深度剖析JS类型转换规律（显示类型转换和隐式类型转换）

jCodeLife
0.077 2020.08.04 19:16:06 字数 1,910 阅读 27

编辑文章

JavaScript的数据类型有number、string、boolean、undefined、null、object、function以及ES6新增的symbol，我们可以通过typeof运算符来查看变量的数据类型，如下：

```
typeof "Alice"                // 返回 string
typeof 3.14                   // 返回 number
typeof NaN                    // 返回 number
typeof false                  // 返回 boolean
typeof [1,2,3,4]              // 返回 object
typeof {name:'Alice', age:12} // 返回 object
typeof new Date()             // 返回 object
typeof function () {}         // 返回 function
typeof a                      // 返回 undefined (如果 a没有声明)
typeof null                   // 返回 object
```

注意：
NaN 的数据类型是 number
数组、日期对象、null的数据类型都是 object
未定义变量的数据类型为 undefined
因为普通对象和数组返回的类型都是object，所以我们得使用 constructor 属性来查看对象是否为数组 (包含字符串 "Array"):

```
function isArray(arr){
  return arr.constructor.tostring().indexof("Array")>-1;
}
```

同样判断是否是日期对象也可以使用constructor来查看：

```
function isDate(myDate) {
    return myDate.constructor.toString().indexOf("Date") > -1;
}
```

变量有数据类型，但并不是一成不变的，有的时候我们希望变量的数据类型发生变化，那么可以手动调用对应的JavaScript方法，有的时候 JavaScript 自身会自动的发生类型转换，我们把手动调用的称为显示类型转换，自动的称为隐式类型转换。

一、显示类型转换

显示类型转换是手动的调用对应的类型转换函数

1. Number()
   千方百计将传入的数据转变成数字，转变不了也会将其变成NaN数字类型。注意：
   Number()不会截断数字

```
var num = Number("123") ;     // 返回 123
Number(true) ;                // 返回 1
Number(false) ;               // 返回 0
Number("true") ;              // 返回 NaN
Number("false") ;             // 返回 NaN
```

推荐阅读
这次，彻底弄清js的继承方式
阅读 478
javascript new一个对象的时候，内部发生了什么
阅读 863
学Vue，就要学会vue JSX（一）
阅读 333
TypeScript基础学习总结
阅读 18
前端性能优化
阅读 543

若传入的类型是数值，转换后还是原来值
若传入的类型是字符串，且该字符串是数字字符串，则转换为对应数值；空字符串则转为0，其他类型字符串都为NaN
若传入的类型是布尔值，true为1，false为0
若传入的是undefined，结果为NaN
若传入的是null，返回0

2. parseInt()

作用1：
用于将传入数据转变成整型

```
parseInt(null);//NaN
parseInt(true);//NaN
parseInt(false);//NaN
parseInt(undefined);//NaN

parseInt(123.9);//123 不会四舍五入 直接截断
parseInt("123abc");//123 从字符串首位开始到非字符串数字截断
```

parseInt不会管那么多，非数字只会转成数字类型，值为NaN
带小数数字直接截断，不会四舍五入
字符串数字或者开头为数值字符串的，从字符串首位开始到非字符串数字截断

作用2：
当传入两个参数时parseInt(num,radix),表示以radix进制为基底，将radix进制的num转成10进制
比如：

```
var num=10101010;
var temp = parseInt(num,2)
```

以上是将2进制的数num，转成10进制

3. parseFloat(s)

作用：用于将传入数据转变成浮点型，作用跟parseInt(s)相似,只是它带小数了：

```
parseFloat(null);//NaN
parseFloat(true);//NaN
parseFloat(false);//NaN
parseFloat(undefined);//NaN

parseFloat(123.9);//123.9
parseInt("123.1abc");//123.1
```

若字符串前几位是数字，它会检测到第一个小数点后的非数字字符串截断，若有第二个小数点，就到第二小数点前截断。

4. toString()

作用1：转成字符串

作用2：将10进制的数据，转成任意进制。如 toString(16)

推荐阅读

这次，彻底弄清js的继承方式
阅读 478

javascript new一个对象的时候，内部发生了什么
阅读 863

学Vue，就要学会vue JSX（一）
阅读 333

TypeScript基础学习总结
阅读 18

前端性能优化
阅读 543

所以我们可以通过parseInt(num,某进制)转成10进制，通过.toString(任意进制)将数字转成任意进制，实现进制之间的转换。

5. String(val)

   作用：将任意类型数据转换为字符串

   数值变成数字字符串

   字符串任是字符串

   布尔值true为”true”,false为”false”

   undefined、null变成对应的字符串“undefined”和“null”

6. Boolean()

   转换为布尔型的值

   被认为是false的有：undefined、null、0、NaN、""空字符串

```
Boolean(undefined);//false
Boolean(null);//false
Boolean(-0);//false
Boolean(0);//false
Boolean(+0);//false
Boolean(NaN);//false
Boolean("");//false
```

二、隐式类型转换

有的时候js会偷偷的进行类型转换，转换了也没有告知我们，我们把这种称之为隐式类型转换。隐式类型转换内部也是调用显示的方法。

1. 加号

   (1) 数字+任意类型

```
<script>
1+1;//2
1+null;//1
1+undefined;//NaN
1+true;//2
1+false;//1
1+"";//"1"
1+[];//"1"
1+[2];//"12"
1+[2,3];//"12,3"
1+{};//"1[object Object]"
1+{a:1,b:2};//"1[object Object]"
</script>
```

规律：

- 数字+数字：直接相加
- 数字+字符串：调用的是String()；最终都为字符串
- 数字+null：调用的Number()；Number(null)为0
- 数字+undefined：调用Number()；Number(undefined)为NaN
- 数字+布尔值：调用的Number()；Number(true)为1，Number(false)为0
- 数字+数组：调用的是String()；String([])为""，String([2,3])为"2,3"
- 数字+对象：调用的也是String()；String({})为[object Object]

