# 数组遍历方法及区别（遍历数组）

for in 方法，遍历的是数组的下标，而不是数组的实际元素

for of 方法可以直接访问数组中元素的值

foreach方法是可以同时访问数组的下标和元素的值

map方法和filter方法一个是遍历数组，一个是过滤数组

forEach()方法不会返回执行结果，而是undefined。他会修改原来的数组。

而map()方法会得到一个新的数组并返回。

foreach里面不可以使用break

foreach遍历时会跳过空元素，而for in 和for of不会跳过空元素

严格模式下使用foreach遍历数组，如果是普通函数，他的this指向时undefine，如果是箭头函数，他的this指向是一个空的对象

# 数组多维变一维？

数组降维：

c、toString()或者jion（）方法  
     通过arr.toString()转成字符串===》

再通过splic变成数组 arr.toString().splic(&apos;,&apos;)变成数组(项为string),

然后便利数组每一项由string变成一维数组

a.es10中的Array.flat()方法可以降维

flat() 方法会按照一个可指定的深度递归遍历数组，并将所有元素与遍历到的子数组中的元素合并为一个新数组返回。

flat参数是向下降几维，无参就是向下降一维

b.原生js： 创建一个函数 语句体是：便利数据中的每一项，并且进行判断，判断为数组（instanceof, isArray），则回调该函数，直到便利结束

# 数组去重

--1.如果不考虑兼容性问题可以使用es6中的set方法进行去重

function unique (arr) {

return Array.from(new Set(arr))

}

///////////

[...new Set(arr)] !!!!!!!!!!!!

--2.在es5中可以使用双重for循环，然后通过【splice】方法进行去重（双层循环，外层循环元素，内层循环时比较值。值相同时，则删去这个值。）

function unique(arr){

for(var i=0; i<arr.length; i++){

for(var j=i+1; j<arr.length; j++){

if(arr[i]==arr[j]){ //第一个等同于第二个，splice方法删除第二个

arr.splice(j,1);

j--;

}

}

}

return arr;

}

--3.使用indexof方法，创建一个空数组，然后遍历原数组时进行判断，使用indexof将数组元素进行搜索，如果返回值为-1代表没有重复的，使用push方法将元素添加到新数组中

function unique(arr) {

var array = [];

for (var i = 0; i < arr.length; i++) {

if (array .indexOf(arr[i]) === -1) {

array .push(arr[i])

}

}

return array;

}

--4.使用sort方法，将数组进行自动排序，然后遍历数组，通过比较两个相邻值是否一致来决定是否将元素添加到一个新数组中

function unique(arr) {

arr = arr.sort()

var arrry= [arr[0]];

for (var i = 1; i < arr.length; i++) {

if (arr[i] !== arr[i-1]) {

arrry.push(arr[i]);

}

}

return arrry;

}

# 遍历对象，对象遍历

for...in 遍历输出的是对象自身的属性以及原型链上可枚举的属性(不含Symbol属性),原型链上的属性最后输出说明先遍历的是自身的可枚举属性,后遍历原型链上的

Object.keys() 遍历对象返回的是一个包含对象自身可枚举属性的数组(不含Symbol属性).

Objcet.getOwnPropertyNames() 输出对象自身的可枚举和不可枚举属性的数组,不输出原型链上的属性

Reflect.ownKeys() 返回对象自身的所有属性,不管属性名是Symbol或字符串,也不管是否可枚举.

