**9·13**

**JavaScript基本数据类型**：

Number 数字类型 是-(253 -1) 到 253 -1范围内的双精度实数

String 字符串类型 所有字符、字符串

Boolean 逻辑类型

null 空引用 指向空对象

undefined 未定义类型变量 新声明变量未初始化

Object 对象类型（非基本数据类型） 包括类对象、函数对象、数组对象等

BigInt 大整数类型 仅存在于谷歌浏览器中 可以安全使用任意大的整数

使用BigInt()构造函数创建或在数字后附小写字母n：BigInt(12)或12n

* 不能与任何Number类型变量互相操作，需要转为同一变量，但可相互比较
* 与Number类型变量宽松相等： 12==12n true || 12===12n false
* 不能使用Math对象中的方法
* BigInt兼容+、-、\*、%、\*\*及除>>>外所有的位运算符，不支持单目运算符+，支持/的整数运算，支持比较运算符

Symbol 类型 表示唯一的值，用来定义对象的唯一属性名，防止改写或覆盖设置的属性值。

唯一性：即使是用同一个变量生成的值也不相等。

隐藏性：for···in，object.keys() 等不能访问Symbol类型变量。

// Symbol 值作为类属性名时，该属性是公有属性。

**学习**：

ES6—let

let不允许重复声明变量

let允许在块状变量域中声明变量，有效作用域为块状变量域

let声明变量不允许使用前声明变量

// 比var更加优异

ES6-const

const用于声明常量

对于基本类型常量，不可修改值

对于对象常量，可修改其内容的值，不可修改其指向

特别的，常量一般以大写命名

ES6-模板字符串

允许使用反引号 ` ` 【在键盘左上方】来代替普通字符串中的单引号或双引号来声明字符串

允许声明时出现换行符

let s = `Onclick

Onmove`;

允许变量拼接, 可以用${}格式在字符串中引用变量

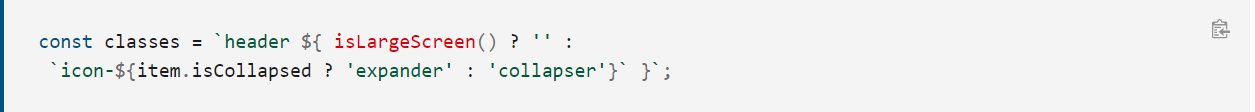
let f = `peach`;

let s = `${f} is my favorite fruit.`;

// s 为 “peach is my favorite fruit.”

