DAY07 字符串对象

2023年4月26日 18:08

字符串对象

1.疑问:

1.1.为什么基本类型具有属性和方法，应该是对象才有的??

1.2.为什么一切皆对象??

2.解答

2.1.为什么基本类型具有属性和方法??

因为number,string,boolean也是对象，不同于真正的对象，是包装对象

2.2.什么叫包装对象

当我们要使用number,string,boolean三个基本类型下面的属性和方法时，ES语法会临时将三个基本类型包装成对象类型，可以使用属性和方法

如果使用完成，即刻销毁包装过程。

```
// var arr = [1, 2, 3, 4]; //真正的对象
// arr.length = 2;
// console.log(arr.length);//2
// var str = 'hello';  // 包装对象
// str.length = 2;//ES语法会临时将三个基本类型包装成对象类型(new String('hello'))，如果使用完成，即刻销毁包装过程
// console.log(str.length);//5  这里也要包装，使用了属性
```

2.3.一切皆对象

number包装对象,string包装对象,boolean包装对象,null空对象,undefined值等于null

object对象

2.4.number,string,boolean三个基本类型都可以包装成对象类型

```
// var num1 = 1; //字面量  这是我们常用的创建数字的方法
// var num2 = new Number(1);//构造函数创建,这是由ES自动完成的，不要我们手动创建。
// console.log(num1 == num2);//true
// console.log(typeof num1); //number
// console.log(typeof num2); //object
字符串和布尔对象同理
```

类数组的概念 - 非常重要

类数组也叫伪数组，它拥有length属性，且拥有非负整数的索引，但是它又不能完全调用数组的方法

```
// 例如:
// 1.arguments对象
// function fn() {
//   console.log(arguments.length);//6
//   console.log(arguments[0]);//1
//   arguments.push(123);//报错
// }
// fn(1, 2, 3, 4, 5, 6);

// 2.字符串对象
// var str = 'abcdefg';
// console.log(str.length);//7
// console.log(str[0]);//a
// str.push(1234);//报错
```

unicode 编码

ASCII 只有这 128 个字符的编码结构，比较少，造成很多国家的文字无法存储，产生unicode编码

所以就出现了 unicode 编码，也叫（万国码，统一码）

unicode 对照表就是一个和 ASCII 一样的对照表，只不过变得很大很大，因为存储的内容特别的多

而且包含了世界上大部分国家的文字

我们的 UTF-8 就是一种 8 位的 unicode 字符集

字符串方法

核心方法:

1.substring
2.split
3.toUpperCase
4.toLowerCase
5.indexOf
6.trim

1.charAt(num)：返回对应字符串索引的内容。

```
// var str = 'aAbcdefg爱';
// console.log(str[0]);//a
// console.log(str.charAt(0));//a
```

2.charCodeAt(num)：返回对应的字符串索引的内容的unicode编码

```
// console.log(str.charCodeAt(0));//97
// console.log(str.charCodeAt(1));//65
// console.log(str.charCodeAt(str.length - 1));//29233
```

3.String.fromCharCode(num):输出num对应的字符

```
// console.log(String.fromCharCode(29233));//爱
// var str = '';
// for (var i = 0; i <= 50000; i++) {
//   str += String.fromCharCode(i);
// }
// box.innerHTML = str;
```

4.大小写转换

```
// var str = 'abcdEFG'
```

toUpperCase()：转换成大写.

```
// console.log(str.toUpperCase());//ABCDEFG
```

toLowerCase()：转换成小写.

```
// console.log(str.toLowerCase());//abcdefg
```

5.substring()：用来截取字符串的内容 - **非常重要**

```
// var str = 'abcdefg';
// console.log(str.substring());//没有参数，全部截取  abcdefg
// console.log(str.substring(1));//1个参数表示截取的起始位置  bcdefg
// console.log(str.substring(1, 4));//第1个表示起始位置 第2个表示结束位置  bcd
```

如果所有是负数，当作0处理，第一个参数比第二个大，自动交换位置

```
// console.log(str.substring(2, -5));//ab
// console.log(str.substring(2, 0));//ab
// console.log(str.substring(0, 2));//ab
```