\_.keys 用underscore插件的遍历方法只可以遍历出对象自身的可枚举属性

# 强制数据类型转换

toString()方法：转换成字符串。调用被转换数据类型的toString()方法,该方法不会影响到原变量，它会将转换的结果返回，但是注意：null和undefined这两个值没有toString

String()函数：String方法的参数如果是对象，返回一个类型字符串；如果是数组，返回该数组的字符串形式。

使用Number()函数：  
（1）如果是纯数字的字符串，则直接将其转换为数字  
（2）如果字符串中有非数字的内容，则转换为NaN  
（3）如果字符串是一个空串或者是一个全是空格的字符串，则转换为0  
（4）布尔值转数字:true转成1,false转成0  
（5）undefined转数字:转成NaN  
（6）null转数字：转成0  
（7）Number() 接受数值作为参数，此时它既能识别负的十六进制，也能识别0开头的八进制，返回值永远是十进制值  
（8）Number方法的参数是对象时，将返回NaN，除非是包含单个数值的数组。  
其他的数据类型转换为Boolean：只有空字符串("")、null、undefined、+0、-0 和 NaN 转为布尔型是 false，其他的都是 true

**parseInt()和parseFloat()两个转换函数。前者把值转换成整数，后者把值转换成浮点数。只有对String类型调用这些方法，**这两个函数才能正确运行；对其他类型返回的都是NaN(Not a Number)。

# for in、for each、for of的区别

For in一般都用它来遍历非数组对象，并且原型链上的属性也会被遍历。

要用hasOwnProperty()方法(可以检测一个属性是存在于实例中还是原型中)去过滤。

For of遍历对象的值value。他能响应break、continue、return语句。

For each遍历数组过程不能终止，必须每个值都遍历一遍后才能停下来

数组.for each(function(value,index,array){})

它与jQuery中的$.each的区别：他遍历数组和类数组，它与jQuery参数相反。

$.each(数组,function(index,value,array){})

## [forEach、map、filter、reduce的区别](https://www.cnblogs.com/kaiqinzhang/p/11496151.html)

1.相同点：

都会循环遍历数组中的每一项；

map()、forEach()和filter()方法里每次执行匿名函数都支持3个参数，参数分别是：当前元素、当前元素的索引、当前元素所属的数组；

匿名函数中的this都是指向window；

只能遍历数组。

2.不同点：

map()速度比forEach()快；

map()和filter()会返回一个新数组，不对原数组产生影响；forEach()不会产生新数组，返回undefined；reduce()函数是把数组缩减为一个值(比如求和、求积)；

reduce()有4个参数，第一个参数为初始值。

# 数组指定位置插入元素

Splice（） 第一个参数：规定添加、删除项目的位置，使用负数可以那个数组结尾开始找位置

第二个参数： 要删除项目的数量，为0则不删除

第三个参数（可选参数）： 想数组中添加的新元素

# 判断一个变量是数组还是对象

第一，使用typeof加length属性：数组有length属性，object没有，而typeof数组与对象都返回object，所以我们可以这么判断

第二，使用instanceof

使用instanceof可以判断一个变量是不是数组，会得到一个布尔值false或true。

第三，使用object。prototype。tostring。call === 【object array/object】

***如何判断一个***变量***是不是数组***

判断一个变量是否是数组有几种不同的方法

1.使用Array.isArray 判断，如果返回 true, 说明是数组.

第一种方法的原理如下：

Array.isArray = function(value) {

return Object.prototype.toString.call(value) === '[object Array]';

}

简单理解就是运用对象原型的toString方法将Array转换为字符串，之后进行比对，下面我们来详细分析下各个部分的组成。

Object.prototype.toString代表这个toString方法本来是对象原型上面的，之后call方法将toString方法中的this转换为参数value，这样传进的参数value（实际上对于此方法应该是个数组[]）就可以通过原型的方法来转换为字符串

2.使用 instanceof Array 判断，如果返回true, 说明是数组

3.使用 Object.prototype.toString.call判断，如果值是 [object Array], 说明是数组

4.通过 constructor 来判断，如果是数组，那么arr.constructor === Array.(不准确，因为我们可以指定obj.constructor = Array)

