Day 06---数组

一、数组的概念

1.1 什么是数组

> 数组是指的数据的有序列表。

1. 数组中每个值称之为数组的一个元素。
2. 数组中的每个元素都有一个位置，这个位置称之为索引(下标、index)。数组的索引是从 0 开始的
3. 同一个数组中，元素的类型不做任何限制。也就是说，同一个数组中可以方法Number、String、Boolean、Object对象等等。可以同时放入任何的类型。甚至数组中的元素可以是另外一个数组(构成多维数组)。

1.2 JavaScript中数组的特点

> 虽然每种语言都有数组这种数据结构，但是JavaScript的数组相比他们有很大不同。

1. 数组长度可以动态改变。
2. 同一个数组中可以存储不同的数据类型。
3. 数据的有序集合
4. 每个数组都有一个length属性，表示的是数组中元素的个数

二、数组的创建

> 数组有两种基本创建方式：字面量方式和构造函数

2.1 字面量的方式

> 数组字面量： 所有的元素用方括号括起来，不同的元素之间用逗号分隔。
>
> 例如：["a", 5, "b"]

```
//创建一个长度为 3 的数组，并初始化了3 个元素："abc" "b" "d"
var colors = ["abc", "b", "d"];
//创建一个长度为 0 的空数组数组。里面一个值都没有
var colors = [];
//不要在最后一个元素的后面添加逗号，添加逗号虽然语法没有问题，但是在不同的浏览器可能得到不同的结果
var colors = [5, 6, 7,]; //这样数组的长度可能为 3 也可能为 4。在不同浏览器会得到不同的结果应避免这种创建方式。
```

2.2 构造函数的方式

> 构造函数在创建对象的时候使用。数组的构造函数式 Array()

> 例如： new Array(数组长度);

```
//创建一个长度为 0 的空数组
var colors = new Array();
//创建一个长度为 5 的数组。每个数组的元素的默认值是 undefined。
var colors = new Array(5);
//创建一个长度为 3 的数组，并且3个元素分别是 "blue" "red" "green"
var colors = new Array("blue", "red", "green");
```

> 注意：

1. 使用构造函数创建数组对象的时候，最后一个元素后面不要添加括号，否则报错。 这样是错误的：~~new Array("a",)~~
2. 使用构造函数如果只传入了一个Number值，则这个值必须 >= 0, 否则会报错。
3. 使用构造函数创建数组对象的时候，new 关键字是可以省略的。 例如：Array(5) 这样是可以的。

三、访问和修改数组中的元素

> 利用索引访问数组中的元素。
>
> 如果数组的长度为 5，那么数组的索引为 0,1,2,3,4

```
//创建一个长度为 5 的数据
var arr = [10, 20, 60, 5, 7];
alert(arr[0]);  //获取下标为 0 的元素，   即：10
alert(arr[2]);  //获取下标为 2 的元素，   即：60

//
arr[1] = 100;  //把下标为 1 的元素赋值为100。
```

四、数组的长度

4.1 获取数组的长度

> 每个数组都有一个叫 length的属性，表示数组的长度(即：元素的个数)。

```
var arr = [10, 20, 60, 5, 7];
alert(arr.length);  //弹出：5
```

4.2 修改数组的长度

> 在一般的强类型语言中，数组的长度是固定的，即：数组一旦创建成功，则不能改变数组的长度。
>
> 但是，JavaScript是一种弱类型的动态语言，数组的长度可以在程序运行期间根据需要进行动态的更改
>
> 数组length属性不是只读，而是可以修改的。

1. 通过设置length的值直接修改数组的长度到指定的数值。

```
var arr = ["a", 8, "bc"]; //数组的长度为 3
arr.length = 6; // 修改数组的长度为 6
alert(arr.length);  //数组的长度已经被修改为了 6 ，所以此处输出6.
// 下标为 3， 4， 5 的元素的值为undefined的。
alert(arr[3]);  //弹出：undefined的。

arr.length = 2; // 修改数组的长度为 2 ，则下标为 >= 的元素被自动从数组移除。
```

2. 通过给最后一个元素赋值来动态修改元素的长度。

```
var arr = [4, 6, 8];
// 给下标为 10 的元素赋值为 100.  由于最初长度为 3 ，这个赋值操作完成后，数组的长度会自动增长为11
arr[10] = 100;
alert(arr.length);  // 弹出：11
// 没有赋值的元素默认都为 undefined
alert(arr[5]); //弹出：undefined

alert(arr[20]); //弹出： undefined
alert(arr.length); // 长度仍然为11.  上一行代码仅仅去访问元素，而没有赋值操作，则不会引起数组长度的变化
```

