链滴

解决 javascript 处理 Long 长度大于 17 位度丢失精度的问题

作者：cloudlang
原文链接：https://ld246.com/article/1578456167226
来源网站：链滴

```
<h3 id="解决问题的第一途径-最好从接口文档里面寻找方法-">解决问题的第一途径，最好从接口文
里面寻找方法。</h3>
<h4 id="JavaScript-数据类型和数据结构">JavaScript 数据类型和数据结构</h4>
<p></p><div class="vditor-linkcard vditor-tooltipped vditor-tooltipped__n" aria-label="https
//developer.mozilla.org/zh-CN/docs/Web/JavaScript/Data_structures">
    <a href="https://ld246.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2Fzh-
N%2Fdocs%2FWeb%2FJavaScript%2FData_structures" class="link-card fn__flex" target="_bla
k">
        <span class="vditor-linkcard__info">
            <span class="vditor-linkcard__title">
                <img height="20px" src="https://developer.mozilla.org/favicon-48x48.97046865.p
g">
                JavaScript 数据类型和数据结构 - JavaScript | MDN
            </span>
            <span class="vditor-linkcard__abstract">编程语言都具有内建的数据结构，但各种编程语
的数据结构常有不同之处。本文试图列出 JavaScript 语言中内建的数据结构及其属性，它们可以用来
建其他的数据结构；同时尽可能地描述与其他语言的不同之处。</span>
            <span class="vditor-linkcard__site">
            developer.mozilla.org
            </span>
        </span>
        <span class="vditor-linkcard__image" data-src="https://developer.mozilla.org/mdn-soci
l-share.0ca9dbda.png"></span>
    </a>
</div><p></p>
<h4 id="直接引用原文-">直接引用原文：</h4>
<hr>
<h4 id="数据类型">数据类型</h4>
<p>最新的 ECMAScript 标准定义了 8 种数据类型:</p>
<ul>
<li>7 种<a href="https://ld246.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%
Fen-US%2Fdocs%2FGlossary%2FPrimitive" title="原始类型: In JavaScript, a primitive (primitive
value, primitive data type) is data that is not an object and has no methods. There are 7 primit
ve data types: string, number, bigint, boolean, null, undefined, and symbol." target="_blank" r
l="nofollow ugc">原始类型</a>:
<ul>
<li><a href="https://ld246.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2Fe
-US%2Fdocs%2FGlossary%2FBoolean" title="Boolean: In computer science, a Boolean is a log
cal data type that can have only the values true or false." target="_blank" rel="nofollow ugc"
Boolean</a></li>
<li><a href="https://ld246.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2Fe
-US%2Fdocs%2FGlossary%2FNull" title="Null: In computer science, a null value represents a r
ference that points, generally intentionally, to a nonexistent or invalid object or address. The
eaning of a null reference varies among language implementations." target="_blank" rel="no
ollow ugc">Null</a></li>
<li><a href="https://ld246.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2Fe
-US%2Fdocs%2FGlossary%2FUndefined" title="Undefined: undefined is a primitive value aut
matically assigned to variables that have just been declared, or to formal arguments for which
there are no actual arguments." target="_blank" rel="nofollow ugc">Undefined</a></li>
<li><a href="https://ld246.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2Fe
-US%2Fdocs%2FGlossary%2FNumber" title="Number: In JavaScript, Number is a numeric dat
 type in the double-precision 64-bit floating point format (IEEE 754). In other programming la
guages different numeric types can exist, for examples: Integers, Floats, Doubles, or Bignums."