判断一个变量是否是数组的案例

function fn() {

console.log(Array.isArray(arguments)); //false; 因为arguments是类数组，但不是数组

console.log(Array.isArray([1,2,3,4])); //true

console.log(arguments instanceof Array); //fasle

console.log([1,2,3,4] instanceof Array); //true

console.log(Object.prototype.toString.call(arguments)); //[object Arguments]

console.log(Object.prototype.toString.call([1,2,3,4])); //[object Array]

console.log(arguments.constructor === Array); //false

arguments.constructor = Array;

console.log(arguments.constructor === Array); //true

console.log(Array.isArray(arguments)); //false

}

fn(1,2,3,4);

# 数组中every和reduce的区别

every方法会依次检测数组中每一个元素 是否符合给定函数的条件,返回布尔值,不会对空

数组处理，不改变原数组。所有元素都满足才返回true。

reduce方法会对数组中的每一个元素依次进行回调函数的方法， 返回最终的值。。

# 给你一个对象怎么判断是一个空对象或者非空对象？object.keys返回的是什么?

可以放到数组里，然后.length为0，就是空。

var obj = {};

(1)for...in... 遍历属性，为真则为“非空数组”;否则为“空数组”

function test1(obj){

for(var i in obj){

return true;

}

return false;

}

(2)通过 JSON 自带的 stringify() 方法来判断

function test2(obj){

if(JSON.stringify(obj)==='{}') return false;

return true;

}

(3)ES6 新增的方法 Object.keys()

function test3(obj){

return Object.keys(obj).length !== 0;

}

返回false为空数组

# Array 对象的数组与字符串相互转换方法

toString() 将数组转换成一个字符串

toLocalString() 把数组转换成本地约定的字符串

join() 将数组元素连接起来以构建一个字符串

split（“&”） 将字符串按照&分割成数组

Array对象—数组对象

创建对象的方式

--var 对象名=new Array();

--var 对象名=new Array(大小);

--var 对象名=new Array(值1,值2,值3,值4);

--var 对象名=[值1,值2,值3,值4]; //建议使用

Array对象的属性

--length属性

--prototype属性(原型)

Array对象的方法

--concat( ) 连接数组

--join( ) 将数组元素按指定分隔符连接起来，返回一个字符串

--pop( ) 删除并返回数组的最后一个元素 （出栈）

--push( ) 给数组添加元素 （入栈）

--reverse( ) 颠倒数组中元素的顺序

--shift( ) 将元素移出数组

--unshift( ) 在数组头部插入一个元素

--slice( ) 返回数组的一部分

--sort( ) 对数组元素进行排序

--splice( ) 插入、删除或替换数组的元素

--toString( ) 将数组转换成一个字符串

# **数组**

## **变异方法**

 (mutation method)会改变被这些方法调用的原始数组

### 1.push( )

定义：push() 方法可向数组的末尾添加一个或多个元素，并返回新数组的长度。

语法：arrayObject.push(newelement1,newelement2,....newelementX)

|  |  |
| --- | --- |
| **参数** | **描述** |
| newelement1 | 必需。要添加到数组的第一个元素。 |
| newelement2 | 可选。要添加到数组的第二个元素。 |
| newelementX | 可选。可添加多个元素。 |

返回值：把指定的值添加到数组后的新长度。

例子：

var arr = new Array(3);

arr[0] = "George";

arr[1] = "John";

arr[2] = "Thomas";

console.log(arr); //(3) ["George", "John", "Thomas"]

console.log(arr.push("James"));//4

console.log(arr);//(4) ["George", "John", "Thomas", "James"]

### 2，pop( )

定义：pop() 方法用于删除并返回数组的最后一个元素。

语法：arrayObject.pop( )

返回值：arrayObject 的最后一个元素。

说明：pop() 方法将删除 arrayObject 的最后一个元素，把数组长度减 1，并且返回它删除的元素的值。如果数组已经为空，则 pop() 不改变数组，并返回 undefined 。

例子：

var arr = new Array(3);

arr[0] = "George";

arr[1] = "John";

arr[2] = "Thomas";

console.log(arr); //(3) ["George", "John", "Thomas"]

console.log(arr.pop());//Thomas

console.log(arr);//(2) ["George", "John"]

### 3,shift( )

定义：shift()方法用于把数组的第一个元素从其中删除，并返回第一个元素的值。

语法：arrayObject.shift( )

返回值：arrayObject 原来的第一个元素的值。

说明：如果数组是空的，那么 shift() 方法将不进行任何操作，返回 undefined .

