目录

[输入输出 2](#_Toc499647556)

[java 2](#_Toc499647557)

[.next() 2](#_Toc499647558)

[nextLine() 3](#_Toc499647559)

[.hasNext() 3](#_Toc499647560)

[.nextFloat() .nextInt().nextDouble( 3](#_Toc499647561)

[.hasNextFloat() .hasNextInt().hasNextDouble().hasNextLine() 3](#_Toc499647562)

[js 3](#_Toc499647563)

[流程 3](#_Toc499647564)

[String 3](#_Toc499647565)

[共有（5+2） 3](#_Toc499647566)

[charAt 3](#_Toc499647567)

[concat 3](#_Toc499647568)

[indexOf 4](#_Toc499647569)

[split 4](#_Toc499647570)

[substr和substring slice 4](#_Toc499647571)

[toLowerCase toUpperCase trim 4](#_Toc499647572)

[valueOf 4](#_Toc499647573)

[js 5](#_Toc499647574)

[slice 5](#_Toc499647575)

[数组 5](#_Toc499647576)

[java 5](#_Toc499647577)

[js 5](#_Toc499647578)

[函数处理 5](#_Toc499647579)

[查找 6](#_Toc499647580)

[增删查改 6](#_Toc499647581)

[变成字符串 7](#_Toc499647582)

[其他 7](#_Toc499647583)

[string正则 7](#_Toc499647584)

[match和matches 7](#_Toc499647585)

[replace 7](#_Toc499647586)

[js 8](#_Toc499647587)

[search 8](#_Toc499647588)

[正则 9](#_Toc499647589)

[exec 9](#_Toc499647590)

[test 9](#_Toc499647591)

[compile 9](#_Toc499647592)

[数据结构set map等 9](#_Toc499647593)

[set 9](#_Toc499647594)

# 输入输出

## java

import java.util.\*;

public class inout{//没有void ,小写class

public static void main(String[] args){//static在前面

Scanner sc=new Scanner(System.in);

int[] a=null;//一定要初始化

while(sc.hasNext()){

int n=sc.nextInt();

a=new int[n];

for(int i=0;i<n;i++){

a[i]=sc.nextInt();

}

System.out.println(a[0]);//不能再while外面

}

}

}

方法

### .next()

不能得到带空格的字符串（字符串之前的空格会去掉，之后的空格作为结束符）

### nextLine()

### .hasNext()

### .nextFloat() .nextInt().nextDouble(

### .hasNextFloat() .hasNextInt().hasNextDouble().hasNextLine()

## js

### 流程

//引入

var readline=require('readline')//记得赋给变量

//创建接口实例

var rl=readline.createInterface({

    input:process.stdin,

    output:process.stdout,

})

监听line事件

rl.on('line',function(input){

    console.log(input)

})

# String

## 共有（5+2）

### charAt

返回在指定位置的字符。

### concat

s = s.concat("www.runoob.com");

js可以连接多个

var n = str1.concat(str2,str3);

### indexOf

返回某个指定的字符串值在字符串中首次出现的位置。

java可以从指定位置处搜索

[int indexOf(int ch, int fromIndex)](http://www.runoob.com/java/java-string-indexof.html)（可以是char,可以是String）

lastIndexOf 最后一次出现的索引

### split

把字符串分割为字符串数组。

string.split(separator正则,limit分数)

js份数可以省略，默认最大数量

### substr和substring slice

js substr(起始位置，数量（省略代表到结尾）)

java substring(起始位置，结束位置(省略到结尾))

js里也有substring和java一样，不能是负数, js里面有slice,可以是负数。

### toLowerCase toUpperCase trim

### valueOf

js 表示返回原始值，一般不需要用

var str="Hello world!";  
document.write(str.valueOf());

java 表示将其他类型值固定成字符串。

double d = 1100.00;

boolean b = true;

long l = 1234567890;

char[] arr = {'r', 'u', 'n', 'o', 'o', 'b' };

System.out.println("返回值 : " + String.valueOf(d) );

System.out.println("返回值 : " + String.valueOf(b) );

System.out.println("返回值 : " + String.valueOf(l) );

System.out.println("返回值 : " + String.valueOf(arr) )

遍历

java foreach

for (double element: myList) { System.out.println(element); }

## js

### slice

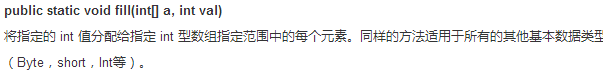
提取字符串片段，并在新的字符串中返回被提取的部分。（用substr或者substring最好，和java保持一致）

# 数组

Array和Arrays属性方法

## java

java.util.Arrays

* 给数组赋值：通过 fill 方法。（用于初始化）
* 
* 对数组排序：通过 sort 方法,按升序。
* 比较数组：通过 equals 方法比较数组中元素值是否相等。
* 查找数组元素：通过 binarySearch 方法能对排序好的数组进行二分查找法操作。

