JavaScript

千锋教育
QIAN FENG EDUCATION

做真实的自己，用良心做教育

张骏
H5教学部

数组

课程概要：

- 创建对象
- 什么是数组
- 创建数组
- 引用类型
- new关键字
- 数组参数传递
- null
- 数组操作(GET/SET)
- 获取数组的长度
- 数组的遍历
- 数组API

创建对象

ECMAScript 中的对象其实就是一组数据和功能的集合。对象可以通过执行new操作符后跟要创建的对象类型的名称来创建。而创建Object类型的实例并为其添加属性和(或)方法，就可以创建自定义对象：

```
var o = new Object();
o.name = "zhangsan";
o.age = 18;
```

直接量语法创建：

```
var obj = {};
```

什么是数组

- 多个元素组成的集合 -- 在一个变量名中储存多个值。
- javascript数组中元素的数据类型可以相同，也可以不同。
- 可以通过元素所在位置的顺序号（下标）做标识来访问每一个元素（下标从0开始，最大到元素的个数-1）
- 前面学习的if、if-else、switch、循环解决的都是流程问题、既算法问题。

数组（Array），就是一种很常用的保存批量数据的数据结构，所谓数据结构，就是把数据与数据间的关系按照特定的结构来保存。

设计合理的数据结构是解决问题的前提。

定义一维数组

- 可以使用如下四中格式定义一个数组变量：

```
var arr1=[];  //定义一个不包含元素的数组
var arr2=[1,2,3];  //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” ,” marry” ,” john” );
//定义一个包含三个字符串元素的数组
```

初始化数组

- 数组中的元素可以在定义时初始化

```
var arr1=[1,2,3];
var arr2=new Array('苹果','橘子','西瓜');
```

- 也可以先声明一个空数组，随后在向其添加元素

```
var  empArray=[];
empArray[0]='Tom';
empArray[1]='Marray';
```

```
var empArray =new Array();
empArray[0]=1;
empArray[2]=true;
```

引用类型

引用类型通常叫做类（class），也就是说，遇到引用值，所处理的就是对象。
本教程会讨论大量的 ECMAScript 预定义引用类型。
从现在起，将重点讨论与已经讨论过的原始类型紧密相关的引用类型。
注意：从传统意义上来说，ECMAScript 并不真正具有类。事实上，除了说明不存在类，在 ECMA–262 中根本没有出现“类”这个词。ECMAScript 定义了“对象定义”，逻辑上等价于其他程序设计语言中的类。

数组是引用类型的对象

- 引用类型：值不保存在变量本地的数据类型

var n=100;

- 值类型：保存在栈中，速度快。

var arr=new Array('北京','哈尔滨');

引用类型：数据保存在堆中，栈中只有内存的编号，要通过此编号到堆中查找真正的数据

new关键字

- 什么是对象？
  - js中的对象，指内存中集中保存一组相关数据和功能的整体。

- 对象中保存了相关的数据和对数据的常用操作方法。
- 如何创建对象？对象都是用new关键字创建的。
- 什么是new关键字？new专门用于在内存中开辟一块储存空间，然后返回存储空间的地址
- 使用new创建出的都是引用类型的对象。

数组参数传递

- 数组赋值给变量时，其实赋值的是地址。
- 数组作为参数，也是按值传递，传递的是数组的地址。

```
function fun(arr){
        arr[0]=0;
}
var arr=[1,2,3];
fun(arr);
console.log(arr);
```

null

- 什么是null？
  - null专门表示一个变量不能再指向任何对象的地址。
- null与undefined：
  - 共同点：
    - 都是原始类型，保存在栈中变量本地
  - 不同点：
    - undefined：表示变量声明过但未赋值。是所有未赋值变量的默认值。一般不主动使用。例如 var a; //a被自动赋值为undefined
    - null：表示一个变量将来可能指向一个对象，但目前暂时什么都没指向。一般用于主动释放指向对象的引用。

    例如：var emps =[“Tom”,” Marray”];

    emps=null;//释放指向数组的引用，清空掉栈中的编号，对中的数据没人用，被回收

数组操作(GET/SET)

- 设置数组元素的值--SET操作

```
var arr = [1,2,3];
arr[2] = 100;  //将值为3的元素重新赋值为100;
arr[3] = 99;  //在数组尾部添加一个新的元素;
```

- 获取数组元素的值--GET操作

```
var add  = new Array('Hello','World');
console.log(add[0]);  //Hello
```

获取数组的长度

- 使用length属性获取数组中元素的个数，既数组的长度;

```
var arr1 = [11,21,33];
console.log(arr1.length);  //长度为3;

var arr2 = new Array(10);
console.log(arr2.length);  //长度为10;
```

遍历数组元素

- 遍历数组元素，通过选择for循环语句，元素的下标做循环变量

```
var arr1 = [10,20,30,40];
for(var i = 0 ; i<arr1.length ; i++){
        console.log(arr1[i]);
}
```