例子：

var arr = new Array(3);

arr[0] = "George";

arr[1] = "John";

arr[2] = "Thomas";

console.log(arr); //(3) ["George", "John", "Thomas"]

console.log(arr.shift());//George

console.log(arr);//(2) ["John", "Thomas"]

### 4,unshift( )

定义：unshift() 方法可向数组的开头添加一个或更多元素，并返回新的长度。

语法：arrayObject.unshift(newelement1,newelement2,....newelementX)

|  |  |
| --- | --- |
| **参数** | **描述** |
| newelement1 | 必需。要添加到数组的第一个元素。 |
| newelement2 | 可选。要添加到数组的第二个元素。 |
| newelementX | 可选。可添加多个元素。 |

返回值：把指定的值添加到数组后的新长度。

说明：unshift() 方法无法在 Internet Explorer 中正确地工作！

例子：

var arr = new Array(3);

arr[0] = "George";

arr[1] = "John";

arr[2] = "Thomas";

console.log(arr); //(3) ["George", "John", "Thomas"]

console.log(arr.unshift("James"));//4

console.log(arr);//(4) ["James","George", "John", "Thomas"]

### 5,splice( )

定义：splice() 方法向/从数组中添加/删除元素，然后返回被删除元素组成的数组。

语法：arrayObject.splice(index,howmany,newelement1,newelement2,....newelementX)

|  |  |
| --- | --- |
| **参数** | **描述** |
| index | 必需。整数，规定添加/删除项目的位置，使用负数可从数组结尾处规定位置。 |
| howmany | 必需。要删除的项目数量。如果设置为 0，则不会删除项目。 |
| newelement1...newelementX | 可选。向数组添加的新元素。 |

返回值：如果从 arrayObject 中删除了元素，则返回的是含有被删除的元素的数组。

例子：

var arr = new Array(3);

arr[0] = "George";

arr[1] = "John";

arr[2] = "Thomas";

console.log(arr); //(3) ["George", "John", "Thomas"]

console.log(arr.splice(1,0,"James"));//[]

console.log(arr);//(4) ["George", "James", "John", "Thomas"]

console.log(arr.splice(1,2,"Martin"));//["James", "John"]

从下标为1的元素开始删除，删除两个元素，添加一个新元素，

console.log(arr);//(3) ["George", "Martin", "Thomas"]

### 6,sort( )

定义：sort() 方法用于对数组的元素进行排序。

语法：arrayObject.sort(sortby)

|  |  |
| --- | --- |
| 参数 | 描述 |
| sortby | 可选。规定排序顺序。必须是函数。 |

返回值：排序后的数组。

说明：如果调用该方法时没有使用参数，将按字母顺序对数组中的元素进行排序，如果想按照其他标准进行排序，就需要提供比较函数，该函数要比较两个值，然后返回一个用于说明这两个值的相对顺序的数字。比较函数应该具有两个参数 a 和 b，其返回值如下：若 a 小于 b，在排序后的数组中 a 应该出现在 b 之前，则返回一个小于 0 的值。若 a 等于 b，则返回 0。若 a 大于 b，则返回一个大于 0 的值。

例子：var arr = new Array(6)

arr[0] = 10;

arr[1] = 5;

arr[2] = 59;

arr[3] = 45;

arr[4] = 34;

arr[5] = 3000;

arr[6]=3;

console.log(arr);//(7) [10, 5, 59, 45, 34, 3000, 3]

console.log(arr.sort());//(7) [10, 3, 3000, 34, 45, 5, 59]

