JavaScript 数据类型 07

JS

JS的数据类型种类

> 类型：类型就值的内部特征，用来决定值在操作的时候的行为，以区别于其它的值。

基本的数据类型：

Undefined类型

> 只有一个值 undefined

- 声明变量未初始化（未赋值）
- 访问对象身上不存在的属性的时候
- 数组只给定长度，并没有值的时候，其中的值就是undefiend。
- ...

Null类型

> 在JS里面代表空，只有一个值：null

- null 代表一个空
- undefined == null 返回的是 true
- undefined 代表声明未赋值，但是null代表没有。

字符串类型

> 由0个或者多个字符组成的集合，并且放在一对单引号或者双引号中，并且引号是成对儿出现的。

- 属性：字符串.length, 用来查看字符串的长度。只能读不能写。
- 字符串在做 + 运算的时候，永远做的都是字符串的拼接。

当需要使用一些特殊字符当作字符串的时候需要进行一下转义。

```
var str = 'hellow yu quan \'er\'';

console.log(str);

//  反斜杠 的意思就是 转义
```

数字类型

> JS中默认都是以10进制来表示的。分为整数和浮点数（小数儿）。

能表示的数字范围：

- 最大整数：1.7976931348623157e+308，查看这个数值使用 Number.MAX_VALUE
- 最小浮点数：5e-324，查看这个数值使用 Number.MIN_VALUE

JS在表示数字的时候会不失时机的把小数变为整数。

```
var a = 1.0;  // 1
var b = 1.2000; // 1.2
```

数字类型里面由一个特殊的值：NaN，代表 不是数字的数字（不是一个数字，但是是一个数字类型），当出现NaN这个结果的时候，只代表一件事：当前的运算失败了。

当数字在运算的时候，一旦超出了运算范围，如果是正数就会得到 正无穷（Infinity），如果运算是个负数，就是会得到负无穷（-Infinity）。

> 需求1：得到数组中的最大值

```
var arr = [10,-7,23,65,1,89,231,-23,54,90];
var n = -Infinity;
for(var i=0; i<arr.length; i++){
    if(n < arr[i]){
```

```
            n = arr[i];
        }
    }
    console.log(n);
```

数字运算时候的一些注意事项：

```
'abc' - 10  // NaN
NaN * 1 // NaN
0 / 0  // NaN
非0正数 / 0  // Infinity
非0负数 / 0  // -Infinity
Infinity / Infinity  // NaN
Infinity - Infinity   // NaN
Infinity % Infinity   // NaN
```

> 在JS中小数最多精确到17位，在小数运算的时候不是很精确。

- 不要去拿小数运算的结果去做判断。

布尔类型:

> 布尔类型只有2个值：true 和 false，分别代表 真 和 假。

引用数据类型

> 引用数据类型 只有一种：对象，除了上面的的几种基本类型以外，所有的都是对象。

- 对象： {}
- 数组：[]
- 函数：function(){}
- HTML中的元素
- ...

所有的对象都可以进行属性操作。

数据类型检测(1)

> 通过一元操作符 typeof 可以来检测数据类型。注意：一元操作符返回的数据类型是字符串。

typeof 可能返回的类型种类：

- "number" 表示数字
- "string" 表示字符串
- "boolean" 表示布尔值
- "undefined" 表示undefined
- "object" 表示一个对象或者null
- "function" 表示一个函数

JS中的类型转换

> 在JS中所有的数据类型，最终只能转换成已下三种数据类型中的一种。

- 数字
- 字符串
- 布尔值

JS中可以将其它数据类型转换成数字类型的方法有下面三个：

> 1. Number();
> 2. Number.parseInt();
> 3. Number.parseFloat();

```
// Number
// 1. 如果扔给Number的数据是一个数字，那么返回的是这个数字本身。
Number(1.2) // 1.2

