前段时间在做k的项目时，有一些客户端的计算，用js写的。发现了js精度的一些问题，借鉴某位大牛的一句话：“内事不决问百度，外事不决问google”的经典名言，上网淘了下相关资料，总结了下，以供参考：

1、js精度问题产生的根源：

这两个网址上洋洋洒洒说了一大堆，归根结底就是类型的精度问题导致，js的number类型按照ECMA的JavaScript标准，它的Number类型就是IEEE 754的双精度数值，相当于java的double类型。

详细的内容可以参考：

http://forum.eaw.com.cn/thread/10/1

http://blog.csdn.net/xiaocon/archive/2005/09/03/470709.aspx

2、js精度问题的解决方案：

1）由于js对于整数计算是没有误差的，可以先转化为整数进行计算

MathUtil.js

//浮点数相加

function dcmAdd(arg1,arg2){

    var r1,r2,m;

    try{r1=arg1.toString().split(”.”)[1].length;}catch(e){r1=0;}

    try{r2=arg2.toString().split(”.”)[1].length;}catch(e){r2=0;}

    m=Math.pow(10,Math.max(r1,r2));

    return (dcmMul(arg1,m)+dcmMul(arg2,m))/m;

}

//浮点数相减

function dcmSub(arg1,arg2){

     return dcmAdd(arg1,-arg2);

}

//浮点数相乘

function dcmMul(arg1,arg2){

    var m=0,s1=arg1.toString(),s2=arg2.toString();

    try{m+=s1.split(”.”)[1].length;}catch(e){}

    try{m+=s2.split(”.”)[1].length;}catch(e){}

    return Number(s1.replace(”.”,”"))\*Number(s2.replace(”.”,”"))/Math.pow(10,m);

}

//浮点数除

function dcmDiv(arg1,arg2){

    return dcmMul(arg1,1/arg2);

}

Number.prototype.add = function (arg){

                   return dcmAdd(this, arg);

}

Number.prototype.sub = function (arg){

                   return dcmSub(this, arg);

}

Number.prototype.mul = function (arg){

                   return dcmMul(this, arg);

}

Number.prototype.div = function (arg){

                   return dcmDiv(this, arg);

}

2）模拟服务端bigdecimal的实现写一个js客户端的bigdecimal算法

    自己写一套客户端的bigdecimal算法或者参考现成的bigdecimal实现

    参考网址：http://stz-ida.de/html/oss/js\_bigdecimal.html.en

3）利用ajax提交到后台进行计算

4）总结：

     1、尽量不要用JS做复杂的运算，特别是浮点数的运算。

     2、如果一定要进行浮点数的运算的话，先将浮点数转化为整形，再运算。