数据展示类

当你拿到 1.4000000000000001 这样的数据要展示时，建议使用 toPrecision 凑整并 parseFloat 转成数字后再显示，如下：

```
parseFloat(1.4000000000000001.toPrecision(12)) === 1.4  // True

```

封装成方法就是：

```
function strip(num, precision = 12) {
return +parseFloat(num.toPrecision(precision));
}
```

为什么选择 12 做为默认精度？这是一个经验的选择，一般选12就能解决掉大部分0001和0009问题，而且大部分情况下也够用了，如果你需要更精确可以调高。

数据运算类

对于运算类操作，如 +-*/，就不能使用 toPrecision 了。正确的做法是把小数转成整数后再运算。以加法为例：

```
/**
* 精确加法
*/
function add(num1, num2) {
const num1Digits = (num1.toString().split('.')[1] || '').length;
const num2Digits = (num2.toString().split('.')[1] || '').length;
const baseNum = Math.pow(10, Math.max(num1Digits, num2Digits));
return (num1 * baseNum + num2 * baseNum) / baseNum;
}
```

以上方法能适用于大部分场景。遇到科学计数法如 2.3e+1（当数字精度大于21时，数字会强制转为科学计数法形式显示）时还需要特别处理一下。

能读到这里，说明你非常有耐心，那我就放个福利吧。遇到浮点数误差问题时可以直接使用

https://github.com/dt-fe/number-precision