JS中数据类型转换汇总

JS中的数据类型分为
【基本数据类型】
数字 number
字符串 string
布尔 boolean
空 null
未定义 undefined
【引用数据类型】
对象 object
普通对象
数组对象 (Array)
正则对象 (RegExp)
日期对象 (Date)
数学函数 (Math)
…
函数 function

真实项目中，根据需求，我们往往需要把数据类型之间进行转换

把其它数据类型转换为number类型

`1.发生的情况`

- isNaN检测的时候：当检测的值不是数字类型,浏览器会自己调用Number方法把它先转换为数字，然后再检测是否为非有效数字

```
1.  isNaN('3') =>false
2.    Number('3')->3
3.    isNaN(3)->false
4.
5.  isNaN('3px') =>true
6.    Number('3px')->NaN
```

```
  isNaN(NaN)->true
```

- 基于parseInt/parseFloat/Number去手动转换为数字类型
- 数学运算：+ - * / %，但是“+”不仅仅是数学运算，还可能是字符串拼接

```
'3'-1 =>2
  Number('3')->3
  3-1->2

'3px'-1 =>NaN

'3px'+1 =>'3px1' 字符串拼接

var i='3';
i=i+1; =>'31'
i+=1; =>'31'
i++; =>4  i++就是单纯的数学运算，已经摒弃掉字符串拼接的规则
```

- 在基于“==”比较的时候，有时候也会把其它值转换为数字类型
- …

2.转换规律

```
//=>转换的方法：Number（浏览器自行转换都是基于这个方法完成的）

【把字符串转换为数字】
只要遇到一个非有效数字字符，结果就是NaN
'' ->0
' ' ->0 空格(Space)
'\n' ->0 换行符(Enter)
'\t' ->0 制表符(Tab)


```

```
【把布尔转换为数字】
true ->1
false ->0

【把没有转换为数字】
null ->0
undefined ->NaN

【把引用类型值转换为数字】
首先都先转换为字符串（toString），然后再转换为数字（Number）
```

把其它类型值转换为字符串

1.发生的情况

- 基于alert/confirm/prompt/document.write等方法输出内容的时候，会把输出的值转换为字符串，然后再输出

```
alert(1) =>'1'
```

- 基于“+”进行字符串拼接的时候
- 把引用类型值转换为数字的时候，首先会转换为字符串，然后再转换为数字
- 给对象设置属性名，如果不是字符串，首先转换为字符串，然后再当做属性存储到对象中（对象的属性只能是数字或者字符串）
- 手动调用toString/toFixed/join/String等方法的时候，也是为了转换为字符串

```
var n=Math.PI;//=>获取圆周率:
n.toFixed(2) =>'3.14'

var ary=[12,23,34];
ary.join('+') =>'12+23+34'
```

- ...

2.转换规律

```
//=>调用的方法：toString

【除了对象，都是你理解的转换结果】
1 ->'1'
NaN ->'NaN'
null ->'null'
[] ->''
[13] ->'13'
[12,23] ->'12,23'
...

【对象】
{name:'xxx'} ->'[object Object]'
{} ->'[object Object]'
不管是啥样的普通对象，最后结果都一样
```

把其它值转换为布尔类型

1.发生的情况

- 基于!/!!/Boolean等方法转换
- 条件判断中的条件最后都会转换为布尔类型
- ...

```
if(n){
  //=>把n的值转换为布尔验证条件真假
}

if('3px'+3){
  //=>先计算表达式的结果'3px3'，把结果转换为布尔true，条件成立
}
```

2.转换的规律

只有“0/NaN/”/null/undefined”五个值转换为布尔的false,其余都是转换为true

特殊情况：数学运算和字符串拼接“+”

```
1. //=>当表达式中出现字符串，就是字符串拼接，否则就是数学运算
2.
3. 1+true =>2 数学运算
4. '1'+true =>'1true' 字符串拼接
5.
6. [12]+10 =>'1210' 虽然现在没看见字符串，但是引用类型转换为数字，首先会转换为字符串，所以变为了字符串拼接
7. ({})+10 =>"[object Object]10"
8. []+10 =>"10"
9.
10. {}+10 =>10 这个和以上说的没有半毛钱关系，因为它根本就不是数学运算，也不是字符串拼接，它是两部分代码
11.   {} 代表一个代码块（块级作用域）
12.   +10 才是我们的操作
13.   严格写法：{}; +10;
```

思考题：

```
1. 12+true+false+null+undefined+[]+'珠峰'+null+undefined+[]+true
2. =>'NaN珠峰nullundefinedtrue'
3.
4. 12+true ->13
5. 13+false ->13
6. 13+null ->13
7. 13+undefined ->NaN
8. NaN+[] ->'NaN'
9. 'NaN'+'珠峰' ->'NaN珠峰'
```

```
...
'NaN珠峰trueundefined'
'NaN珠峰trueundefined'+[] ->'NaN珠峰trueundefined'
...
=>'NaN珠峰trueundefinedtrue'
```

特殊情况：“==”在进行比较的时候，如果左右两边的数据类型不一样，则先转换为相同的类型，再进行比较

对象==对象：不一定相等，因为对象操作的是引用地址，地址不相同则不相等

```
{name:'xxx'}=={name:'xxx'} =>false
[]==[] =>false

var obj1={};
var obj2=obj1;
obj1==obj2 =>true
```

————————————————-上面是重点强调的

对象数字：把对象转换为数字
对象布尔：把对象转换为数字，把布尔也转换为数字
对象字符串：把对象转换为数字，把字符串也转换为数字
字符串数字：字符串转换为数字
字符串布尔：都转换为数字
布尔数字：把布尔转换为数字
————————————————不同情况的比较，都是把其它值转换为数字，然后再进行比较的

null==undefined：true
null===undefined：false

null&&undefined和其它值都不相等

NaN==NaN：false
NaN和谁都不相等包括自己
————————————————以上需要特殊记忆

```
1==true =>true
1==false =>false
2==true =>false  规律不要混淆，这里是把true变为数字1


[]==true: false  都转换为数字 0==1
![]==true: false

[]==false: true  都转换为数字 0==0
![]==false: true  先算![]，把数组转换为布尔取反=>false
=>false==false
```