数据类型转换

2020年02月24日, 星期一　　08:38

1.转换为 Object 类型

对象转换为自身

undefined 和 null 转换为空对象{}

string/number/boolean 转换为包装对象

强制转换：Object()

值	结　果
(无参数调用)	{}
undefined	{}
null	{}
布尔值 bool	new Boolean(bool)
数字　num	new Number(num)
字符串 str	new String(str)
对象 obj	obj (未改变，不需要转化)

2.Object 转换为 Number

先调用 valueOf() 方法，结果为原始值，返回；

再调用 toString() 方法，结果为原始值，返回；

原始值转换为 Number 类型

```
> var a={};
  Number(a);
< NaN
> a.valueOf()
< ▶{}
> a.valueOf().toString()
< "[object Object]"
> Number(a.valueOf().toString())
< NaN
```

```
> var a = [2];
  Number(a);
< 2
> a.valueOf()
< ▶[2]
> a.valueOf().toString()
< "2"
> Number(a.valueOf().toString())
< 2
```

```
> var a = [2,3];
  Number(a);
< NaN
> a.valueOf()
< ▶(2) [2, 3]
> a.valueOf().toString()
< "2,3"
> Number(a.valueOf().toString())
< NaN
```

只有一个值的数组，如：[3]，转换成Number的结果是3

有多个值的数组，如[2，3]，转换成Number的结果是NaN

（“2，3”中与一个逗号，是非纯数字的字符串，非纯数字的字符串转换成Number类型结果为NaN）

空数组[]转后为Number结果是0

3.Object 转换为 String

先调用 toString() 方法，结果为原始值，返回；

再调用 valueOf() 方法，结果为原始值，返回；

原始值转换为 String 类型

```
> var a = {};
  String(a);
< "[object Object]"
> a.toString()
< "[object Object]"
```

```
> var a = [1,2,3];
  String(a);
< "1,2,3"
> a.toString()
< "1,2,3"
```

```
> var a = function(){};
  String(a);
< "function(){}"
> a.toString()
< "function(){}"
```

空数组[]转后为string结果是空字符串 “ ”

数组[3]转换为String结果是字符 “3”

4.Object 转换为 Boolean

任意对象转换为布尔值为 true，包括空对象

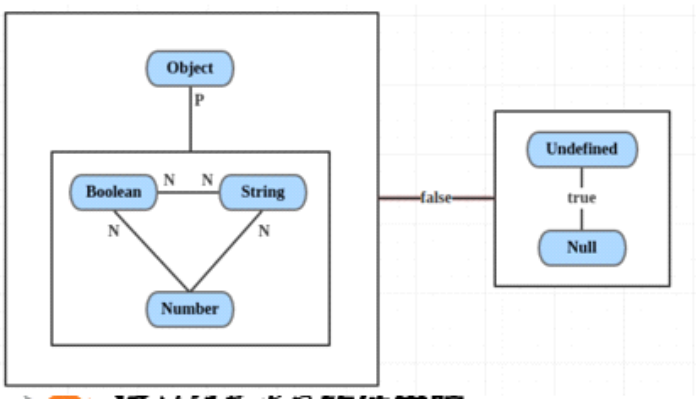
真值（truthy）与假值（falsy）

在 JavaScript 中，真值指的是在强制转换布尔值时，转换后的值为真的值。所有值都是真值，除非它们被定义为假值（即**除 false、0、""、null、undefined 和 NaN 以外皆为真值**）。

5.

N表示ToNumber操作，即将操作数转为数字。可以用JS中的Number()函数来等价替代。

P表示ToPrimitive操作，即将操作数转为原始类型的值。具体通过valueOf和toString方法。



undefined == null，结果是true。且它俩与所有其他值比较的结果都是false。

```
> var a;
  var b = null;
  a == b;
< true
```

String == Boolean，需要两个操作数同时转为Number。

String/Boolean == Number，需要String/Boolean转为Number。

Object == Primitive，需要Object转为Primitive。

6.练习：分析 **console.log([ ] == [ ])** 输出的值

两个值都是对象 (引用值) 时，比较的是两个引用值在内存中是否是同一个对象。 虽然左操作数和右操作数同为空数组， 但此 [ ] 非彼 [ ]，在内存中是两个互不相关的空数组， 所以结果为 false。

7.练习：分析 **console.log([ ] == ![ ])** 输出的值

涉及到了 JavaScript 的运算符优先级 、宽松相等（即 ==）的判断过程以及类型转换

1. 等号右边有！，优先级比 == 更高，优先计算右边的结果。 [ ] 为非假值，
   所以右边的运算结果为 false，即：![ ] ==> false
2. == 的两边分别是 object 和 boolean 类型的值
   把 object 转换成 number 类型，需要对 object 进行 ToNumber 操作，即
   Number([ ].valueOf()) ==> 0
   boolean 类型的值时先把这个值转换成 number 类型，右边转换成了 0，
   即Number(false) ==> 0

8.

```
> var str = 'hello';
  var stt = new String('hello');
< undefined
> str == stt
< true
```