**字符串**

——在底层是以字符数组的形式保存的

注意: ECMAScript 中的字符串是不可变的; 也就是说，字符串一旦创建，它们的值就不能改变.

例如: var str = “亲,包邮哦”;

str[0] = “唉”; //不会改变

如果要改变某个变量保存的字符串，首先要销毁原来的字符串，然后再用另一个包含新值的字符串填充该变量.

var str = “Hello”;

str = str+” world!”;



var str ="Hello";

≈["H","e","l","l","o"]

**属性：length**

——用来获取字符串的长度

语法：var str = "hello";

document.write(str.length);

**字符**

**1.charAt(index)**

——可以返回字符串中指定位置的字符

——根据索引获取指定位置的字符

——如果index不在0-str.length(不包含str.length)之间，返回空字符串。

**语法：**

var result = str.charAt(6);

var result = str[6];

**2.charCodeAt(index)**

——返回在指定的位置的字符的Unicode编码

——这个返回值是 0 - 65535 之间的整数。

**语法：**

var result = str.charCodeAt(6);

**3.String.fromCharCode( )**

——可以根据Unicode字符编码去获取字符

——可传入多个参数

**语法：**

result = String.fromCharCode(0x2692);//0x表示16进制

console.log(result);

**合并**

**4.concat()**

——可以用来连接两个或多个字符串

——作用和+—样

**语法：**

result = str.concat("你好","再见");

等于result = "你好"+"再见";

**查找**

**5.indexOf( )**

——该方法可以检索一个字符串中是否含有指定内容

——如果字符串中含有该内容，则会返回其第一次出现的索引

如果没有找到指定的内容，则返回-1

——可以指定一个第二个参数，指定开始查找的位置

不写第二个参数则从0开始

**语法：**

str ="hello h";

result = str.indexof("h",2);

所有字符串indexOf("")索引都是0

**6.lastIndexOf( )**

——该方法的用法和indexOf()一样

不同的是indexOf是从前往后找

而lastIndexOf()是从后往前找

——也可以用第二个参数指定开始查找的位置

**语法：**

str ="hello h";

result = str.lastIndexof("h",1);

**7.search()**

——正则匹配 (返回第一次出现的索引)

——为了支持正则表达式而诞生的一种方法 支持字符也支持正则表达式

**语法：**

str.search("c")

**正则语法:**

var str = “Abcdabcd”;

var index = str.search(/abc/gi);

注: g表示进行全局匹配，i表示匹配的时候忽略大小写

**8.match()**

——找到了返回数组，找不到返回null

——也既支持字符，也支持正则表达式

**语法：**

str.match("c")

**截取**

**9.slice()**

——可以从字符串中截取指定的内容

——不会影响原字符串，而是将截取到内容返回

——参数：

1.开始位置的索引（包括开始位置)

2.结束位置的索引（不包括结束位置)

——如果省略第二个参数，则会截取到后边所有的

——也可以传递一个负数作为参数，负数的话将会从后边计算

**语法：**

result = str.slice(1,4);

result = str.slice(1,-1);

**10.substring()**

——可以用来截取一个字符串，可以slice( )类似

——参数:

1.开始截取位置的索引(包括开始位置)

2.结束位置的索引(不包括结束位置)

——如果省略第二个参数，则会截取到后边所有的

——不同的是这个方法不能接受负值作为参数，

如果传递了一个负值，则默认使用0

而且它还自动调整参数的位置，如果第二个参数小于第一个，则自动交换

**语法：**

result = str.substring(1,0);等于(0,1)

**11.substr( )**

——用来截取字符串-参数;

——参数:

1.截取开始位置的索引

2.截取的长度

**语法：**

str = "abcdefg";

result = str.substr( 3,2);

ECMAscript 没有对该方法进行标准化，因此反对使用它。

**切割**

**12.split()**

——可以将一个字符串拆分为一个数组

——参数:

——需要一个字符串作为参数，将会根据该字符串去拆分数组

如果传递一个空串作为参数，则会将每个字符都拆分为数组中的一个元素

**语法：**

str = "abc , bcd ,efg,hij";

result = str.split( ", ");//根据逗号拆分 把逗号当做分隔符

**大小写转换**

**13.toUpperCase()**

——将一个字符串转换为大写并返回

**语法：**

result = str.toUpperCase();

**14.toLowerCase()**

——将一个字符串转换为小写并返回

**语法：**

result = str.toLowerCase();

**替换**

**15.replace()**

替换字符串

**语法：**

var newStr = str.replace(“are”, “old are”);

这种写法替换只能执行一次，不能够进行全局匹配，如果需要全局匹配，则应使用正则表达式：

str.replace(/are/gi, "old are")

注意：这个函数不修改原字符串 会返回修改后的字符串；

**去掉前后空白**

**16.trim()**

——用于去掉前后空白，保留中间的空白

**语法：**

str.trim();

**保留小数**

**17.toFixed(n)**

——保留小数点后n位

**语法：**

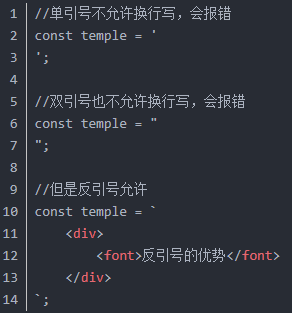
var str = 12.1231231;

str.toFixed(2);

**模板字符串**

ES6允许我们使用反引号（数字1旁边的那个）来定义字符串，然后在字符串中使用${变量名}来引用变量，引用时可以对变量进行一些运算。

这一方式也让我们能够很好地使用html元素，而不用不断地使用字符串来拼接。

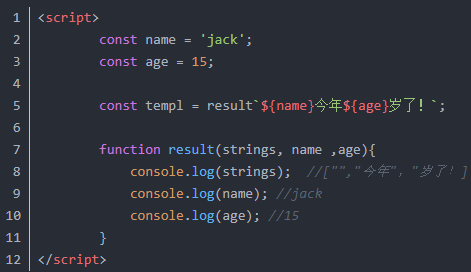


另外，模板字符串是允许嵌套的。

**标签模板字符串**

模板字符串里主要分为两种元素，一种就是普通的字符串，还有一种就是引用的变量。

可以在模板字符串前添加一个无分隔的标签。然后使用该标签名建立函数时，就能获取到模板字符串里的元素，并加以处理。



**要注意的几个地方**

因为js中函数无法通过类型来确定，所以函数会第一个接收字符串数组，然后各变量按顺序传递。

字符串会被引用的变量分隔，然后拆成一个字符串数组。而且如果变量在模板字符串的第一个或者最后一个位置，获取的字符串数组就会在前或后多一个空字符串。

如果参数直接使用变量相同的名字，就可以直接获取，不管有没有在参数中明确定义。（但是字符串数组的参数不可省）。

如果变量太多，可以使用剩余参数的形式来定义，此时会将所有的变量存进一个数组。

function result(strings, ...values){ //剩余参数用三个点来表示

console.log(strings); //["","今年"，"岁了！]

console.log(values); //["jack",15 ]

}

以上三个点的表示方法用于接收剩余参数（剩余参数只能放在最后），另一方面，也可以用该表示方法来对数据进行展开，比如

const arr1 = [1, 2, 3, 4];

const arr2 = [5, 6, 7, 8];

const arr = [...arr1, ...arr2];

//效果相当于

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

同理也可以对json进行展开。

另外还有很多字符串处理方法，具体可以参考js参考文档<https://www.w3school.com.cn/jsref/jsref_obj_string.asp>