五、数组的遍历

> 一般有3种方法遍历数组:
>
> 1. for循环
> 2. for... in
> 3. for each (ES5 新增)

5.1 使用普通for循环遍历数组

```
var arr = [50, 20, 10, 5, 15, 6];
for(var i = 0; i < arr.length; i++){     //数组长度多长，就遍历多少次。  循环变量作为数组的下标
  console.log(arr[i]);
}
```

5.2 使用 for...in 循环遍历数组

> for-in 语句是一种精准的迭代语句，可以用来枚举对象的属性和数组的元素。
>
> 示例：

```
var arr = [50, 20, 10, 5, 15, 6];
// 每循环一轮，都会把数组的下标赋值给变量index，然后num就拿到了每个元素的下标。
//注意:这里index是元素的下标,不是与萨努
//对数组来说，index从从0开始顺序获取下标
for (var index in arr) {
  console.log(num);  //循环输出：  0 1 2 3 4 5
}
//这里var 关键字也是可以省略的，但是不建议省略。
for(i in arr){
  console.log(arr[i]);
}
```

5.3 使用for ...each遍历数组

> ES5为每个数组新增了一个方法 array.forEach(function) ,使用这个方法，可以自动帮我们遍历数组中的所有元素

```
var arr = [50, 20, 10, 5, 15, 6];
//调用数组的forEach方法，传入一个匿名函数
//匿名函数接受两个参数：    参数1--迭代遍历的那个元素   参数2：迭代遍历的那个元素的下标
//可以在匿名函数内部书需要的代码
arr.forEach( function(element, index) {
  alert(element);
});
```

六、数组常用方法

6.1 转换方法

> toString()转换方法:
>
> - 返回由数组中每个值的字符串形式拼接而成的一个以逗号分隔的字符串

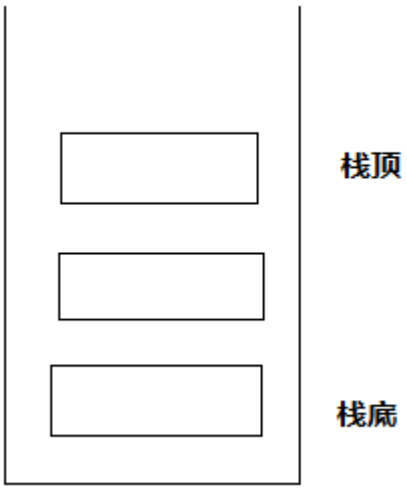
```
<script type="text/javascript">
  var arr = [50, 20, 10, 5, 15, 6];
  alert(arr.toString());  //  50,20,10,5,15,6
  alert(arr);  //  50,20,10,5,15,6   当把一个对象直接给alert，则会调用这个对象的toString方法，然后再输出。
</script>
```

> join() 方法：
>
> - toString() 方法只能使用逗号连接，而 join() 方法可以使用指定的连接符连接

```
<script type="text/javascript">
    var arr = [50, 20, 10, 5, 15, 6];
    alert(arr.join("="));   //  50=20=10=5=15=6
</script>
```

6.2 栈方法

> 栈：一种数据结构。特点：FILO (先进后出)
>
> 向栈中存入元素 称之为 入栈(push)、从栈中移除元素称之为出栈(pop)。先入栈的元素在栈地下，后入栈的元素在栈顶。这两个动作都是对栈顶的元素进行操作。一般栈提供这两个操作足矣。



-

> JavaScript中，支持像操作栈一样去操作数组。

```
<script type="text/javascript">
  var arr = ["张三", "李四", "王五"];
  //向栈中添加元素(最后添加的元素肯定在栈顶)    数组中的元素："张三", "李四", "王五", "志玲"
  var len = arr.push("志玲");   //push方法返回添加成功后的数组的长度
  alert(len);  // 弹出：4
  arr.push("a", "b");  //也可以向在push方法中传入多个参数，这样每个参数都会添加到数组中。   栈顶元素是
"b"

  //pop出栈，一次只能出栈一个元素
  var item = arr.pop();  //把栈顶的元素从栈(数组)中移除。并返回移除的这个元素
  alert(item); // 弹出：b
</script>
```

> 说明：

- 入栈其实就是把新的元素添加到数组的后面
- 出栈其实就是把数组中的最后一个元素从数组中移除

6.2队列方法

> 队列也是一种数据结构。 特点：FIFO(先进先出)
>
> JavaScript中，对数组的操作也提供了模拟队列的方法。
>
> 1. 向队列头部添加元素(unshift)、从队列头部移除元素(shift)
> 2. 向队列尾部添加元素、从队列尾部移除元素

