**JavaScript的高精度计算与JSON.parse的BIGINT**

在JavaScript处理整数的时候会遇到某些特别奇怪的问题,比如后台给你返回了一个超长的数字,然后js在计算的时候突然发现计算不对,不是后面为0就是计算得不到想要的结果.这里涉及到一个很简单的知识 也就是NUMBER的安全整数.

**Number安全整数**

Number.MAX\_SAFE\_INTEGER // 9007199254740991

9007199254740991+2 // 9007199254740992

复制代码

首先我们拿到了安全范围内的数字 然后对他进行简单的加法,正常的结果应该是9007199254740993 但是这里的尾数只有2, 大家可以去测试一下也是挺有意思的。因为js中所有的整数都是用浮点类型(double-precision 64-bit binary format IEEE 754 value)导致的,虽然可以承受的范围比较大,但是计算精度却不怎么好

**计算精度**

说完了安全整数这里来简单说说计算精度

0.1+0.2 //0.30000000000000004

复制代码

原理也是同上 那么解决办法也很简单

使用[math.js](http://mathjs.org/)库即可解决

// prevent round-off errors showing up **in** output

var ans = math.add(0.1, 0.2); // 0.30000000000000004

math.format(ans, {precision: 14}); // '0.3'

复制代码

**JSON.parse中遇到的BIGINT**

比较常见的场景为后台返回了一串JSON但是数字为BIGINT导致解析错误,一般为16位

var c='{"num": 90071992547409999}'

JSON.parse(c) // {num: 90071992547410000}

复制代码

比如我们使用jQuery的ajax开启了json:true功能 jQuery会自动parse 这样的话会造成精度不够的问题,所以这里可以使用json-bigint

var JSONbig = require('json-bigint');

var json = '{ "value" : 9223372036854775807, "v2": 123 }';

console.log('Input:', json);

console.log('');

console.log('node.js bult-in JSON:')

var r = JSON.parse(json);

console.log('JSON.parse(input).value : ', r.value.toString());

console.log('JSON.stringify(JSON.parse(input)):', JSON.stringify(r));

console.log('\n\nbig number JSON:');

var r1 = JSONbig.parse(json);

console.log('JSON.parse(input).value : ', r1.value.toString());

console.log('JSON.stringify(JSON.parse(input)):', JSONbig.stringify(r1));

// output

Input: { "value" : 9223372036854775807, "v2": 123 }

node.js bult-in JSON:

JSON.parse(input).value : 9223372036854776000

JSON.stringify(JSON.parse(input)): {"value":9223372036854776000,"v2":123}

big number JSON:

JSON.parse(input).value : 9223372036854775807

JSON.stringify(JSON.parse(input)): {"value":9223372036854775807,"v2":123}

复制代码

如果不想要引入额外的库则可以参考这段函数

let stringedJSON = origJSON.replace(/:\s\*([-+Ee0-9.]+)/g, ': "uniqueprefix$1"');

let o = JSON.parse(stringedJSON, (key, value) => {

// only changing strings

**if** (typeof value !== 'string') return value;

// only changing number strings

**if** (!value.startsWith('uniqueprefix')) return value;

// chop off the prefix

value = value.slice('uniqueprefix'.length);

// pick your favorite arbitrary-precision library

return new Big(value);

});

复制代码

参考文献

* [stackoverflow](https://stackoverflow.com/questions/47916160/alternative-to-json-parse-for-maintaining-decimal-precision)
* [json-bigint](https://github.com/sidorares/json-bigint#optionsstoreasstring-boolean-default-false)
* [ecma-262/5.1](http://www.ecma-international.org/ecma-262/5.1/)