(2) 字符串+任意类型

```
<script>
"1"+1;//"11"
```

推荐阅读

这次，彻底弄清js的继承方式
阅读 478

javascript new一个对象的时候，内部发生了什么
阅读 863

学Vue，就要学会vue JSX（一）
阅读 333

TypeScript基础学习总结
阅读 18

前端性能优化
阅读 543

```
"1"+[];//"1"
"1"+[2];//"12"
"1"+[2,3];//"12,3"
"1"+{};//"1[object Object]"
"1"+{a:1,b:2};//"1[object Object]"
</script>
```

记住一条规律：任何类型加字符串都得字符串。

(3) 布尔值+任意类型
上面两种情况有包含到两个规律：

- 布尔值+数字，调用的Number(),
- 布尔值+字符串，调用的是String()
  再来看布尔值+其他类型的情况

```
true + false;//1
true + null;//1
true + undefined;//NaN
true + [];//"true"
true + {};//"true[object Object]"
```

- 布尔值+布尔值，调用的Number()
- 布尔值+null，调用的Number()
- 布尔值+undefined，调用的Number()
- 布尔值+数组，调用的是String()
- 布尔值+对象，调用的也是String()；String({})为[object Object]

（4）null+任意类型
根据上面情况我们可以得出以下规律：

- null+数字，调用的Number()
- null+布尔值串，调用的是Number()
- null+字符串，调用的是String()

再来看看其他情况

```
null + null; //0
null + undefined; //NaN
null + [];//"null"
null + {};//"null[object Object]"
```

- null+null，调用的Number()
- null+undefined，调用的是Number()
- null+数组，调用的是String()
- null+对象，调用的是String()

(5) undefined+任意类型
根据上面情况我们可以得出以下规律：

- undefined+数字，调用的Number()，值为NaN
- undefined+布尔值串，调用的是Number()，值为NaN
- undefined+null，调用的是Number()，值为NaN

推荐阅读
这次，彻底弄清js的继承方式
阅读 478
javascript new一个对象的时候，内部发生了什么
阅读 863
学Vue，就要学会vue JSX（一）
阅读 333
TypeScript基础学习总结
阅读 18
前端性能优化
阅读 543

```
undefined + {};// undefined[object Object]
```

- undefined+undefined，调用的是Number()
- undefined+数组，调用的是String()
- undefined+对象，调用的是String()

(6) 数组+任意类型
规律：都调用的是String()

- []+数字，调用的String()
- []+布尔值串，调用的String()
- []+null，调用的String()
- []+字符串，调用的是String()
- []+undefined，调用的String()
- []+[]，调用的是String()
- []+{}，调用的是String();
  特别注意[]+{}为"[object Object]"，但是{}+[]为0

(7) 对象+任意类型
规律：
对象+任意类型除数组，都是调用String()
对象+数组时，比较奇怪：比如{}+[]为0，调用的既不是String也不是Number()，因为它返回的是0，按道理是Number，但是Number({})的值是NaN，我继续来看：

```
{}+[];//0
{a:1}+[]; //0
{a:1}+[12]; //12
{a:1}+[12, 1]; //NaN
```

经过观察，我们初步可以发现一个规律：当对象+数组时，返回的就是Number(数组)的值。有待进一步验证。

(8) function类型+任意类型
调用的是String()

2. 减号、乘号、除号、取余
   调用的都是Number()
3. ++ -- 一元正负
   调用的都是Number()
4. isNaN()
   查看一个数是否是NaN，内部调用的是Number()
5. if(表达式)
   表达式里面经常会放逻辑运算符&& || ！，但最终会把表达式的结果运算出来，通过Boolean()转换成true/false
6. 比较运算符也存在类型转换

```
<小于
>大于
<=小于等于
>=大于等于
==等于
!=不等于
```

推荐阅读
这次，彻底弄清js的继承方式
阅读 478
javascript new一个对象的时候，内部发生了什么
阅读 863
学Vue，就要学会vue JSX（一）
阅读 333
TypeScript基础学习总结
阅读 18
前端性能优化
阅读 543

- 数与数比较，比较的是数值大小
- 仅一个是数字，调用Number将另外一个变成数字，在进行数值比较
- 字符串与其他非数值数字比较，将另外一个通过String()变成字符串，比较逐位的ASCII码
- 两个中任意一个既不是数字也不是字符串，将两个比较对象转成数字或者字符串，再进行比较。
- 无法转成字符串或数字，返回false
- 与NaN比较，返回false

```
true > false;//true
100>10>0;//true
100>10>1;//false
null>0;//false
null<0;//false
null==0;//false
undefined>0;//false
undefined<0;//false
undefined==0;//false
null==undefined;//true
```

7. 也存在不发生类型转换的，比如：

   ===绝对等于

   !==绝对不等于

```
1 === "1";//false
1 === true;//false
null === undefined;//false
```

参考资料

https://www.runoob.com/js/js-type-conversion.html

https://blog.csdn.net/q535999731/article/details/79643665

http://c.biancheng.net/view/5445.html

jCodeLife 书山有路勤为径，学海无涯苦作舟

总资产5 (约0.53元) 共写了9.8W字 获得119个赞 共19个粉丝

被以下专题收入，发现更多相似内容

投稿管理

+收入我的专题 领跑计划笔记

推荐阅读

这次，彻底弄清js的继承方式
阅读 478

javascript new一个对象的时候，内部发生了什么
阅读 863

学Vue，就要学会vue JSX（一）
阅读 333

TypeScript基础学习总结
阅读 18

前端性能优化
阅读 543