JS数组/API函数

2020年4月6日 18:25

索引数组：（下表为数字的数组）

创建：创建时未知数组里元素个数时采用第二种创建方式

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

```
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
```

数组GET创建操作与SET获取操作：

```
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
```

- 数组中没有这个下标时则表示添加一个新元素
- 数组中有对应下标元素，则表示替换原数组中的元素
- 下标从0开始到数组长度减一

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

Length属性

Var arr4=new Array（n）长度为n

```
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
```

- 1，2位置是空的，但是他还是有长度的

数组的遍历：

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

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
}
```

数组的三个不限制：

1. 不限制数组的元素个数:长度可变

关联数组：（可自定义下标名称的数组）

如何：

1. 创建空数组
2. 向空数组中添加新元素，并自定义下表名称

```
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

关联数组遍历：

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

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

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

注：只要希望快速查找元素时，就用hash数组

数组API函数：

（一）、数组转字符串

- String（arr）：将arr中的每个元素转为字符串，用逗号分隔

  固定套路: 对数组拍照: 用于鉴别是否数组被修改过

- Arr.join（‘连接符’）：将arr中每个元素转为字符串，用自定义的连接符连接

```
//将字符拼接为单词
var chars=["H","e","l","l","o"];
console.log(chars.join(""));          //Hello
```

固定套路：

1. 判断数组是否为空
2. 将数组转化为页面元素的内容

1.将字符组成单词: chars.join("")->无缝拼接

扩展: 判断数组是空数组: arr.join("")==""

数组的三个不限制：

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

获取元素值 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

2. 将数组转化为页面元素的内容

（二）、拼接和选取

拼接

不直接修改原数组，而返回新数组

- concat（）拼接两个或更多数组，并返回结果
- Var newArr=arr.concat（值1，值2，arr2，值3……）

注意：其中arr2的元素会被先打散，再拼接

```
var arr1 = [90, 91, 92];
var arr2 = [80, 81];
var arr3 = [70, 71, 72,73];
var arr4 = arr1.concat(50, 60, arr2, arr3);

console.log( arr1 );          //现有数组值不变
console.log( arr4 );
```

选取

不直接修改原数组，而返回新数组

- slice（）返回现有数组的一个子数组
- Var subArr=arr.slice（starti,endi+1）如（4，6）表示从第4个到第5个元素

注意：凡是两个参数都是下标函数，都是：含头不含尾

```
var arr1 = [10, 20, 30, 40, 50];
var arr2 = arr1.slice(1, 4);          //20,30,40
var arr3 = arr1.slice(2);             //30,40,50
var arr4 = arr1.slice(-4, -2);        //20,30

console.log(arr1);                    //现有数组元素不变
```

只有一个下标函数时是指从那里一直到数组末尾

1. 一直选取到结尾: 可省略第二个参数

2. 如果选取的元素离结尾近: 可用倒数下标:

arr.slice(arr.length-n,arr.length-m+1)
可简写为:arr.slice(-n,-m+1);

3. 复制数组:

arr.slice(0,arr.length);
可简写为:arr.slice();

删除

直接修改原数组

- Arr.splice（starti，n）删除从starti开始的n个元素

```
var arr1 = [10, 20, 30, 40, 50];
var arr2 = arr1.splice(2, 1);
//var arr2 = arr1.splice(2, 2, 21,22,23);
//var arr2 = arr1.splice(2, 2, [91,92,93]);
```

删除从下标为2开始的2个元素，并将后面的三个元素添加到那个位置

插入

- Arr.splice（starti，0，值1，值二……）
- 在arr中starti位置，插入新值1，值二，原starti位置及其之后的值

向后顺延

替换

Arr.splice（starti，n，值1，值2）

删除旧的，插入新的。先删除starti位置的n个值，再在starti位置插入新值

注意：删除的元素个数和插入的新元素个数不必一致

注意：选取是slice（），删除插入替换是splice（）

（三）、颠倒数组

- reverse（）颠倒数组中元素的顺序
- Arr.reverse（）

```
var arr1 = [10, 20, 30, 40, 50];
arr1.reverse();
```

注意：只负责颠倒数组，不负责排序

（四）、排序

- Arr.sort（）将元素按从小到大排序
- 默认将所有元素转换为字符串再排序

只能排列字符串类型的元素