使用起来更加优雅美观





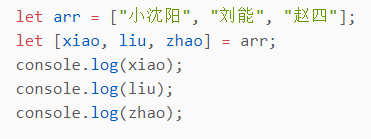
特别的，在模板字符串中使用反引号，需要加转义符\

ES6-变量解构赋值

按照一定模式从数组和对象中提取值，对变量进行赋值

数组使用[ ]解构

对象使用{ }解构

ES6-对象简化写法

ES6 允许在对象的大括号里面，直接写入变量名和函数名，作为对象的属性和方法

属性变量可以直接用变量名声名

对象方法可以省去 function 声名

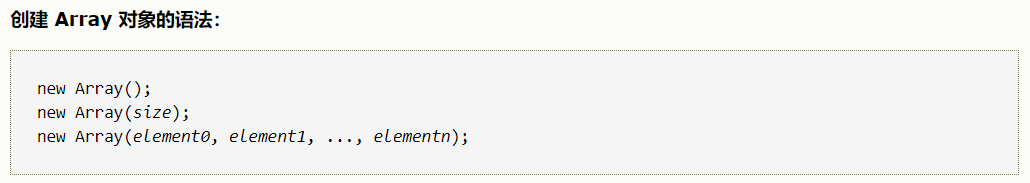


**其他**：

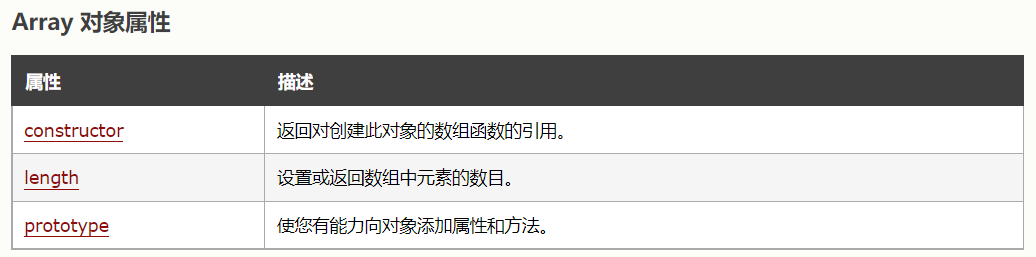
**9·14**

数组方法：

构造方法——



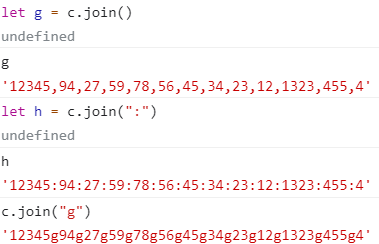
对象属性——



对象方法——



数组join()方法



数组join()方法可不传参数，默认使用逗号；也可传入一个字符串，方法将使用这个字符串作为数组各元素间的分隔符

数组concat()方法不会改变当前的数组内容，返回的是一个副本

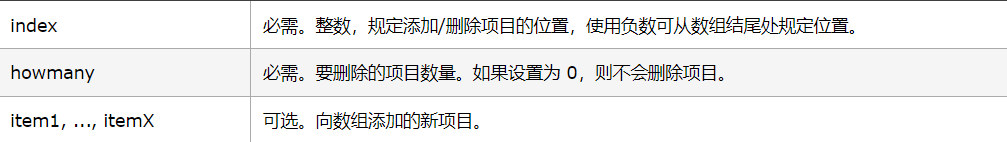
slice() 方法可从已有的数组中返回选定的元素，不会改变数组原有内容

正数从数组开头开始，负数从数组末尾开始

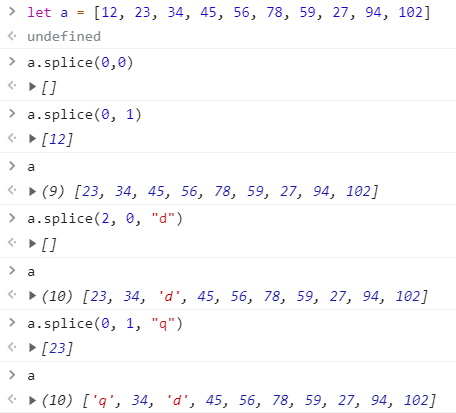
俩参数，一个为开始下标，另外一个是结束下标，可省略

splice() 方法向/从数组中添加或删除项目，然后返回被删除的项目，会改变数组原有内容

参数表



测试

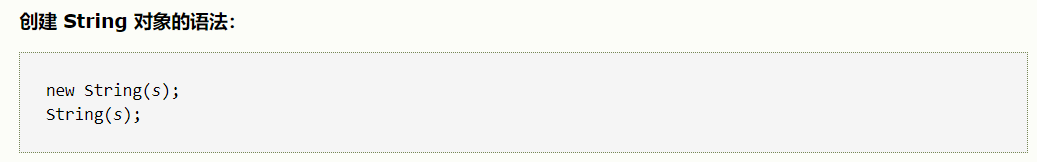


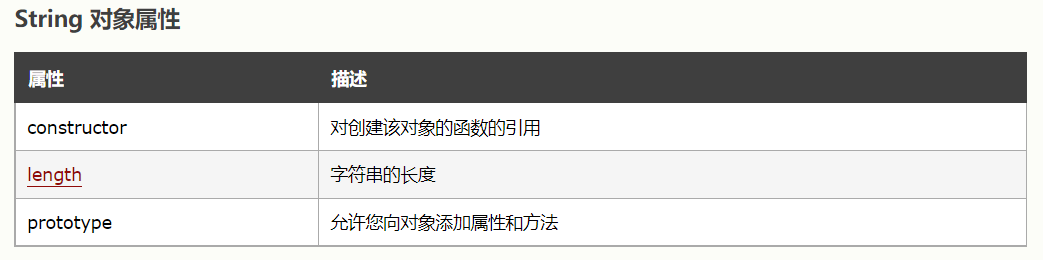
学习：

其他：

**9·15**

字符串方法：

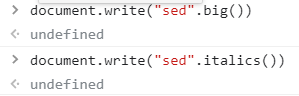
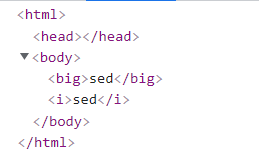








anchor()、big()、blink()、bold()、fixed()、fontcolor()、fontsize()、italics()、link()、small()、strike()、sub()、sup()等方法，用于向document中写入格式化的文本。实际效果相当于向文档写入以对应字符串为内容的HTML标签或包含对应CSS属性的font标签内容。