## js

### 函数处理

every filter find findInex forEach map reduce reduceRight some sort

**记得return！！！**

#### 传参

正常传参属性（**function**（当前位置值，位置，整个数组），**thisValue(**传递给回调函数，用作this的值，省略则this为undefined)）包括八个

其他

**reduce** reduceRight(function（total,currentValue,currentIndex,arr）,initalValue)(多了一个属性total)

|  |  |
| --- | --- |
| *total* | 必需。*初始值*, 或者计算结束后的返回值。 |
| *currentValue* | 必需。当前元素 |
| *currentIndex* | 可选。当前元素的索引 |
| *arr* | 可选。当前元素所属的数组对象。 |

**sort**(function) ,通常有a,b 也可以是a b c

#### 分类

返回boolean every some

无返回值 forEach

返回一个值 find findIndex（索引位置）（都是返回找到的第一个）reduce recudeRight

返回部分数组filter

返回整个数组map sort

### 查找

indexOf lastIndexOf

find finIndex

### 增删查改

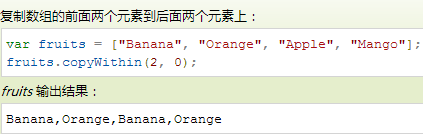
concat copyWithin pop push reverse shift slice **splice** unshift

***.*splice**(index,howmany,item1,.....,itemX) (起始位置,数量（省略表示删到最后，0表示不删），添加的新元素（不是另一个数组）)

**copyWithin**(target, start, end)(兼容性不好)

从数组的指定位置拷贝元素到数组的另一个指定位置中。返回这个数组（即原数组变了）

|  |  |
| --- | --- |
| *target* | 必需。复制到指定目标索引位置。 |
| *start* | 必需。元素复制的起始位置。 |
| *end* | 可选。停止复制的索引位置 (默认为 *array*.length) |



### 变成字符串

join toString

### 其他

valueOf

fill 填充初始值 fill(value, start（可选）, end（可选）)

# string正则

java和js语法有些区别

在String的方法中，java放在字符串里，js直接是正则，所以在String方法中，java没有标志符

Str.replaceAll("(.\*)taobao(.\*)","runoob" )

str.replace(/blue/gi, "red");

## match和matches

js match有返回一个数组(不管有没有g)，只有一个参数。没匹配到返回null。

java matches返回Boolean值，等效Pattern.matches(regex, str)



## replace

成功则返回替换的字符串，失败则返回原始字符串。

java replace单个字符，replaceAll正则字符串，replaceFirst第一个

### js

*str*.replace(*regexp*|*substr*, *newSubstr*|*function*)

字符串

var newstr = str.replace(re, '$2, $1');

函数

function replacer(match, p1, p2, p3, offset, string) {

// p1 is nondigits, p2 digits, and p3 non-alphanumerics

return [p1, p2, p3].join(' - ');

}// 和正则对象一致。

可以用对象去匹配

function formatDate(t,str){

var obj = {

yyyy:t.getFullYear(),

yy:(""+ t.getFullYear()).slice(-2),

M:t.getMonth()+1,

MM:("0"+ (t.getMonth()+1)).slice(-2),

d:t.getDate(),

dd:("0" + t.getDate()).slice(-2),

H:t.getHours(),

HH:("0" + t.getHours()).slice(-2),

h:t.getHours() % 12,

hh:("0"+t.getHours() % 12).slice(-2),

m:t.getMinutes(),

mm:("0" + t.getMinutes()).slice(-2),

s:t.getSeconds(),

ss:("0" + t.getSeconds()).slice(-2),

w:['日', '一', '二', '三', '四', '五', '六'][t.getDay()]

};

return str.replace(/([a-z]+)/ig,**function($1){return obj[$1]}**);

}

## search

查找与正则表达式相匹配的值。

匹配到返回起始匹配位置，没匹配到返回-1

java没有search

## split

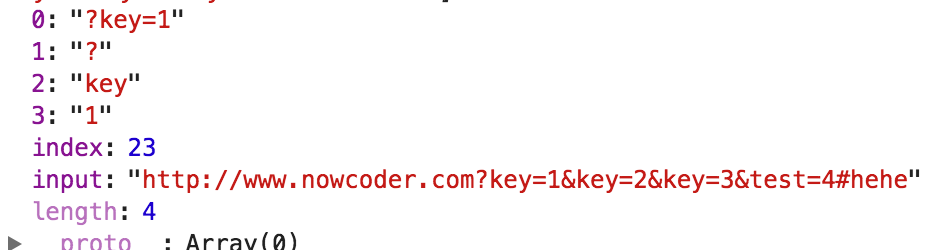
js有，将字符串切割成字符数组。

# 正则

## exec

js

返回一个类数组。包括整个匹配到的和子串。



## test

返回true或false

## compile

编译正则。

# 数据结构set map等

## set

add

detele 和remove

has 和contains

clear

Array.from 和 toArray()