target="_blank" rel="nofollow ugc">Number</a></li>
```

```
<li><a href="https://ld246.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2Fe
-US%2Fdocs%2FGlossary%2FBigInt" title="BigInt: In JavaScript, BigInt is a numeric data type
hat can represent integers in the arbitrary precision format. In other programming languages
different numeric types can exist, for examples: Integers, Floats, Doubles, or Bignums." target
"_blank" rel="nofollow ugc">BigInt</a></li>
<li><a href="https://ld246.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2Fe
-US%2Fdocs%2FGlossary%2FString" title="String: In any computer programming language, a
string is a sequence of characters used to represent text." target="_blank" rel="nofollow ugc"
String</a></li>
<li><a href="https://ld246.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2Fe
-US%2Fdocs%2FGlossary%2FSymbol" title="Symbol: In JavaScript, Symbol is a primitive value
" target="_blank" rel="nofollow ugc">Symbol</a></li>
</ul>
</li>
<li>和 <a href="https://ld246.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2
en-US%2Fdocs%2FGlossary%2FObject" title="Object: Object refers to a data structure contain
ng data and instructions for working with the data. Objects sometimes refer to real-world thi
gs, for example a car or map object in a racing game. JavaScript, Java, C++, Python, and Ruby
are examples of object-oriented programming languages." target="_blank" rel="nofollow ugc
>Object</a></li>
</ul>
<h4 id="Number-数字类型">Number 数字类型</h4>
<p>根据 ECMAScript 标准，JavaScript 中只有一种数字类型：基于 IEEE 754 标准的双精度 64 位
进制格式的值（-(2^53 -1) 到 (2^53 -1)）。<strong>它并没有为整数给出一种特定的类型</strong
。除了能够表示浮点数外，还有一些带符号的值：<code>+Infinity</code>，<code>-Infinity</c
de> 和 <code>NaN</code> (非数值，Not-a-Number)。</p>
<p>要检查值是否大于或小于 <code>+/-Infinity</code>，你可以使用常量 <a href="https://ld2
6.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2Fzh-CN%2Fdocs%2FWeb%2
JavaScript%2FReference%2FGlobal_Objects%2FNumber%2FMAX_VALUE" title="Number.MAX
VALUE 属性表示在 JavaScript 里所能表示的最大数值。" target="_blank" rel="nofollow ugc"><c
de>Number.MAX_VALUE</code></a> 和 <a href="https://ld246.com/forward?goto=https%
A%2F%2Fdeveloper.mozilla.org%2Fzh-CN%2Fdocs%2FWeb%2FJavaScript%2FReference%2FG
obal_Objects%2FNumber%2FMIN_VALUE" title="Number.MIN_VALUE 属性表示在 JavaScript
所能表示的最小的正值。" target="_blank" rel="nofollow ugc"><code>Number.MIN_VALUE</
ode></a>。另外在 ECMAScript 6 中，你也可以通过 <a href="https://ld246.com/forward?goto
https%3A%2F%2Fdeveloper.mozilla.org%2Fzh-CN%2Fdocs%2FWeb%2FJavaScript%2FRefere
ce%2FGlobal_Objects%2FNumber%2FisSafeInteger" title="Number.isSafeInteger() 方法用来
断传入的参数值是否是一个“安全整数” （safe integer）。" target="_blank" rel="nofollow ugc"
<code>Number.isSafeInteger()</code></a> 方法还有 <a href="https://ld246.com/forward?g
to=https%3A%2F%2Fdeveloper.mozilla.org%2Fzh-CN%2Fdocs%2FWeb%2FJavaScript%2FRef
rence%2FGlobal_Objects%2FNumber%2FMAX_SAFE_INTEGER" title="Number.MAX_SAFE_INT
GER 常量表示在 JavaScript 中最大的安全整数（maxinum safe integer)（253 - 1）。" target="_bl
nk" rel="nofollow ugc"><code>Number.MAX_SAFE_INTEGER</code></a> 和 <a href="https
//ld246.com/forward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2Fzh-CN%2Fdocs%2FW