> 注意：对队列尾部的操作没有提供新的方法，使用push和pop可以完成相应的操作。

```
<script type="text/javascript">
  //把arr当做队列对待，那么 队列头部元素就是 "张三"，队尾元素就是 "王五"
  var arr = ["张三", "李四", "王五"];
  var firstItem = arr.shift();  //把队首元素从队列中移除，并返回移除的这个元素
  alert(firstItem); //张三
  alert(arr);  // 李四，王五
  var len = arr.unshift("志玲");  //向队列头部添加元素，并返回添加成功后队列(数组)的长度
  alert("数组长度：" + len);  // 数组长度：3
  alert(arr);  // 志玲，李四，王五
  arr.unshift("a", "b");
  alert(arr);  // a，b，志玲，李四，王五
</script>
```

6.3 数组中元素的倒置

```
<script type="text/javascript">
    var arr = ["张三", "李四", "王五"];
    alert("数组倒置前：" + arr);
    //对数组元素进行倒置。
    arr.reverse();
    alert("数组倒置后：" + arr);
</script>
```

> 注意：

- 倒置操作是对原数组本身做了操作，返回的也是原数组对象，并不是一个新创建的数组。

6.4 查找指定元素在数组中的索引

> indexOf(item): 从前面开始向后查找 item 第一次出现的位置
>
> lastIndexOf(item): 从尾部开始向前查找 item 第一次出现的位置

- 如果找不到元素，则返回 -1

```
<script type="text/javascript">
    var arr = ["张三", "张三", "李四", "王五", "张三", "李四", "王五"];
    alert(arr.indexOf("张三"));  // 0
    alert(arr.lastIndexOf("张三"));  // 4
</script>
```

> indexOf(item, fromBack): 从第二个参数的位置开向后始查找 item 第一次出现的位置
>
> lastIndexOf(item, fromForward): 从第二个参数的位置开始向前查找 item 第一次出现的位置

```
<script type="text/javascript">
    var arr = ["张三", "张三", "李四", "王五", "张三", "李四", "王五"];
    alert(arr.indexOf("张三", 2));  // 4
    alert(arr.lastIndexOf("张三", 3));  // 1
</script>
```

6.4 获取新的数组

1. arr.contact(arrayX,arrayX,.....,arrayX)

该方法用于连接两个或多个数组。至少传入一个参数，参数可以是数组也可以是元素。

注意：该方法是返回的一个新的数组，原数组没有做任何改变

```
<script type="text/javascript">
    var arr1 = ["a", "b", "c"];
    //把参数数组与arr1连接起来，并返回连接后的新数组
    var newArr = arr1.concat(["c", "d"]);
    //新数组的长度是 5
    alert(newArr.length);
    //原数组的长度还是 3  。原数组中的元素没有做任何变化
    alert(arr1.length);

    //把两个元素和一个数组与原数组arr1连接起来，并返回新的数组
    var newArr2 = arr1.concat("e", "f", ["g", "h"]);
    //新数组长度为：7
    alert(newArr2.length);
</script>
```

2. arr.slice(start,end) : 截取数组，并返回截取到的新数组

- start:必须。从原数组中的start位置开始截取(包括下标为start的元素)。 如果是负数表示从尾部开始截取： -1表示最后一个元素
- end: 可选。截取到指定位置(不包括下标为end的元素)。如果没指定，则指的是截取到最后一个元素
- end要大于start，否则截取不到元素

注意：该方法是返回的一个新的数组，原数组没有做任何改变

```
<script type="text/javascript">
    var arr1 = ["a", "b", "c", "d", "e", "f"];
    // 从下标为0的位置开始截取，截取到下标2，但是不包括下标为2的元素. 原数组没有任何的变化
    var newArr = arr1.slice(0, 2);
    alert(newArr);// a, b
    alert(arr1.slice(1, 4)); // b,c,d
    //从下标为2的元素开始截取，一直到最后一个元素
    alert(arr1.slice(2));  //c,d,e,f
    //从倒数第5个元素，截取到倒数第2个
    alert(arr1.slice(-5, -2)); // b c d
</script>
```

3. arr.splice(index,howmany,item1,.....,itemX) 方法向/从数组中添加/删除元素，然后返回被删除的元素组成的数组。

> - 必需。整数，规定添加/删除元素的位置，使用负数可从数组结尾处规定位置。
> - 必需。要删除的元素数量。如果设置为 0，则不会删除元素。 如果添加元素这里应该是0
> - 可选。向数组添加的新项目。
>
> 注意：这个方法会对原数组做出修改。

