Day 07---字符串操作和数学函数

一、字符串操作

> 提示：JavaScript中字符串是不可变的。任何涉及到字符串变化的操作都不会修改源字符串，而是创建了一个新的字符串

1.1 创建字符串

> 在JavaScript中，有两种字符串，一种是基本类型的字符串，一种是对象类型(引用类型)的字符串。

1. 获取基本类型的字符串一般有两种办法：

- 使用字符串直接量。例如： "good"

```
var s = "good"; // s为基本类型的字符串
alert(typeof s);  //弹出: string
```

- 使用String()转换函数。 (注意这个地方是不用new关键字的)

```
var s = String(123);  //把一个基本类型的Number转换成基本类型的String. (当然参数可以是任意类型。)
alert(typeof s); // 弹出: string
```

2. 获取引用类型的字符串。通过关键字new来使用String()构造方法。这时，获取到是一个对象，引用类型

```
var s = new String("abc");
alert(typeof s);  //弹出: object
```

> 注意：

- 使用new关键字获取的是引用类型(对象类型)的字符串， 不使用new的是基本类型的字符串

1.2 字符串的length属性

> 字符串的length属性的值表示的是字符串中字符的个数。

```
var s = "abc";
var s = "123你好";
//因为是属性，所以不需要添加括号，只有调用方法或者函数的时候才添加圆括号
alert(s.length);  // 弹出: 3
alert(s.length);  // 弹出: 5
```

1.3 字符串常用方法

1.3.1 字符相关方法

> 1. s.charAt(index)

index 必需。表示字符串中某个位置的数字，即字符在字符串中的下标。

返回值：返回的是指定位置的字符。但是Javascript没有字符类型，其实返回的是长度为 1 的字符串。

说明：可以把字符串想象成一个数组，他的下标也是从0开始，最后一个元素的下标是：s.length - 1

```
var s = "a你好bcd";
alert(s.charAt(0)); // 弹出：  a
alert(s.charAt(2)); // 弹出：  你
//另外，访问字符串中的字符，完全也可以像访问数组一样，s[index]
alert(s[0]);  // 等同与：s.charAt(0)
```

> 2. s.charCodeAt(index)

返回指定位置的字符的 Unicode 编码。这个返回值是 0 - 65535 之间的整数。

```
var s = "a你好bcd";
alert(s.charCodeAt(0)); // 弹出：  97
alert(s.charCodeAt(2)); // 弹出：  22909
```

1.3.2 字符串连接方法

> s.concat(stringX,stringX,...,stringX) 方法用于连接两个或多个字符串。
>
> 注意：
>
> 1. 并不会改变源字符串
> 2. 实际开发中，使用 字符串连接符(+)，更简洁

```
var s = "你好";
alert(s.concat("啊", "志玲")); // 弹出：  你好啊志玲
//如果连接的是长度为 0 的字符串，则返回的是源字符串自己(this).
alert(s.concat("") === s);  // 弹出：  true
```

1.3.3 查找子字符串出现的位置

> 1. s.indexOf(searchvalue,fromindex)方法可返回某个指定的字符串值在字符串中首次出现的位置.
>   - searchvalue必需。规定需检索的字符串值。
> - fromindex 可选的整数参数。规定在字符串中开始检索的位置。它的合法取值是 0 到 s.length - 1。如省略该参数，则将从字符串的首字符开始检索。

```
var v = "abcabdefgh";
//省略第2个参数，表示从字符串开始位置查找
alert(s.indexOf("ab"));  //弹出：  0
//从下标为 1 的位置开始查找
alert(s.indexOf("ab", 1)); //弹出：  3
```

2. s.lastIndexOf(searchvalue,fromindex)　方法可返回一个指定的字符串值最后出现的位置，在一个字符串中的指定位置从后向前搜索。

searchvalue	必需。规定需检索的字符串值。
fromindex	可选的整数参数。规定在字符串中开始检索的位置。它的合法取值是 0 到 s.length - 1。如省略该参数，则将从字符串的最后一个字符处开始检索。

```
var s = "abcabcdab";
alert(s.lastIndexOf("ab")); //弹出： 7
alert(s.lastIndexOf("ab", 5)) //弹出：3
```