**请注意，上面的代码没有按照数值的大小对数字进行排序，要实现这一点，就必须使用一个排序函数sortby:**

function sortby(a,b){

if(a<b){

return a-b;

}else if(a=b){

return 0;

}else{

return a-b;

}

}

arr.sort(sortby);

console.log(arr);//(7) [3, 5, 10, 34, 45, 59, 3000]

### 7,reverse( )

定义：reverse() 方法用于颠倒数组中元素的顺序。

语法：arrayObject.reverse( )

说明：该方法会改变原来的数组。

例子：

var arr = new Array(3);

arr[0] = "George";

arr[1] = "John";

arr[2] = "Thomas";

arr[3]="刘能";

arr[4]="赵四";

arr[5]="tom";

console.log(arr);//(6) ["George", "John", "Thomas", "刘能", "赵四", "tom"]

console.log(arr.reverse());//(6) ["tom", "赵四", "刘能", "Thomas", "John", "George"]

console.log(arr);//(6) ["tom", "赵四", "刘能", "Thomas", "John", "George"]

### 8,forEach( )

forEach() 方法用于调用数组的每个元素，并将元素传递给回调函数。forEach(): 没有返回值，本质上等同于 for 循环，对每一项执行 function 函数。是变异性方法。

注意: forEach() 对于空数组是不会执行回调函数的。

语法：array.forEach(function(currentValue, index, arr), thisValue)；

currentValue（必需，数组中的元素。）

**注：forEach() 本身是不支持的 continue 与 break 语句的，我们可以通过 some 、 every、try catch 来实现。**

使用 return 语句实现 continue 关键字的效果：

例：

var arr = [1, 2, 3, 4, 5];

arr.forEach(function (item) {

if (item === 3) { //元素等于3的时候跳过

return;

}

console.log(item); //1,2,4,5

});

使用try catch实现break

例：try {

var array = ["first","second","third","fourth"]; // 执行到第3次，结束循环

array.forEach(function(item,index){

if (item == "third") {

throw new Error("EndIterative");

}

alert(item);// first,sencond

});

} catch(e) {

if(e.message!="EndIterative") throw e;

};

使用erery()实现break

var arr = [1,2,3,4,5];

var num = 3;

arr.every(function(v){

if(v == num) {

return false;

}else{

console.log(v);

return true;

}

});

## **非变异方法**

(non-mutating method)，不会改变原始数组，但总是返回一个新数组。

### 1,filter( )

定义：filter() 方法创建一个新的数组，新数组中的元素是通过检查指定数组中符合条件的所有元素。

语法：arrayObject.filter(function(currentValue,index,arr),thisValue)

|  |  |
| --- | --- |
| **参数** | **描述** |
| function(currentValue, index,arr) | 必须。函数，数组中的每个元素都会执行这个函数 函数参数:   |  |  | | --- | --- | | **参数** | **描述** | | currentValue | 必须。当前元素的值 | | index | 可选。当前元素的索引值 | | arr | 可选。当前元素属于的数组对象 | |
| thisValue | 可选。对象作为该执行回调时使用，传递给函数，用作 "this" 的值。 如果省略了 thisValue ，"this" 的值为 "undefined" |

例子：

var arr = [7,32,33,16,40,45,4656];

function check(num) {

return num >= 18;

}

function myFunction() {

console.log(arr.filter(check));