- 也可以倒序、或者跳续遍历

```
for(var i = arr1.length-1 ; i>=0 ; i--){
        console.log(arr1[i]);
}
```

for...in声明

- for...in遍历数组

```
var nums = [10,21,32,24];
for(var i  in  nums ){
        console.log(nums[i]);
}
```

扩展：关联数组

- 索引数组：以0、1、2、3...数字作为下标
- 关联数组：以一个字符串作为下标

```
var stu = [];
stu['name'] = 'Tom';
stu['age'] = 20;
```

数组模拟栈

ECMAScript数组也提供了一种让数组的行为类似于其他数据结构的方法。具体来说数据可以表现的像栈一样，后者是一种可以限制插入和删除项的数据结构。栈是一种LIFO(Last-In-First-Out)的数据结构，也就是最新添加的项最早被移除。而栈中项的插入(叫做推入)和移除(叫做弹出)，只发生在一个位置——栈的顶部。ECMAScript为数组专门提供了push()和pop()方法。以便实现类似栈的行为。

Array.push()

- **语法**
- *array*.push(*value*, ...)
- **参数**
- *value, ...* 要添加到array尾部的值，可以是一个或多个。
- **返回值**
- 把指定的值添加到数组后的新长度。
- **描述**
- 方法push()将把它的参数顺次添加到array的尾部。它直接修改array,而不是创建——个新的数组。方法push()和方法pop()用数组提供先进后出栈的功能。

Array.pop()

- **语法**
- *array*.pop( )
- **返回值**
- array的最后一个元素。
- **描述**
- 方法pop()将删除array的最后一个元素，把数组长度减1，并且返回它删除的元素的值。如果数组已经为空，则pop()不改变数组，返回undefined。
- **示例**
- 方法pop()和方法push()可以提供先进后出(FILO)栈的功能。
- var stack = []; // 栈：[]
- stack.push(1, 2); // 栈: [1,2] 返回 2
- stack.pop( ); // 栈: [1] 返回 2
- stack.push([4,5]); // 栈: [1,[4,5]] 返回 2
- stack.pop( ) // 栈: [1] 返回 [4,5]

Array.concat()

- **语法**
- *array*.concat(*value*, ...)
- **参数**
- *value, ...* 要增加到array中的值，可以是任意多个。
- **返回值**
- 一个新数组
- **描述**
- 方法concat()将创建并返回一个新数组，这个数组是将所有参数都添加到array中生成的。它并不修改array。如果要进行concat()操作的参数是一个数组，那么添加的是数组中的元素，而不是数组。
- **示例**
- var a = [1,2,3];
- a.concat(4, 5) //返回 [1,2,3,4,5]

Array.join()

- **语法**
- *array*.join( )
- *array*.join(*separator*)
- **参数**
- *separator* 在返回的字符串中用于分隔数组元素的字符或字符串，它是可选的。如果省略了这个参数，用逗号作为分隔符。
- **返回值**
- 一个字符串，通过把array的每个元素转换成字符串，然后把这些字符串连接起来，在两个元素之间插入separator字符串而生成。
- **描述**
- 方法join()将把每个数组元素转换成一个字符串，然后把这些字符串连接起来，在两个元素之间插入指定的separator字符串。返回生成的字符串。
- 可以用String对象的split()方法执行相反的操作，即把一个字符串分割成数组元素。
- **示例**
- a = new Array(1, 2, 3, "testing");
- s = a.join("+"); // s 是字符串"1+2+3+testing"

Array.reverse()

- **语法**
- *array*.reverse( )
- **描述**
- Array对象的方法reverse()将颠倒数组中元素的顺序。它在原数组上实现这一操作，即重排指定的array的元素，但并不创建新数组。如果对array有多个引用，那么通过所有引用都可以看到数组元素的新顺序。
- **示例**
- a = new Array(1, 2, 3); // a[0] == 1, a[2] == 3;
- a.reverse( ); // Now a[0] == 3, a[2] == 1;

Array.shift()

- **语法**
- *array*.shift( )
- **返回值**
- 数组原来的第一个元素。
- **描述**
- 方法shift()将把array的第一个元素移出数组，返回那个元素的值，并且将余下的所有元素前移一位，以填补数组头部的空缺。如果数组是空的，shift()将不进行任何操作，返回undefined。注意，该方法不创建新数组，而是直接修改原有的数组。
- 方法shift()和方法Array.pop()相似，只不过它在数组头部操作，而不是在尾部操作。该方法常常和unshift()一起使用。
- **示例**
- var a = [1, [2,3], 4]
- a.shift( ); // 返回 1; a = [[2,3], 4]
- a.shift( ); // 返回 [2,3]; a = [4]

Array.slice()