1.3.4　截取字符串

1. s.substring(start,stop) 方法用于提取字符串中介于两个指定下标之间的字符。

参数	描述
start	必需。一个非负的整数，规定要提取的子串的第一个字符在 s中的位置。
stop	可选。一个非负的整数。如果省略该参数，那么返回的子串会一直到字符串的结尾。截取到的结果中，不包括stop位置的元素。

```
var s = "生命诚可贵";
// 没有传递参数，则从 指定下标开始截取，一直到字符串结尾。返回截取到的字符串
alert(s.substring(1));  // 弹出：命诚可贵
// 参数1：开始截取位置 参数2：截取结束的位置。　　注意：前面包括后面不包括
alert(s.substring(1, 3)); // 弹出：命诚
```

2. s.substr(start, length) 方法可在字符串中抽取从 start 下标开始的指定数目的字符。

参数	描述
start	必需。要抽取的子串的起始下标。必须是数值。如果是负数，那么该参数声明从字符串的尾部开始算起的位置。也就是说，-1 指字符串中最后一个字符，-2 指倒数第二个字符，以此类推。
length	可选。子串中的字符数。必须是数值。如果省略了该参数，那么返回从 stringObject 的开始位置到结尾的字串。

```
var s = "生命诚可贵";
// 没有传递参数，则从 指定下标开始截取，一直到字符串结尾。返回截取到的字符串
alert(s.substr(1)); // 弹出：命诚可贵。
// 参数1：开始截取位置 参数2：截取的长度。
alert(s.substr(1, 3)); // 弹出：命诚可
```

3. s.slice(start,end) 方法可提取字符串的某个部分，并以新的字符串返回被提取的部分。

注意：这个方法与substring使用方式一样，只是slice允许负值。

参数	描述
start	要抽取的片断的起始下标。如果是负数，则该参数规定的是从字符串的尾部开始算起的位置。也就是说，-1 指字符串的最后一个字符，-2 指倒数第二个字符，以此类推。
end	紧接着要抽取的片段的结尾的下标。若未指定此参数，则要提取的子串包括 start 到原字符串结尾的字符串。如果该参数是负数，那么它规定的是从字符串的尾部开始算起的位置。

```
var s = "生命诚可贵";
//如果是正值，则和substring一样。
alert(s.slice(0, 1))     // 弹出：  生
alert(s.slice(-3, -0))   // 弹出：  诚可
```

1.3.4 大小写转换方法

1. s.toUpperCase 字符串中所有的字符转变成为大写

```
var s = "abcAbc";
alert(s.toUpperCase());  // ABCABC
```

2. s.toLowerCase 字符串中的所有的字符转变成小写

```
var s = "ABcAbc";
alert(s.toLowerCase());
```

1.3.5 去除字符串首尾空白字符

s.trim() : 只是去除字符串的首尾的所有空白字符. 字符串内部的空白字符不做任何处理

```
var s = " \n \t ABc   Abc   \t \n \t";
alert(s.trim());//ABc   Abc
```

1.3.6 字符串替换方法

1. s.replace(regexp/substr,replacement) 方法用于在字符串中用一些字符替换另一些字符，或替换一个与正则表达式匹配的子串。

参数	描述
regexp/substr	必需。规定子字符串或要替换的模式的 RegExp 对象。请注意，如果该值是一个字符串，则将它作为要检索的直接量文本模式，而不是首先被转换为 RegExp 对象。
replacement	必需。一个字符串值。规定了替换文本或生成替换文本的函数。

说明：

1. 正则表达式，仅了解。后面再细讲。 /abc/gi
2. 如果是普通的字符串，则只替换第一个满足的要求。

```
var s = "abcaba";
// "ab"是普通的字符串，则只用js去替换第一个 ab
var newStr = s.replace("ab", "js"); // jscaba
alert(newStr); //弹出：jscaba
// 正则表达式格式：    /字符/属性
alert(s.replace(/ab/gi, js));  //替换所有的ab，且忽略大小写
```

> 2. s.match(匹配值)：在字符串内检索指定的值

- 匹配的参数只有一个，要么正则表达式，要么是字符串项
- 返回值：存放匹配结果的数组

```
var s = "abcaba";
var arr = s.match("ab"); //因为是 普通字符串，  所以只匹配第一个
alert(arr); // ["ab"]
alert(s.match(/ab/gi)); // ["ab", "ab"]

var s1 = "今天是是2016年11月5日";
//匹配所有的数组
var nums = s1.match(/\d+/gi);
console.log(nums.toString()); //输出 : 2016,11,5
```

> 3. s.search(匹配的参数)

- 匹配的参数只有一个，要么是正则表达式，要么是字符串
- 返回值：第一个匹配项的索引，否则返回-1，始终从字符串的头部开始查找，忽略全局g

```
var s = "abcaba";
var arr = s.search("ab"); //不管是普通字符串，还是正则表达式，只匹配第一个。返回值是索引
alert(arr); // 0
alert(s.search(/ab/gi)); // 0  忽略全局

var s1 = "今天是是2016年11月5日";
//匹配所有的数组
var nums = s1.search(/\d+/gi);
console.log(nums.toString()); //输出 : 4
```

