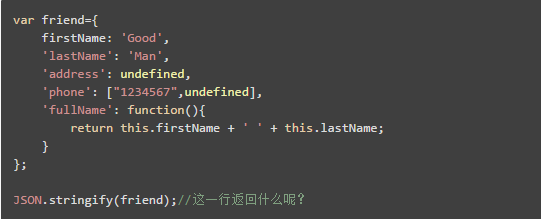
由里到外步步深挖，深入理解 JSON

我们先来看一个JS中常见的JS对象序列化成JSON字符串的问题。

请问：以下JS对象通过JSON.stringify后的字符串是怎样的？



第二个问题，如果我想在最终JSON字符串将这个'friend'的姓名全部变成大写字母，也就是把"Good"变成"GOOD"，把"Man"变成"MAN"，那么可以怎么做？

基于以上两个问题，我们再追本溯源问一下：

JSON究竟是什么东西？

为什么JSON就是易于数据交换？

JSON和JS对象的区别？

JS中JSON.parse、JSON.stringify和不常见的toJSON，这几个函数的参数和处理细节到底是怎样的？

欢迎进入本次“深挖JSON之旅”，下文将从以下几个方面去理解JSON：

首先是对“JSON是一种轻量的数据交换格式”的理解；

然后来看经常被混为一谈的JSON和JS对象的区别；

最后我们再来看JS中这几个JSON相关函数具体的执行细节。

希望全文能让如之前的我一样对JSON一知半解的亲能说清楚JSON是什么，也能熟练运用JSON，不看控制台就知道JS对象序列化成JSON字符串后输出是啥。

JSON 是一种格式

  - 基于文本，优于轻量，用于交换数据

如果没有去过JSON的官方介绍可以去一下这里，官方介绍第一、二段已经很清楚地表述了JSON是什么，我将JSON是什么提炼成以下几个方面：

1

一种数据格式

什么是格式？就是规范你的数据要怎么表示，举个栗子，有个人叫“二百六”，身高“160cm”，体重“60kg”，现在你要将这个人的这些信息传给别人或者别的什么东西，你有很多种选择：

姓名“二百六”，身高“160cm”，体重“60kg”

name="二百六"&height="160cm"&weight="60kg"

<person><name>二百六</name><height>160</height><weight>60</weight></person>

{"name":"二百六","height":160,"weight":60}

... ...

以上所有选择，传递的数据是一样的，但是你可以看到形式是可以各式各样的，这就是各种不同格式化后的数据，JSON是其中一种表示方式。

2

基于文本的数据格式

JSON是基于文本的数据格式，相对于基于二进制的数据，所以JSON在传递的时候是传递符合JSON这种格式(至于JSON的格式是什么我们第二部分再说)的字符串，我们常会称为“JSON字符串”。

3

轻量级的数据格式

在JSON之前，有一个数据格式叫xml，现在还是广泛在用，但是JSON更加轻量，如xml需要用到很多标签。

像上面的例子中，你可以明显看到xml格式的数据中标签本身占据了很多空间，而JSON比较轻量，即相同数据，以JSON的格式占据的带宽更小，这在有大量数据请求和传递的情况下是有明显优势的。

4

被广泛地用于数据交换

轻量已经是一个用于数据交换的优势了，但更重要的JSON是易于阅读、编写和机器解析的，即这个JSON对人和机器都是友好的，而且又轻，独立于语言（因为是基于文本的），所以JSON被广泛用于数据交换。

以前端JS进行ajax的POST请求为例，后端PHP处理请求为例：

前端构造一个JS对象，用于包装要传递的数据，然后将JS对象转化为JSON字符串，再发送请求到后端；

后端PHP接收到这个JSON字符串，将JSON字符串转化为PHP对象，然后处理请求。

前端构造一个JS对象，用于包装要传递的数据，然后将JS对象转化为JSON字符串，再发送请求到后端；

后端PHP接收到这个JSON字符串，将JSON字符串转化为PHP对象，然后处理请求。

可以看到，相同的数据在这里有3种不同的表现形式，分别是前端的JS对象、传输的JSON字符串、后端的PHP对象。

JS对象和PHP对象明显不是一个东西，但是由于大家用的都是JSON来传递数据，大家都能理解这种数据格式，都能把JSON这种数据格式很容易地转化为自己能理解的数据结构。

