### JavaScript的浮点数

### （一）关于结果返回的格式

大多数编程语言都有集中数值型数据类型，但是Js中就一种。可以通过typeof查看数字的类型，返回”number”。不管是整数还是浮点数，Js都将他们简单的归类为数字。

事实上，所有js中的数字都是双精度浮点数。但是这并不代表js无法表示整数。因为双精度浮点数能够完美的表示高大53位精度的整数。从-2^53 ~ 2^53。

0.1 \* 1.9 // 0.19

-99 + 100 // 1

21 - 12.3 // 8.7

2.5 / 5 // 2.5

21 % 8 // 5

计算的结果就是浮点数，不过如果的得出的结果是整数，不管进行运算的各个数是什么类型的（不管带不带小数点），结果如果是整数，就显示整数，比如1.0 +2.00 结果就是3。

而不是3.0或者3.00或者3.0000...。

如果计算的结果是小数，就直接把正确的小数结果显示出来，小数能显示到几位就显示到几位。

比如结果是2.17 3.1415 1.33333333..... 不存在2.170000.... 3.14150000...的情况。

**如果想要看看数字的几进制可以这样做**

*console*.log((8).toString(2)); // "1000"

*console*.log(typeof (8).toString(2)); // "string"



### 关于计算的不准确性

浮点数的计算结果不精确是出了名的。甚至一些看起来简单的基本运算都会产生不正确的结果。

*console*.log(0.1 + 0.2); // 0.30000000000000004



### 处理货币等数据时的推荐计算方式

因为浮点数的计算偶尔存在不精确性，因此尽量将浮点数的计算转换成整数的计算。整数的计算还是很精确的。

比如程序员在处理货币的运算时，通常会按照比例将数值转换为最小的货币单位在进行计算，这样就可以以整数进行计算。

例如：计算美元，就换成美分。一美元十美分。

(11 + 22) + 35; // 68