day06 字符串

字符串概述

字符串也是一种数据结构叫做串（共同的内容串在一块）。字符串是一种基础值类型，基础值类型不可变。字符串在算法解决问题的时候会使用的比较频繁，常用于解决一些查找相关的问题（马拉车算法 解决回文字符串的）。

字符串的声明

赋值声明

```
//可以使用对应的单引号也可以使用双引号
var str = '字符串'
```

通过new关键词来构建

```
var str = new String('传入任意类型')
```

赋值声明和new关键词声明的区别

- new关键词声明的字符串 用typeof验证得到的是object 是赋值声明得到是string
- 使用new关键词声明的字符串俩个是不相等的

```
var str = 'hello'
var str1 = 'hello'
console.log(str == str1) //true
console.log(str === str1) //true
var str2 = new String('hello')
var str3 = new String('hello')
console.log(str2 == str3)//false
console.log(str2 === str3)//false
console.log(str == str2)//true
console.log(str === str2)//false
```

es6新增字符串模板 ``

```
var str = 'hello'
//在字符串模板内容 ${}表示引用变量 传入的是变量名 支持标准js语法
var str1 = `jack ${str}`
//var str1 = 'jack'+str
console.log(str1)
```

字符串的属性以及访问

- length属性 表示字符串长度（只读属性）

```
var str = 'hello'
console.log(str.length)//5
str.length = 10 //无意义的操作 字符串不可变
console.log(str.length)//5
```

- 字符串访问其他的某个内容可以通过下标来访问

```
//串这个数据结构是有序的 可以通过下标来访问对应的里面的某个字符
console.log(str[0])//h
str[0] = 'A' //字符串不可变 没有意义
console.log(str)//hello
console.log(str[0])//h
```

字符串相关的方法（字符串不可变）

字符串相关方法是以返回值来操作 （本身是不变的）

indexOf 根据传入的字符串返回第一次出现的下标（找不到返回-1）从前往后

```
console.log('aaaabbbb'.indexOf('a'))//0
console.log('aaaabbbb'.indexOf('ab'))//3
```

lastIndexOf 根据传入的字符串返回第一次出现的下标 （找不到返回-1） （从后往前）

```
console.log('aaaabbbb'.lastIndexOf('a')) //3
console.log('aaaabbbbab'.lastIndexOf('ab')) //8
```

search 方法类似于indexOf 支持正则表达式

```
// search 返回对应的下标 找不到返回-1 支持正则表达式
console.log('aaabbb'.search('ab'))//2
console.log('aaabbb'.search('ab',3))//2 search只有一个参数  没有开始位置
```

底层实现lastIndexOf及indexOf

```
//简易实现indexOf
var str = 'abcbcaabcd'
function myIndexOf(value,index){
    if(value == undefined){
        throw new Error('传参个数错误')
    }
    //如果index没有传入设置默认值
    if(index == undefined){
        index = 0
    }
    //遍历查找
    //得到对应的长度
    //如果当前的开始位置+value的长度大于本身的字符串长度 那么返回-1
    if(index+value.length > str.length){
        return -1
    }else{
        //遍历对应的字符串
        for(var i=index;i<str.length - value.length;i++){
            //根据value长度来拼接
```

```
            var v = str[i]
            for(var j=1;j<value.length;j++){
                v+=str[i+j]
            }
            //根据对应的value来比较
            if(value == v){
                return i
            }
        }
    }
    return -1
}
console.log(myIndexOf('abc',2))
//lastIndexOf的实现
function myLastIndexOf(value,index){
    if(value == undefined){
        throw new Error('传参个数错误')
    }
    //如果index没有传入设置默认值
    if(index == undefined){
        index = str.length
    }
    //遍历查找
    //得到对应的长度
    for(var i=index;i>=0;i--){
        //字符串拼接
        var v = str[i]
        for(var j=1;j<value.length;j++){
            v+=str[i+j]
        }
        if(v == value){
            return i
        }
    }
    return -1
}
```

charAt 根据下标返回字符

```
console.log('abc'.charAt(0))//a
```

charCodeAt 根据下标返回字符串的ascii码

```
console.log('abc'.charCodeAt(0))//97
```

静态方法 String.fromCharCode

```
//静态方法
// 将asscii码变成字符串 String
var str = String.fromCharCode(97) //返回一个字符串
console.log(str)
```

截取相关的方法 (如果只传递一个参数那么默认都是截取到最后)

substring 传入开始下标及结束下标（包含开始的下标不包含结束的）

```
var str = 'abcdef'.substring(3,5)//de
```

substr 传入开始下标和个数

```
var str = 'abcdef'.substr(2,2)//从下标2开始截取俩个 cd
```

slice 传入开始下标及结束下标 （可以一个参数都不传递默认截取所有）

```
var str = 'abcdef'.slice(2,4)//cd
```

连接的方法

concat

```
var str = 'hello'
//返回一个新的字符串
var concatStr = str.concat('world')
console.log(concatStr) //helloworld
```

支持正则的四个方法

- search

```
//search 查找（根据传入的字符串返回第一次出现位置的下标）
var str = '1_abcd'
//\w数字字母下划线
var index = str.search(/\w/)
console.log(index)//0
```

- match

```
//match 匹配 返回一个数组（包含匹配的内容）执行一次
var str = '_abc?789'
var strArr = str.match(/\w/)
console.log(strArr)
```