这就方便啦，在其他各种语言环境中交换数据都是如此。

JSON 和 JS 对象之间的“八卦”

很多时候都听到“JSON是JS的一个子集”这句话，而且这句话我曾经也一直这么认为，每个符合JSON格式的字符串你解析成js都是可以的，直到后来发现了一个奇奇怪怪的东西...

1

两个本质不同的东西为什么那么密切

JSON和JS对象本质上完全不是同一个东西，就像“斑马线”和“斑马”，“斑马线”基于“斑马”身上的条纹来呈现和命名，但是斑马是活的，斑马线是非生物。

同样，"JSON"全名"JavaScript Object Notation"，所以它的格式（语法）是基于JS的，但它就是一种格式，而JS对象是一个实例，是存在于内存的一个东西。

说句玩笑话，如果JSON是基于PHP的，可能就叫PON了，形式可能就是这样的了['propertyOne' => 'foo', 'propertyTwo' => 42,]，如果这样，那么JSON可能现在是和PHP比较密切了。

此外，JSON是可以传输的，因为它是文本格式，但是JS对象是没办法传输的，在语法上，JSON也会更加严格，但是JS对象就很松了。

那么两个不同的东西为什么那么密切，因为JSON毕竟是从JS中演变出来的，语法相近。

2

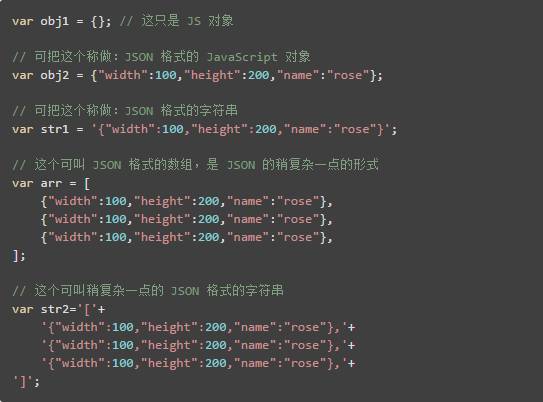
JSON 格式比 JS 对象语法严格的表现

先就以“键值对为表现的对象”形式上，对比下两者的不同，至于JSON还能以怎样的形式表现，对比完后再罗列。



可以看到，相对于JS对象，JSON的格式更严格，所以大部分写的JS对象是不符合JSON的格式的。

以下代码引用请查看（阅读原文）

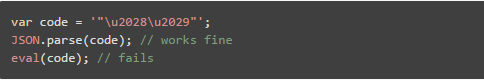


完整的JSON语法参考（阅读原文）

3

JSON 不是 JS 的子集

首先看下面的代码，你可以copy到控制台执行下：



这两个字符\u2028和\u2029分别表示行分隔符和段落分隔符，JSON.parse可以正常解析，但是当做js解析时会报错。

这几个 JS 中的 JSON 函数

在JS中我们主要会接触到两个和JSON相关的函数，分别用于JSON字符串和JS数据结构之间的转化。

一个叫JSON.stringify，它很聪明，聪明到你写的不符合JSON格式的JS对象都能帮你处理成符合JSON格式的字符串，所以你得知道它到底干了什么，免得它只是自作聪明，然后让你Debug long time。

另一个叫JSON.parse，用于转化json字符串到JS数据结构，它很严格，你的JSON字符串如果构造地不对，是没办法解析的。

而它们的参数不止一个，虽然我们经常用的时候只传入一个参数。

此外，还有一个toJSON函数，我们较少看到，但是它会影响JSON.stringify。

1

JS 数据结构转化为 JSON 字符串

将JS数据结构转化为JSON字符串 —— JSON.stringify

这个函数的函数签名是这样的：

C:\Users\ADMINI~1\AppData\Local\Temp\WeChat Files\02c2e3324c90d838a7fe7190551b1e2c.png