其他一些方法与对字符串的修改查找有关

学习：

其他：

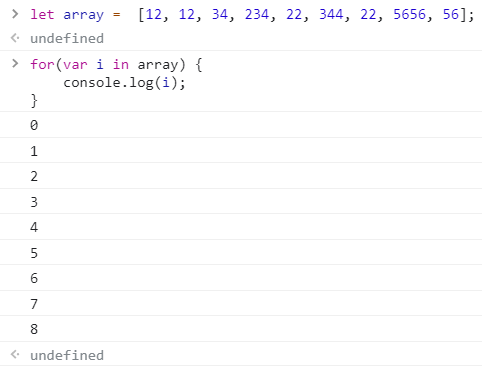
**9·16**

**增强for循环**：

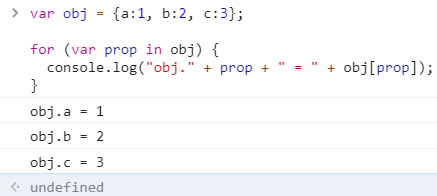
for-in

for in 循环只会遍历我们自定义的属性，原型上默认的属性不会遍历出来，但会将原型中新增的属性和方法遍历出来

for...in语句以任意顺序遍历一个对象的除Symbol以外的可枚举属性



特别的，for ... in是为遍历对象属性而构建的，不建议与数组一起使用



foreach

forEach() 方法用于调用数组的每个元素，并将元素传递给传入的回调函数参数





for...of

适用于遍历任何可迭代的数据结构

数组

const array = [‘mini’,’ ko’];

for (const t of array) {console.log(array);}

// 输出依次为 “mini” “ko”