}

console.log(arr);//(7) [7, 32, 33, 16, 40, 45, 4656]

myFunction();//(5) [32, 33, 40, 45, 4656]符合条件的元素被筛选出来

console.log(arr);//(7) [7, 32, 33, 16, 40, 45, 4656]原数组没有发生改变

### 2.concat( )

定义：concat() 方法用于连接两个或多个数组。该方法不会改变现有的数组，而仅仅会返回被连接数组的一个副本。

语法：arrayObject.concat(arrayX,arrayX,..........arrayX)

|  |  |
| --- | --- |
| arrayX | 必需。该参数可以是具体的值，也可以是数组对象。可以是任意多个。 |

返回值：返回一个新的数组。该数组是通过把所有 arrayX 参数添加到 arrayObject 中生成的。

说明：如果要进行 concat() 操作的参数是数组，那么添加的是数组中的元素，而不是数组。

例子：

var arr = new Array(3)

arr[0] = "George"

arr[1] = "John"

arr[2] = "Thomas"

var arr2 = new Array(3)

arr2[0] = "James"

arr2[1] = "Adrew"

arr2[2] = "Martin"

var arr3 = new Array(2)

arr3[0] = "William"

arr3[1] = "Franklin"

console.log(arr.concat(arr2,arr3));//(8) ["George", "John", "Thomas", "James", "Adrew","Martin", "William", "Franklin"]

console.log(arr);//(3) ["George", "John", "Thomas"]

console.log(arr2);//(3) ["James", "Adrew", "Martin"]

console.log(arr3);//(2) ["William", "Franklin"]

### 3,slice( )

定义：slice() 方法可从已有的数组中返回选定的元素。

语法：arrayObject.slice(start,end)

|  |  |
| --- | --- |
| start | 必需。规定从何处开始选取。如果是负数，那么它规定从数组尾部开始算起的位置。也就是说，-1 指最后一个元素，-2 指倒数第二个元素，以此类推。 |
| end | 可选。规定从何处结束选取。该参数是数组片断结束处的数组下标。如果没有指定该参数，那么切分的数组包含从 start 到数组结束的所有元素。如果这个参数是负数，那么它规定的是从数组尾部开始算起的元素。 |

返回值：返回一个新的数组，包含下标从 start 到 end **（不包括下标为end的元素）**的 arrayObject 中的元素。

说明：请注意，该方法并不会修改数组，而是返回一个新数组。如果 没有设置end ，那么 slice() 方法会选取从 start 到数组结尾的所有元素。

例子：

var arr = new Array(6)

arr[0] = "George";

arr[1] = "John";

arr[2] = "Thomas";

arr[3] = "James";

arr[4] = "Adrew";

arr[5] = "Martin";

console.log(arr);//(6) ["George", "John", "Thomas", "James", "Adrew", "Martin"]

console.log(arr.slice(2,4));//(2) ["Thomas", "James"]

console.log(arr);//(6) ["George", "John", "Thomas", "James", "Adrew", "Martin"]

### 4,map()

map() 方法返回一个新数组，数组中的元素为原始数组元素调用函数处理后的值。

map() 方法按照原始数组元素顺序依次处理元素。

注意： map() 不会对空数组进行检测。

注意： map() 不会改变原始数组。

例：var numbers = [4, 9, 16, 25];

function myFunction() {

x = document.getElementById("demo")

x.innerHTML = numbers.map(Math.sqrt);

}

### 5,some()

some() 方法用于检测数组中的元素是否满足指定条件（函数提供）。

some() 方法会依次执行数组的每个元素：

如果有一个元素满足条件，则表达式返回true , 剩余的元素不会再执行检测。

如果没有满足条件的元素，则返回false。

注意： some() 不会对空数组进行检测。

注意： some() 不会改变原始数组。

语法：array.some(function(currentValue,index,arr),thisValue)

返回值：布尔值。如果数组中有元素满足条件返回 true，否则返回 false。

例：

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <script>

**var** ages = [4, 12, 16, 20];

**function** myFunction() {

    document.getElementById("demo").innerHTML = ages.some(v **=>** {return v>document.getElementById("ageToCheck").value});