下面将分别展开带1~3个参数的用法，最后是它在序列化时做的一些“聪明”的事，要特别注意。

1.1 基本使用 —— 仅需一个参数

这个大家都会使用，传入一个JSON格式的JS对象或者数组，JSON.stringify({"name":"Good Man","age":18})返回一个字符串"{"name":"Good Man","age":18}"。

可以看到本身我们传入的这个JS对象就是符合JSON格式的，用的双引号，也没有JSON不接受的属性值，那么如果像开头那个例子中的一样，how to play？

不急，我们先举简单的例子来说明这个函数的几个参数的意义，再来说这个问题。

1.2 第二个参数可以是函数，也可以是一个数组

如果第二个参数是一个函数，那么序列化过程中的每个属性都会被这个函数转化和处理

如果第二个参数是一个数组，那么只有包含在这个数组中的属性才会被序列化到最终的JSON字符串中

如果第二个参数是null，那作用上和空着没啥区别，但是不想设置第二个参数，只是想设置第三个参数的时候，就可以设置第二个参数为null

这第二个参数若是函数



如果制定了第二个参数是函数，那么这个函数必须对每一项都有返回，这个函数接受两个参数，一个键名，一个是属性值，函数必须针对每一个原来的属性值都要有新属性值的返回。

那么问题来了，如果传入的不是键值对的对象形式，而是方括号的数组形式呢？，比如上面的friend变成这样：friend=["Jack","Rose"]，那么这个逐属性处理的函数接收到的key和value又是什么？

如果是数组形式，那么key是索引值，而value是这个数组项，你可以在控制台在这个函数内部打印出来这个key和value验证，记得要在函数内部返回value，不然会出错。

这第二个参数若是数组



如果第二个参数是一个数组，那么只有在数组中出现的属性才会被序列化进结果字符串，只要在这个提供的数组中找不到的属性就不会被包含进去，而这个数组中存在但是源JS对象中不存在的属性会被忽略，不会报错。