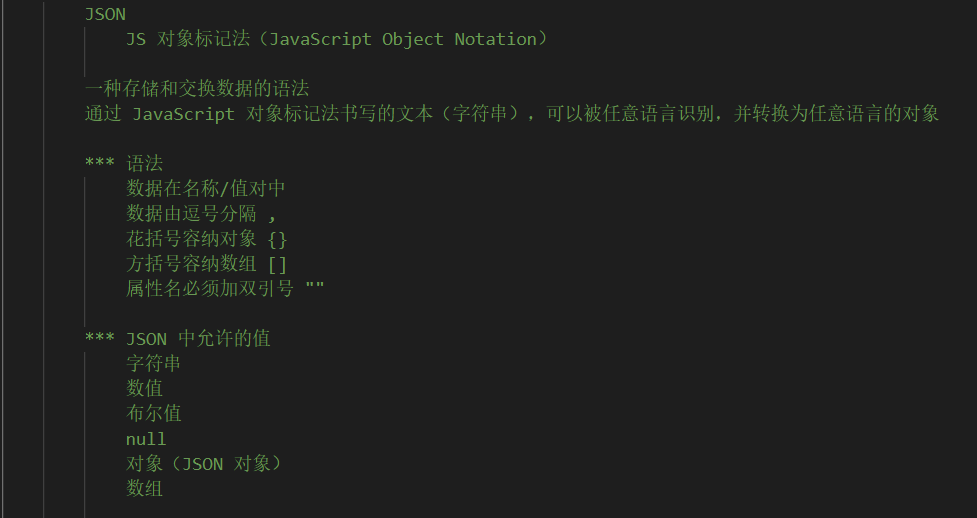
这非常类似Java中的for-冒号循环遍历数组

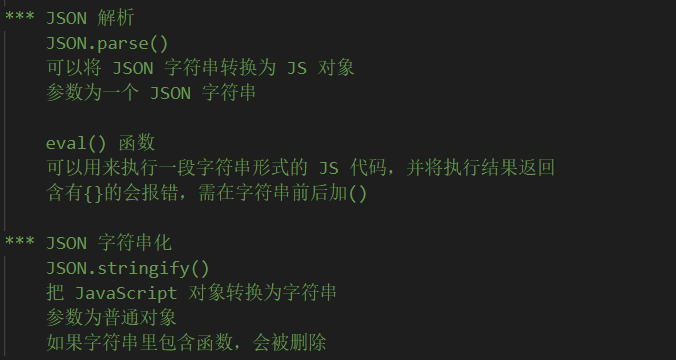
其他还可遍历的包括：映射Maps、集合Set、TypedArray对象，字符串、参数对象、DOM对象

特别的，普通对象并不被允许使用

**学习**：

JSON：







**其他**：

4种结束JavaScript循环方式：break、continue、return、throw

arguments：

是一个对应于传递给函数的参数的类数组对象

可以在所有非箭头函数中使用

参数索引自0开始

arguments的内容可被修改

不是Array类型，仅具length和索引属性，但可转为Array类型变量









使用typeof返回‘object’

严格模式下，剩余参数、默认参数和解构赋值参数的存在不会改变 arguments对象的行为

非严格模式中的函数没有包含剩余参数、默认参数和解构赋值，那么arguments对象中的值会跟踪参数的值——修改arguments对象中的值也会修改参数的值，反之亦然

非严格模式中的函数有包含剩余参数、默认参数和解构赋值，那么arguments对象中的值不会跟踪参数的值，反映了调用时提供的参数——修改arguments对象中的值不会修改参数的值，反之亦然

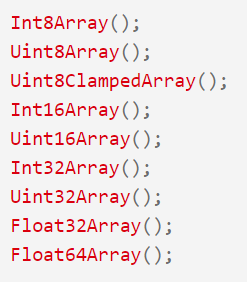


\*\*\*使用 arguments.length来确定传递给函数参数的个数

\*\*\*使用Function.length属性确定函数签名中（输入）参数的数量

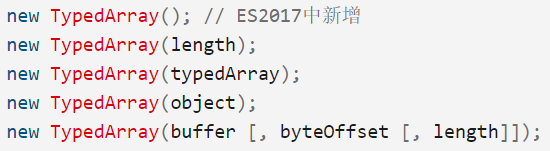
类型化数组（TypedArray）对象：

描述了一个底层的二进制数据缓冲区（binary data buffer）的一个类数组视图（view）



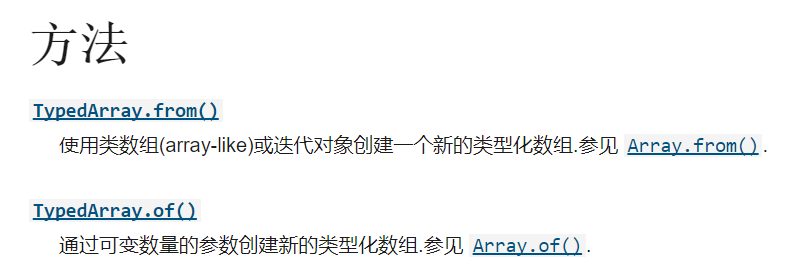
TypedArray对象是其中之一

构造方法包含：









Object.hasOwnProperty(prop) 用来判断某个对象是否含有指定的属性的，返回值为Boolean ，该方法会忽略掉那些从原型链上继承到的属性

**9·17**

JS类型转换：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| To From | Number | String | Boolean | Null | Undefined |
| Number |  | Number()  纯数值转数字  有非数字字符转NaN  空串或全空格转0  parseInt()  parseFloat()  提取字符串开头整数/实数 | Number()  1或0  parseInt()  parseFloat()  NaN | Number()  0  parseInt()  parseFloat()  NaN | Number()  NaN  parseInt()  parseFloat()  NaN |
| String | toString()  String() |  | toString()  String()  ‘true’或’false’ | String()  "null" | String()  "undefined" |
| Boolean | 0转false，其他true | 空串false，其他true |  | false | false |
| Null |  |  |  |  |  |
| Undefined |  |  |  |  |  |

parseInt()&parseFloat()函数——先转字符串再转数字

parseInt(str, hex) 可用来提取出有效非10进制的数

学习：

其他：