b%2FJavaScript%2FReference%2FGlobal_Objects%2FNumber%2FMIN_SAFE_INTEGER" title=
Number.MIN_SAFE_INTEGER 代表在 JavaScript中最小的安全的integer型数字 (-(253 - 1))." targe
="_blank" rel="nofollow ugc"><code>Number.MIN_SAFE_INTEGER</code></a> 来检查值是
在双精度浮点数的取值范围内。 超出这个范围，JavaScript 中的数字不再安全了，也就是只有 second
mathematical interger 可以在 JavaScript 数字类型中正确表现。</p>
<p>数字类型中只有一个整数有两种表示方法： 0 可表示为 -0 和 +0（"0" 是 +0 的简写）。 在实
中，这也几乎没有影响。 例如 <code>+0 === -0</code> 为真。 但是，你可能要注意除以 0 的
候：</p>
<pre><code class="language-js highlight-chroma"><span class="highlight-line"><span clas
="highlight-cl"><span class="highlight-mi">42</span> <span class="highlight-o">/</span
```

```
 <span class="highlight-o">+</span><span class="highlight-mi">0</span><span class="hi
hlight-p">;</span> <span class="highlight-c1">// Infinity
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span><span class="highlight-mi">42</span> <span class="highlight-o
>/</span> <span class="highlight-o">-</span><span class="highlight-mi">0</span><spa
 class="highlight-p">;</span> <span class="highlight-c1">// -Infinity
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>
</span></span></code></pre>
<p>尽管一个数字常常仅代表它本身的值，但 JavaScript 提供了一些<a href="https://ld246.com/f
rward?goto=https%3A%2F%2Fdeveloper.mozilla.org%2Fen-US%2Fdocs%2FWeb%2FJavaScri
t%2FReference%2FOperators%2FBitwise_Operators" title="en/JavaScript/Reference/Operator
/Bitwise_Operators" target="_blank" rel="nofollow ugc">位运算符</a>。 这些位运算符和一个
一数字通过位操作可以用来表现一些布尔值。然而自从 JavaScript 提供其他的方式来表示一组布尔值
如一个布尔值数组或一个布尔值分配给命名属性的对象）后，这种方式通常被认为是不好的。位操作
容易使代码难以阅读，理解和维护， 在一些非常受限的情况下，可能需要用到这些技术，比如试图应
本地存储的存储限制。 位操作只应该是用来优化尺寸的最后选择。</p>
<h3 id="javascript最大的安全数为17位--Math-pow-2--53----1-即-9007199254740991-">javasc
ipt 最大的安全数为 17 位， Math.pow(2, 53) - 1，即 9007199254740991。</h3>
<p>根据双精度浮点数的构成，精度位数是 53 bit。安全数的意思是在 -2^53 ~ 2^53 内的整数(不
括边界)与唯一的双精度浮点数互相对应。举个例子比较好理解：Math.pow(2, 53) === Math.pow(2,
53) + 1 // trueMath.pow(2, 53) 竟然与 Math.pow(2, 53) + 1 相等！这是因为 Math.pow(2, 53) +
 已经超过了尾数的精度限制(53 bit)，在这个例子中 Math.pow(2, 53) 和 Math.pow(2, 53) + 1 对应
同一个双精度浮点数。所以 Math.pow(2, 53) 就不是安全数了。最大的安全数为 Math.pow(2, 53) -
，即 9007199254740991。</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">作者：牧云云
</span></span><span class="highlight-line"><span class="highlight-cl">链接：https://www
zhihu.com/question/29010688/answer/503100778
</span></span><span class="highlight-line"><span class="highlight-cl">来源：知乎
</span></span><span class="highlight-line"><span class="highlight-cl">著作权归作者所有
商业转载请联系作者获得授权，非商业转载请注明出处。
</span></span></code></pre>
<h3 id="javascript处理Long长度大于17位度丢失的问题">javascript 处理 Long 长度大于 17 位度
失的问题</h3>
<blockquote>
<p>对于 Long 类型的数据，如果我们在 Controller 层将结果序列化为 json，直接传给前端的话，在
Long 长度大于 17 位时会出现精度丢失的问题。如何避免精度丢失呢？最常用的办法就是将 Long
型字段统一转成 String 类型。</p>
</blockquote>
<p>参考：<br>
1.<a href="https://ld246.com/forward?goto=https%3A%2F%2Fblog.csdn.net%2Fu01002886
%2Farticle%2Fdetails%2F86563382" target="_blank" rel="nofollow ugc">彻底解决 JS 处理 Lo
g 类型精度丢失问题（一）</a><br>
2.<a href="https://ld246.com/forward?goto=https%3A%2F%2Fblog.csdn.net%2Fu01002886
%2Farticle%2Fdetails%2F86563729" target="_blank" rel="nofollow ugc">彻底解决 JS 处理 Lo
g 类型精度丢失问题（二）</a></p>
```