}

    </script>

</head>

<body>

    <p>最小年龄: <input type="number" id="ageToCheck" value="18"></p>

    <button onclick="myFunction()">点我</button>

    <p>判断结果: <span id="demo"></span></p>

</body>

</html>

### 6,every()

every() 方法用于检测数组所有元素是否都符合指定条件（通过函数提供）。

every() 方法使用指定函数检测数组中的所有元素：

**如果数组中检测到有一个元素不满足，则整个表达式返回 false ，且剩余的元素不会再进行检测。**

如果所有元素都满足条件，则返回 true。

注意： every() 不会对空数组进行检测。

注意： every() 不会改变原始数组。

例：

**var** arr = [1,2,3,4,5];

**var** num = 4;

        arr.every(v**=>**{

             if(v == num) {

                   return false;

             }else{

                   console.log(v);

                   return true;

        }

# 二、字符串

## 1、str.charAt(index);

返回子字符串，index为字符串下标，index取值范围[0,str.length-1]

## 2、str.charCodeAt(index);

返回子字符串的unicode编码，index取值范围同上

## 3、String.fromCharCode(num1,num2,...,numN);

根据unicode编码返回字符串

## 4、str.indexOf(searchString,startIndex);

返回子字符串第一次出现的位置，从startIndex开始查找，找不到时返回-1

## 5、str.lastIndexOf(searchString,startIndex);

从由往左找子字符串，找不到时返回-1

## 6、str.substring截取字符串 （变异方法）

### 1.str.substring(start,end); 两个参数都为正数，返回值：[start,end) 也就是说返回从start到end-1的字符

### 2.str.slice(start,end); 两个参数可正可负，负值代表从右截取，返回值：[start,end) 也就是说返回从start到end-1的字符

不建议用：str.substr(start,length); start参数可正可负，负数代表从右截取

除了 slice() 和 substr() 方法里的负值是代表从右截取，其他方法里的负值一律作为0处理

## 7、字符串分割成数组

### 1.str.split(separator,limit); 参数1指定字符串或正则，参照2指定数组的最大长度

### 2.toLowerCase方法 返回一个字符串，该字符串中的字母被转换成小写。

toUpperCase方法 返回一个字符串，该字符串中的所有字母都被转换为大写字母。

### 3.str.replace(rgExp/substr,replaceText) 返回替换后的字符串

### 4.str.match(rgExp); 正则匹配

注意：string中的方法返回值可能是一个新字符串，但都不会改变原字符串

# 1.String(字符串对象)

String对象—字符串对象

说明:凡是用“”或‘’包裹的都是字符串，表单文本框所有内容都是字符串

创建对象的方式

--直接赋值

--new String();构造函数创建

String对象的属性

--length属性

--prototype属性(原型)

String对象的方法

--charAt( ) 返回字符串中的第n个字符

--charCodeAt( ) 返回字符串中的第n个字符的Unicode编码

--concat( ) 连接字符串

--fromCharCode( ) 从字符编码创建—个字符串

--indexOf( ) 检索字符串

--lastIndexOf( ) 从后向前检索一个字符串

--localeCompare( ) 用本地特定的顺序来比较两个字符串是否相等

--match( ) 找到一个或多个正则表达式的匹配

--replace( ) 替换一个与正则表达式匹配的字符串

--search( ) 检索与正则表达式相匹配的字符串

--slice( ) 截取取一个字符串串

--split( ) 将字符串分割成字符串数组

--substr( ) 截取字符串

--substring( ) 返回字符串的一个子串

-- toLowerCase( ) 将字符串转换成小写

--toUpperCase( ) 将字符串转换成大写

--valueOf( ) 返回字符串

其他类型的数据转换为字符串

# 三．对象的方法

## Object.assign()

通过复制一个或多个对象来创建一个新的对象

## Object.keys()

返回一个包含所有给定对象自身可枚举属性名称的数组。

## Object.values()

返回给定对象自身可枚举值的数组。

## Object.create()

使用指定的原型对象和属性创建一个新对象。

## Object.is()

比较两个值是否相同。所有 NaN 值都相等（这与==和===不同）。