- replace

```
//replace 替换 根据对应的匹配内容进行替换 执行一次
var str = 'abcabc'
// var str1 = str.replace('a','f')//传入字符串形式 将a替换为f
var str1 = str.replace(/a/,'f')//传入正则形式 将a替换为f
console.log(str1)
//高阶函数写法的replace 支持传入函数作为参数
//传入一个函数作为参数的形式 里面的参数是匹配的内容 里面的return是替换的内容
var replaceStr = str.replace(/a/,function(value){
    console.log(value)//a
    return 'f'
})
console.log(replaceStr)
```

- split

```
//split 分割的 将对应的字符串分割为一个数组返回
//数组的join相反 将数据拼接成一个字符串返回 将字符串拆分为数组
var str = 'a,b,c'
//如果不传参 他就将这个字符串直接填入数组返回
// var arr = str.split(',')
var arr = str.split(/,/)
console.log(arr)
```

练习

将"abcabcabcabc" 中的bc全部替换为hello

```
var str = "abcabcabcabc"
//先找bc 再进行替换没有了退出
while(str.search('bc')!=-1){
    str = str.replace('bc','hello')
}
console.log(str)
```

统计一个字符串在另一个字符串内的出现次数

```
//统计一个字符串在另外一个字符串中出现的次数
function getCount(from,str){
    //abcabcabc  bc
    var arr = from.split(str)
    return arr.length-1
}
var count = getCount('bacfddesdasda','dd')
console.log(count)
```

其他相关的函数

去除首尾空格 trim

```
var str = ' a b c '
console.log(str.trim())
```

转大小写

- toUpperCase 转大写
- toLowerCase 转小写

```
console.log(str.toUpperCase())//ABCBC
console.log(str.toLowerCase())//abcbc
```

- toLocalUpperCase
- toLocalLowerCase

```
//toLocalUpperCase
//toLocalLowerCase
//根据本地格式进行转换
console.log('a'.toLocaleUpperCase())
console.log('A'.toLocaleLowerCase())
//toString 转为字符串 他是所有对象都存在的方法（万物皆对象）
```

统计一个字符串中大写字母的个数

```
//遍历比较
function fn(str) {
    var count = 0
    //先遍历
    for (var i = 0; i < str.length; i++) {
        if (str.charAt(i) >= 'A' && str.charAt(i) <= 'Z') {
            count++
        }
    }
    return count
}
console.log(fn('AbcASna'))
//利用正则
console.log( 'AbcASna'.match(/[A-Z]/g).length)
```

html元素相关的方法

- **sub 返回一个sub标签**

```
var str = 'hello world'
console.log(str.sub())//<sub>hello world</sub>
document.write(str.sub())
```

- **fontColor 返回font标签 color属性**

```
//fontColor
console.log(str.fontcolor('red'))
document.write(str.fontcolor('red'))
```

- **fontSize 返回font标签 size属性**

```
//fontSize
console.log(str.fontsize(30))
document.write(str.fontsize(30))
```

练习

HOW ARE YOU ? I AM DJ. DJ DJ DJ... 将dj变成**并且变成红色 将首字母大写 其他全部小写

```
var str = 'HOW ARE YOU ? I AM DJ. DJ DJ DJ... '
str = str.trim().toLowerCase() //先全部小写
var arr = str.split(' ') //拆分成一个个的内容
//遍历数组进行操作
for (var i = 0; i < arr.length; i++) {
    //出现dj 全部转为**
    arr[i] = arr[i].replace('dj', '**'.fontcolor('red'))
    //arr[i]就是里面一个个的字符串
    //将首字母大写 其他不变
    arr[i] = arr[i][0].toUpperCase() + arr[i].slice(1)
}
var str = arr.join(' ') //将数据连接变成字符串
document.write(str)
```

总结

- 字符串不可变 字符串的方法以返回一个新的字符串来进行操作
- 字符串indexOf方法用于获取下标的 charAt方法用于或者字符串
- 字符串截取方法substring slice 开始下标和结束下标作为参数 substr 以个数来作为截取
- 字符串支持正则的方法 search match replace split
- 字符串进行拼接的方法concat
- 字符串去除首尾空格的方法trim， toUpperCase 转大写 toLowerCase 转小写

Math类

Math是数学类，他里面包含关于数学操作的方法及属性（里面的内容设计为静态 直接Math去点）

属性

PI 数学Π

E 科学计数法e

方法

- max 最大值 传入多个值比较出最大值并返回
- min 最小值 传入多个值比较出最小值返回
- pow 取幂次方
- sqrt 开平方
- ceil 向上取整
- floor 向下取整
- round 四舍五入
- random 取0-1之间的随机数 包含0 不包含1
- abs 取绝对值

```
//相关方法
console.log( Math.max(1,2,3,4,5,6))
console.log( Math.min(1,2,3,4,5,6))
//返回2的三次方
console.log( Math.pow(2,3))
//开平方
console.log( Math.sqrt(4))
//取整
//向上取整
console.log( Math.ceil(2.4321))//3
//四舍五入
```

```
console.log( Math.round(2.4321))//2
//向下取整
console.log( Math.floor(2.4321))//2
//随机数 0-1 包含0不包含1
console.log(Math.random())
//绝对值 正值
console.log(Math.abs(-10))
//三角函数 一度= π/180
console.log(Math.sin(45*Math.PI/180))
console.log(Math.cos(45*Math.PI/180))
console.log(Math.tan(45*Math.PI/180))
```

取区间的随机数

```
//min max
function randomNumber(min,max){
    return Math.random()*(max-min)+min
}
```