```

```
// 2. 如果扔给Number的数据是一个字符串，有已下几种情况
Number('00000000123');  // 123;
Number('1.23')  // 1.23
// 如果字符串中有非数字，那么转换结果就是NaN
Number('a1.0') // NaN
Number('')  // 0;
Number('  ') // 0;

// 3.如果传入的是一个undefined
Number(undefined)  // NaN

// 4.如果传入的是一个 null
Number(null)  // 0

// 5.如果传入的是一个布尔值
Number(true);  // 1
Number(false); // 0

// 6.如果传入的是一个数组，如果数组里面只有一个数据，并且这个数据是一个数字或者一个纯数字的字符串都可以被转换，如果有多个数据就是NaN，如果是空数组会被转换成0；
Number([])  // 0
Number([768])  // 768
Number(['a'])  // NaN
Number([1,2])  // NaN
Number(['0123']) // 123

// 7.如果传入的是一个{}
Number({})  // NaN

// 8.如果传入的是一个函数
var fn = function (){};
Number(fn);  // NaN
```

```
// Number.parseInt();

// 如果只传入一个整数，那么传入什么返回什么
Number.parseInt(1)  //  1
// 如果传入的是一个浮点数，那么会去掉小数点和小数点后面的数，也就是会把小数转换为整数。
Number.parseInt(1.32423423423) // 1

// 如果传入的是一个字符串，分为已下几种情况：1.如果字符串是纯数字组成的，那么就和传入的是数字一样的。2.如果字符串是数字和字符组成的，那么就返回第一个不是数字之前的部分。3.如果字符串首字符不是数字，那么就会返回NaN。4.如果是一个空字符串，返回的是一个N
```

```
aN。5.如果字符串前面是空格，那么会被忽略。

Number.parseInt('1.2'); // 1
Number.parseInt('');  // NaN
Number.parseInt('  123.2gdfddf'); // 123

// 如果是传入undefined
Number.parseInt(undefined);  // NaN

// 如果是传入的 null
Number.parseInt(null);  // NaN

// 如果是对象类型
Number.parseInt([]);  // NaN
Number.parseInt({}); // NaN
Number.parseInt(function (){}); // NaN

// 如果是一个布尔值
Number.parseInt(true);  // NaN

/*
    常用于取到字符串前面整数部分和把小数转换为整数。
*/

// 除了parseInt可以把小数转换为整数以外，还可以使用 ~~ num  /  num >> 0;
var a = 1.2;

// ~~a  == a >> 0;
```

```
// Number.parseFloat();

// 1.如果传入的是一个数字，传入什么返回什么

Number.parseFloat(1) ; // 1
Number.parseFloat(1.2) // 1.2

// 2.如果传入的是一个字符串

Number.parseFloat('1') // 1
Number.parseFloat('1.2') // 1.2
Number.parseFloat('1.2abc') // 1.2
Number.parseFloat('1.2000abc') // 1.2
Number.parseFloat('000001.2000abc') // 1.2
Number.parseFloat('abc1.2') // NaN
```

```
Number.parseFloat('')  // NaN

// 3. undefined
Number.parseFloat(undefined); // NaN

// 4 null
Number.parseFloat(null);  // NaN

// 5 对象
Number.parseFloat([])  // NaN
Number.parseFloat({})  // NaN

// 6 布尔值
Number.parseFloat(true)  // NaN
Number.parseFloat(false)  // NaN
```

如何判断是不是NaN

1 使用内置的函数 `Number.isNaN()`，如果传入的参数是一个NaN，那么就返回true，如果是一个运算失败的表达式，返回的也是一个true，其它情况都是fasle。

2 如果说一个数据和它自己都不相等，那么就是NaN;

```
if(a != a){
    alert(a + '是NaN');
}
```

JS中转成字符串类型的操作：

1 可以使用内置的 **String()**

```
String(1.2)
"1.2"
String(0)
"0"
String(NaN)
"NaN"
String(true)
"true"
String(false)
"false"
```

```
String(undefined);
"undefined"
String(null);
"null"
String(function (){});
"function (){}"
String([1,2,3,4,5]);
"1,2,3,4,5"
String({a:1});
"[object Object]"
String({});
"[object Object]
```

2 还可以直接用 数据 + ''。除了 函数 还有 对象（不是数组）。

```
[1,2,3] + '' // "1,2,3"
{} + '' // 0
undefined + '' //"undefined"
true + '' // "true"
null + '' // "null"
```

JS中转换为布尔值

> 可以通过内置的 `Boolean()` 把其它的数据类型转换为布尔值。

```
// 1.如果传入的是数字，0会被转为false，其它的数字会被转为true，如果是NaN，那么会被转为false。
Boolean(0)  // false
Boolean(1.23213)  // true
Boolean(NaN) // false

// 2. undefined
Boolean(undefined)  // false

// 3. null
Boolean(null)  // false

// 4. 字符串，除了空字符串会被转为false，其它的都会被转为true
Boolean('');  // false
Boolean(' ');  // true

```

```
// 5.对象，所有的对象都是真的。
Boolean([]); // true
Boolean(function (){}); // true
```

以上所有的类型转换都是显示强制类型转换，对应的就有隐式强制类型转换。隐式类型转换的情况会非常多。以下式部分例子：

```
// 字符串数字和数字做运算
'1' + 1  // 11 字符串拼接

'1' * 2  // 2
'3' - 1 // 2
'3' / 1 // 3
'1' % 2  // 1;

// 对象和数字做运算
[] - 5 // -5

// 在做非严格比较的时候（==）：
// 1.如果等号两边有一边式布尔值，那么会被隐式的转换为数字，false会被转换为0， true会被转换为 1。
// 2. 如果是字符串数字和数字做比较的时候，那么字符串会被转换为数字做比较。
// 3. 如果是 undefined 和 null 做比较的时候，返回true， 不会发生类型转换。
// 4.如果是对象在做比较的时候，需要看比较的对象的引用是否相同。
```

> 需求2：编写一个简单的计算器。

```
<body>
  <input id="text1" type="text">
  <select class="select" name="">
    <option value="0">+</option>
    <option value="1">-</option>
    <option value="2">*</option>
    <option value="3">/</option>
  </select>
  <input id="text2" type="text">
  <button id="res">=</button>
  <input id="text3" type="text">
  <script>
    var inputs = document.querySelectorAll('input');
    var btn = document.querySelector('#res');
```

```
15.       var select = document.querySelector('.select');
16.
17.       // alert(select.value)  可以拿到 select 标签的 value 值
18.
19.       btn.onclick = function (){
20.         var t1 = inputs[0].value*1;
21.         var t2 = inputs[1].value*1;
22.         var val = select.value;
23.         // alert(val)
24.         if(val == '0'){
25.           inputs[2].value = t1 + t2;
26.         }else if(val == '1'){
27.           inputs[2].value = t1 - t2;
28.         }else if(val == '2'){
29.           inputs[2].value = t1 * t2;
30.         }else{
31.           inputs[2].value = t1 / t2;
32.         }
33.       }
34.     </script>
35.   </body>
```