位置无关
	受数组元素个数影响	和数组中元素个数无关

总结: 今后只要希望快速查找元素时，就用hash数组

小结

数组

- 索引数组
- 关联数组