6.split()：根据分隔符、将字符串拆分成数组

参1是分割字符串的符号。

参2用来限制数组的长度。

```
// var str = 'abcdefg';
// console.log(str.split('d'));//['abc', 'efg']
// console.log(str.split(''));//['a', 'b', 'c', 'd', 'e', 'f', 'g']
// console.log(str.split('', 3));//['a', 'b', 'c']
```

```
// console.log(str.split());//['abcdefg']
```

反转字符串

```
// console.log(str.split('').reverse().join(''));//gfedcba
```

7.trim()：创建一个字符串的副本，删除前置及后缀的所有空格，然后返回结果

trimStart和trimEnd：分别删除前置或者后缀的所有空格

```
// var str = '     abc  de  fg        ';
// console.log('(' + str + ')');
// console.log('(' + str.trim() + ')');
// console.log('(' + str.trimStart() + ')');
// console.log('(' + str.trimEnd() + ')');
```

8.substr(start, length)：用来截取字符串的内容，参1是开始的索引，参2表示截取的长度。

```
// var str = 'abcdefg';
// console.log(str.substr(1, 3));//bcd
// console.log(str.substr(2, 4));//cdef
// console.log(str.substr(1, -2));//负数当作0处理
// console.log(str.substr(-5, 2));//cd    负数从后往前数
```

9.slice() 和数组方法使用一致。

10.indexOf和lastIndexOf():区别是如果第二个参数为负数，数组是从后往前找，字符串是当做0处理。

11.concat( )：和数组方法使用一致。

多维数组

多维数组:是一种嵌套的数组

二维数组:是以一维数组作为数组元素的数组, 即数组中的数组

var arr = ['zhangsan', [1, 2, 3, [4, 5, [7, 8, 9]]]]

1.如何获取多维数组里面的值。

利用数组的下标

```
// var arr = [100, [200, 300], [400, 500, [600, [700, 800]]]];
// console.log(arr[0]);//100
// console.log(arr[1]);//[200, 300]
// console.log(arr[1][0]);//200
// console.log(arr[1][1]);//300
```

```
// console.log(arr[2][2][1][1]);//800
```

2.如何设置多维数组(理解二维数组即可)。

必须告知程序数组当前的这一项里面还是数组。

认真理解索引的配置

```
// var arr = [];
// arr[0] = [];//告知数组的第一项还是数组
// arr[1] = [];//告知数组的第二项还是数组
// arr[0][0] = 'zhangsan';
// arr[1][0] = 'wangwu';
// arr[2][0] = 'lisi';//这里报错，因为没有告知数组第三项还是数组
// console.log(arr);
```

对象数组

数组取值通过中括号([])配合索引获取

对象取值通过中括号([])和点符号(.)获取

```
// 获取对应的值
// var obj1 = {
//   names: ['zhangsan', 'lisi', 'wangwu'],
//   ages: [18, 19, 20],
//   info: {
//     address: ['北京', '杭州']
//   }
// }
// console.log(obj1.names);//['zhangsan', 'lisi', 'wangwu']
// console.log(obj1.names[1]);//lisi
// console.log(obj1.info.address[1]);//杭州
// console.log(obj1['info']['address'][1]);//杭州

// var arr = [
//   { name: 'zhangsan', age: 18 },
//   { name: 'lisi', age: 17 },
//   { name: 'wangwu', age: 19 }
// ];
// console.log(arr[1].age);//17
// console.log(arr[1]['age']);//17
```

按照年龄的升序进行排序

sort排序数字传比较函数，排序字符串无需比较函数

注意：只能对对象属性值是数字的进行排序，如果是字符串不能使用sort

```
// var result = arr.sort(function (a, b) {//a,b代表数组项，根据arr，知道a,b表
```

示两个对象

```
//   return a.age - b.age;
// });
// console.log(result);
```

数组扁平化(将一个多维数组转换成一维数组)

1.利用转字符串

```
// var arr = [1, 2, [3, 4, [5, 6, [7, 8, [9, [10]]]]]];
// console.log(arr.toString().split(',').map(function (v) { return
+v }));//[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

2.利用数组提供的方法flat，flat专门实现扁平化，参数扁平的维度。

设置维度的参数一般都可以设置为：Infinity(无穷)

```
// var arr = [1, 2, [3, 4, [5, 6, [7, 8, [9, [10]]]]]];
// console.log(arr.flat(Infinity));//[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```