1.3.7 字符串比较

> 1. == 比较两个字符串的内容是否相等。只有内容相等就返回true。

```
var s = "今天是是2016年11月5日";
var s1 = new String("今天是是2016年11月5日");
//虽然是一个基本类型的字符串，一个引用类型的对象，但是他们的内容是相等的。所以返回  true
alert(s == s1);
```

> 2. === 恒等 只有类型和内容都相等的时候才返回true

```
var s1 = "今天是是2016年11月5日";
var s2 = String("今天是是2016年11月5日");  //使用转换函数,得到的是基本类型的string
var s3 = new String("今天是是2016年11月5日");
var s4 = new String("今天是是2016年11月5日");

//内容和类型都相等，算恒等
alert(s1 === s2);   // true
//虽然内容相等，但是类型不等。所以不算恒等
alert(s1 === s3);   //false
//s3 和 s4 类型和内容都相等，但是他们是对象，所以必须是同一个对象才算恒等。 面向对象阶段细讲
alert(s3 === s4);   //false
```

> 3. s.localeCompare(other):
>
> - 如果字符串在字母表中应该排在字符串参数之前，则返回一个负值；
> - 如果字符串的等于字符串参数，返回0；
> - 如果字符串在字母表中应该排在字符串参数之后，则返回一个正数；

```
var s = "b";
var compare = s.localeCompare("aaa"); //返回正数 其实是 1
alert(compare);

alert(s.localeCompare("c"));   //返回负数    其实是 -1
```

1.3.8 字符串切割方法

> s.split(separator,howmany) 方法用于把一个字符串分割成字符串数组。

参数	描述
separator	必需。字符串或正则表达式，从该参数指定的地方分割 s。
howmany	可选。该参数可指定返回的数组的最大长度。如果设置了该参数，返回的子串不会多于这个参数指定的数组。如果没有设置该参数，整个字符串都会被分割，不考虑它的长度。(一般情况不设定这个参数)

```
var s = "How do you do";
var arr = s.split(" "); // 使用 " " 空格来切割字符串
alert(arr.length);  // 4
alert(arr); // How,do,you,do
```

二、Math对象

> Math主要做一些数学上的常用运算：比如平方、绝对值、开方、三角函数等。

2.1 常用属性

> 1. Math.PI : π的值
> 2. Math.E: 自然对数的底数：

```
console.log(Math.PI);  // 3.141592653589793
console.log(Math.E);   // 2.718281828459045
```

2.2 常用方法

> 1. Math.abs(x) : 返回x的绝对值

```
console.log(Math.abs(5));  // 5
console.log(Math.abs(-5));  // 5
```

> 2. Math.max(任意个数值) ： 返回传入的数值中的最大值

```
console.log(Math.max(40, 6, 80));  // 80
```

> 3. Math.min(任意个数值) ： 返回传入的数值中的最小值

```
console.log(Math.min(40, 6, 80));  // 6
```

> 4. Math.ceil(number) ： 返回大于等于number的最小整数(向上取整)

```
console.log(Math.ceil(13.1));  // 14
console.log(Math.ceil(-13.1));  // -13
```

> 5. Math.floor(number) ： 返回小于等于number的最大整数(向下取整)

```
console.log(Math.floor(13.1));  // 13
console.log(Math.floor(-13.1)); // -14
```

> 6. Math.round(number): 四舍五入

```
console.log(Math.round(13.4));  // 13
console.log(Math.round(13.5));  // 14
console.log(Math.round(-13.5)); // -13
console.log(Math.round(-13.6)); // -14
```

> 7. Math.pow(x, y) : 返回 x^y

```
console.log(Math.pow(2, 3)); // 8
```

> 8. Math.random() : 返回 0-1之间的随机小数。包括0，但是不包括1

```
console.log(Math.random());
```

> 9. Math.sqrt(x) : 返回x的平方根

```
console.log(Math.sqrt(4));  // 2
```

> 10. Math.sin(x) 正弦, Math.cos(x) 余弦, Math.tan(x) 正切

注意：三角函数的参数都是弧度。

```
console.log(Math.sin(Math.PI / 4));  // 45度的正弦
console.log(Math.cos(Math.PI / 4));  // 45度的余弦
console.log(Math.tan(Math.PI / 4));  // 90度的正切
```