- **语法**
- *array*.slice(*start*, *end*)
- **参数**
- *start*
- 数组片段开始处的数组下标。如果是负数，它声明从数组尾部开始算起的位置。 也就是说， -1指最后一个元素， -2指倒数第二个元素，以此类推。
- *end*
- 数组片段结束处的后一个元素的数组下标。如果没有指定这个参数 包含从start开始到数组结束的所有元素。如果这个参数是负数， 从数组尾部开始算起的元素。
- **返回值**
- 一个新数组，包含从start到end(不包括该元素)指定的array元素。
- **描述**
- 方法slice()将返回数组的一部分，或者说是一个子数组。返回的数组包含从start 开始到end之间的所有元素，但是不包括end所指的元素。如果没有指定end，返回的数组包含从start开始到原数组结尾的所有元素。
- 注意：该方法并不修改数组。如果想删除数组中的一段元素，应该使用方法Array.splice。
- **示例**
- var a = [1,2,3,4,5];
- a.slice(0,3); // 返回 [1,2,3]
- a.slice(3); // 返回 [4,5]
- a.slice(1,-1); // 返回 [2,3,4]

Array.sort()

- **示例**
- 下面的代码展示了如何编写按数字顺序，而不是按字母顺序对数组进行排序的比较函数：

```
// 按照数字顺序排序的排序函数
function numberorder(a, b) {
        return a - b;
 }
a = new Array(33, 4, 1111, 222);
a.sort( ); // 按照字母顺序的排序结果为: 1111, 222, 33, 4
a.sort(numberorder); // 按照数字顺序的排序结果为: 4, 33, 222, 1111
```

Array.toString()

- **语法**
- *array*.toString( )
- **返回值**
- array的字符串表示。
- **抛出**
- TypeError 调用该方法时，若对象不是Array，则抛出该异常。
- **描述**
- 数组的toString()方法将把数组转换成一个字符串，并且返回这个字符串。当数组用于字符串环境中时，JavaScript会调用这一方法将数组自动转换成一个字符串。但在某些情况下，需要明确地调用这个方法。
- toString()在把数组转换成字符串时，首先要将数组的每个元素都转换成字符串 (通过调用这些元素的toString()方法)。当每个元素都被转换成字符串时，它就以列表的形式输出这些字符串，字符串之间用逗号分隔。返回值与没有参数的join() 方法返回的字符串相同。

Array.unshift()

- **语法**
- *array*.unshift(*value*, ...)
- **参数**
- *value, ...* 要插入数组头部的一个或多个值。
- **返回值**
- 数组的新长度
- **描述**
- 方法unshift()将把它的参数插入array的头部，并将已经存在的元素顺次地移到较高的下标处，以便留出空间。该方法的第一个参数将成为数组新的元素0，如果还有第二个参数，它将成为新的元素1，以此类推。注意，unshift()不创建新数组，而是直接修改原有的数组。
- **示例**
- 方法unshift()通常和方法shift()一起使用。例如：
- var a = []; // a:[]
- a.unshift(1); // a:[1] 返回 1
- a.unshift(22); // a:[22,1] 返回 2
- a.shift( ); // a:[1] 返回 22
- a.unshift(33,[4,5]); // a:[33,[4,5],1] 返回 3

Array.splice( )

- **参数**
- *start*
- 开始插入和(或)删除的数组元素的下标。
- *deleteCount*
- 从start开始，包括start所指的元素在内要删除的元素个数。这个参数是可选的，如果没有指定它，splice()将删除从start开始到原数组结尾的所有元素。
- *value, ...*
- 要插人数组的零个或多个值，从start所指的下标处开始插入。
- **返回值**
- 如果从array中删除了元素，返回的是含有被删除的元素的数组。但是要注意，由于存在—个bug，因此在JavaScriptl.2的Netscape实现中，返回的并不总是数组。
- **描述**
- 方法splice()将删除从start开始(包括start所指的元素在内)的零个或多个元素，并且用参数列表中声明的一个或多个值来替换那些被删除的元素。位于插入或删除的元素之后的数组元素都会被移动，以保持它们与数组其他元素的连续性。注意，虽然splIce()方法与slice()方法名字相似，但作用不同，方法splice()直接修改数组。

二维数组

•二维数组：从整体上来看，是一个数组，只是其中的每个元素又是一个数组，既数组的数组;

```
var arr=[[1,2,3],[3,4,5],[6,7,8]];


var arr1=[
            ['北京','黑龙江','辽宁'],
            ['朝阳','海淀','丰台'],
            ['大连','沈阳','长春']
          ]
```

使用二维数组

- 访问二维数组中的某个元素

```
var arr=[[1,2,3],[3,4,5],[6,7,8]];
console.log(arr[0][0]);
arr[1][2] = 200;
```

- 遍历二维数组中的每一个元素----循环嵌套

```
for (var i = 0; i < arr.length; i++) {
    for (var j = 0; j < arr[i].length; j++) {
      document.write(arr[i][j]+" ");
    }
    document.write("<br/>");
  }
```

THANK YOU

千锋教育
QIANFENG EDUCATION

做真实的自己，用良心做教育