**原因：**

在JavaScript中数字是以[IEEE 754 双精度64位浮点数](https://link.zhihu.com/?target=https://en.wikipedia.org/wiki/Floating_point%23Internal_representation" \t "https://zhuanlan.zhihu.com/p/_blank)（需翻墙）来存储的，它的表示格式为：(s) \* (m) \* (2 ^ e)

其中s表示符号位，m 表示尾数，占52位，e 表示指数，占11位，根据ECMAScript 5 规范，e 的范围是 [-1074, 971]，这样可以得出 js 能表示的最大值为1 \* (2^53 - 1) \* (2^971) = 1.7976931348623157e+308，而这个值恰好是 Number.MAX\_VALUE 的值；同理可以推出 js 能表示的大于 0 的最小值是1 \* 1 \* (2 ^ -1074) = 5e-324，这个值恰好是 Number.MIN\_VALUE 的值。

在计算机中，数字都是以二进制存储的，所以我们要先将 0.1 和 0.2 转化成二进制，对于十进制转二进制，整数部分除二取余，倒序排列，小数部分乘二取整，顺序排列，所以

0.1 转化为二进制  
 0.0 0011 0011 0011 0011 0011 0011 ... （0011循环）

0.2 转化为二进制  
 0.0011 0011 0011 0011 0011 0011 0011 ... （0011循环）

然后用[IEEE 754 双精度64位浮点数](https://link.zhihu.com/?target=https://en.wikipedia.org/wiki/Floating_point%23Internal_representation" \t "https://zhuanlan.zhihu.com/p/_blank)来表示：

0.1e = -4;m = 1.1001100110011001100110011001100110011001100110011010 (52位)

0.2e = -3;m = 1.1001100110011001100110011001100110011001100110011010 (52位)

这里的 m 指的是小数点后的52位，而小数点前的整数部分1是隐藏位。

然后把它相加，指数不一致时往右移，因为即使右边溢出了，损失的精度远远小于左移时的溢出。

e = -4; m = 1.1001100110011001100110011001100110011001100110011010 (52位)

+

e = -3; m = 1.1001100110011001100110011001100110011001100110011010 (52位)

转化

e = -3; m = 0.1100110011001100110011001100110011001100110011001101 (52位)

+

e = -3; m = 1.1001100110011001100110011001100110011001100110011010 (52位)

得到

e = -3; m = 10.0110011001100110011001100110011001100110011001100111 (52位)

保留一位整数

e = -2; m = 1.00110011001100110011001100110011001100110011001100111 (53位)

已经溢出来了（超过了52位），这个时候要做四舍五入，规则是保留偶数的那一个。

所以就是取m=1.0011001100110011001100110011001100110011001100110100 （52位）

然后得到最终的二进制数

1.0011001100110011001100110011001100110011001100110100 \* 2 ^ -2

=0.010011001100110011001100110011001100110011001100110100

现在转化为十进制，二进制小数转化为十进制的方法是小数点后第一位 \*2 ^ -1，第二位 \*2 ^ -2，以此类推，最终用等比数列的求和公式得到十进制数为 0.30000000000000004，所以0.1 + 0.2 的最终结果是

0.30000000000000004

**解决：**

1. 把0.1和0.2换成别的数字，比如先让0.1和0.2分别乘以10，求和之后再除以10 ，就不会有问题：(0.1\*10+0.2\*10)/10 ===0.3
2. 将得到的答案保留前十位：parseFloat((0.1+0.2).toFixed(10)) ===0.3