> - 删除元素

```
<script type="text/javascript">
    var arr1 = ["a", "b", "c", "d", "e", "f"];
    //因为第2个参数不为0，所以表示删除元素：从小标为1的位置开始删除，共删除2个元素。(2个中包括下标为1的元素)
    var deleted = arr1.splice(1, 2);     //返回值为删除的元素组成的数组
    //原数组
    alert(arr1);  // a,d,e,f
    alert(deleted); // b,c
</script>
```

> - 添加元素

```
<script type="text/javascript">
    var arr1 = ["a", "b", "c", "d", "e", "f"];
    //因为第2参数为0，所以表示添加元素：从下标为1的位置插入元素。其余的元素会自动向后移动
    var v = arr1.splice(1, 0, "m", "n");     // 因为是添加元素，所以返回的数组长度为 0
    alert(v.length);  // 0
    alert(arr1);    // a,m,n,b,c,d,e,f
</script>
```

七、数组排序

> JavaScript中，所有的数组对象都提供了一个排序函数。
>
> arr.sort(sortby) 方法用于对数组的元素进行排序。
>
> - sortby 可选。规定排序顺序。必须是函数。

1. 不传入参数的时候，是默认的升序排列。但是做升序排列的时候，是把每个元素转换成string之后，按照编码表中的顺序排序的。

```
<script type="text/javascript">
    var arr1 = ["a", "ab", "fca", "cd", "eb", "f"];
    arr1.sort();    //默认情况按照编码表中的顺序排列
    alert(arr1);    // a, ab, cd, eb, f, fca

    var arr2 = [10, 8, 6, 20, 30, 15];
    arr2.sort();
    console.alert(arr2); // 10,15,20,30,6,8
</script>
```

2. 从上面可以看出来，当数组中的元素是Number的时候，按照编码表排序并不太符合我们的预期，我们更想按照数字的大小排序。这时，我们可以传递一个 "比较函数"。

```
<script type="text/javascript">
    /*
        sort方法进行排序的时候，会调用这个函数，来确定谁大谁小，从而来确定他们的位置。
        排序函数的规则：
        1、比较函数必须具有两个参数  num1, num2
        2、若返回值 > 0, 则认为num1 > num2, 排序的时候num1在num2之后
        3、若返回值 == 0,则认为num1== num2，排序的时候两个数的顺序就保持不变
        4、若返回值 < 0, 则认为num < num2,  排序的时候num1在num2之前。

        总结：
        1、若num1 > num2 返回 正数，  num1 < num2 返回 负数，则是升序
        2、若num1 > num2 返回 负数，  num1 < num2 返回 正数，则是降序
    */
    function sortNumber (num1, num2) {
        //升序
        if(num1 > num2){
            return 1;
        }else if(num1 == num2){
            return 0;
        }else {
            return -1;
        }
    }
    var arr2 = [10, 8, 6, 20, 30, 15];
    arr2.sort(sortNumber);
    console.log(arr2.toString());
</script>
```

- 纯数字的数组，还有一种更简洁的排序函数。

```
//升序函数
function sortAsc(num1, num2){
    return num1 - num2;   //num1 > num2 就返回正数
}
// 降序函数
function sortDesc(num1, num2){
    return num2 - num1; //num1 > num2 就返回负数
}
```

八、数组检测

> 如何检测一个对象是不是一个Array。
>
> 1. 使用instanceof运算符。
> 2. 使用Array.isArray(arr) 方法。

8.1 instanceof运算符

> JavaScript中instanceof运算符会返回一个 Boolean 值，指出对象是否是特定构造函数的一个实例。

```
var arr = [];
alert(arr instanceof Array); //true
```

8.2 Array.isArray(arr) 方法

> Array.isArray(arr) , 如果arr是数组，则返回true，否则返回false

```
var arr = [];
alert(Array.isArray(arr));  //true
alert(Array.isArray("abc"));    // false
```

九、二维数组

> 如果数组中的元素存储的是数组，则就构成了二维数组。

```
//数组中的每个元素都是数组，则就是一个二维数组
var towDimArr = [
                  [4, 5],
                  [7, 8],
                  [50, 9, 10],
                  [5]
                ];
alert(towDimArr.length); //数组的长度为 4

//使用嵌套循环来遍历二维数组。
for (var i = 0; i < towDimArr.length; i++) {
    for (var j = 0; j < towDimArr[i].length; j++) {
      alert(towDimArr[i][j]);
    }
}
```