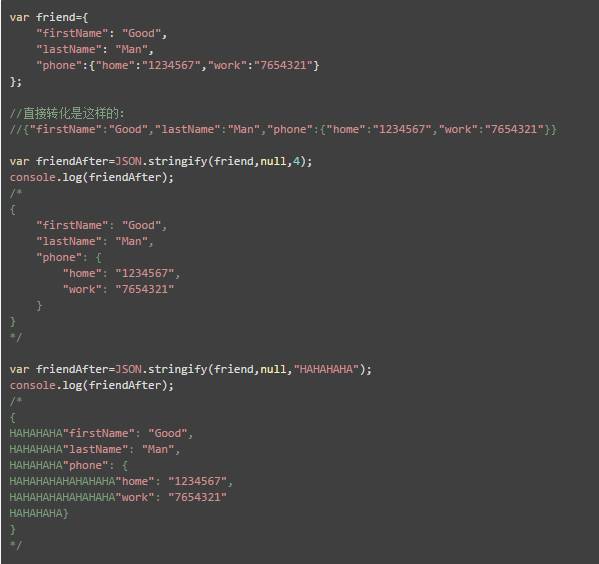
1.3 第三个参数用于美化输出 —— 不建议用

指定缩进用的空白字符，可以取以下几个值：

是1-10的某个数字，代表用几个空白字符

是字符串的话，就用该字符串代替空格，最多取这个字符串的前10个字符

没有提供该参数 等于 设置成null 等于 设置一个小于1的数



笑笑就好，别这样用，序列化是为了传输，传输就是能越小越好，加莫名其妙的缩进符，解析困难（如果是字符串的话），也弱化了轻量化这个特点。。

1.4 注意这个函数的“小聪明”（重要）

如果有其他不确定的情况，那么最好的办法就是"Have a try"，控制台做下实验就明了。

键名不是双引号的（包括没有引号或者是单引号），会自动变成双引号；字符串是单引号的，会自动变成双引号

最后一个属性后面有逗号的，会被自动去掉

非数组对象的属性不能保证以特定的顺序出现在序列化后的字符串中

这个好理解，也就是对非数组对象在最终字符串中不保证属性顺序和原来一致

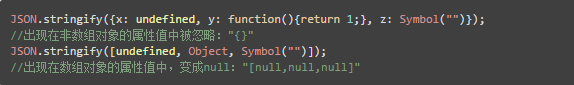
布尔值、数字、字符串的包装对象在序列化过程中会自动转换成对应的原始值

也就是你的什么new String("bala")会变成"bala"，new Number(2017)会变成2017

undefined、任意的函数（其实有个函数会发生神奇的事，后面会说）以及 symbol 值（symbol详见ES6对symbol的介绍）

出现在非数组对象的属性值中：在序列化过程中会被忽略

出现在数组中时：被转换成 null



NaN、Infinity和-Infinity，不论在数组还是非数组的对象中，都被转化为null

所有以 symbol 为属性键的属性都会被完全忽略掉，即便 replacer 参数中强制指定包含了它们

不可枚举的属性会被忽略

2

JSON 字符串解析为 JS 数据结构

将JSON字符串解析为JS数据结构 —— JSON.parse

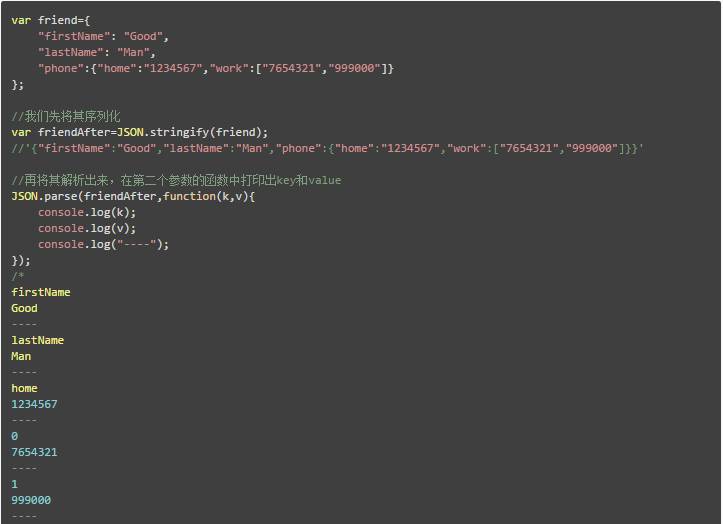
这个函数的函数签名是这样的：

C:\Users\ADMINI~1\AppData\Local\Temp\WeChat Files\58b030141d7b45d5023a7690cdc7d439.png

如果第一个参数，即JSON字符串不是合法的字符串的话，那么这个函数会抛出错误，所以如果你在写一个后端返回JSON字符串的脚本，最好调用语言本身的JSON字符串相关序列化函数。

而如果是自己去拼接实现的序列化字符串，那么就尤其要注意序列化后的字符串是否是合法的，合法指这个JSON字符串完全符合JSON要求的严格格式。

值得注意的是这里有一个可选的第二个参数，这个参数必须是一个函数，这个函数作用在属性已经被解析但是还没返回前，将属性处理后再返回。



仔细看一下这些输出，可以发现这个遍历是由内而外的，可能由内而外这个词大家会误解，最里层是内部数组里的两个值啊，但是输出是从第一个属性开始的，怎么就是由内而外的呢？

这个由内而外指的是对于复合属性来说的，通俗地讲，遍历的时候，从头到尾进行遍历，如果是简单属性值（数值、字符串、布尔值和null），那么直接遍历完成。

如果是遇到属性值是对象或者数组形式的，那么暂停，先遍历这个子JSON，而遍历的原则也是一样的，等这个复合属性遍历完成，那么再完成对这个属性的遍历返回。

本质上，这就是一个深度优先的遍历。

有两点需要注意：

如果 reviver 返回 undefined，则当前属性会从所属对象中删除，如果返回了其他值，则返回的值会成为当前属性新的属性值。

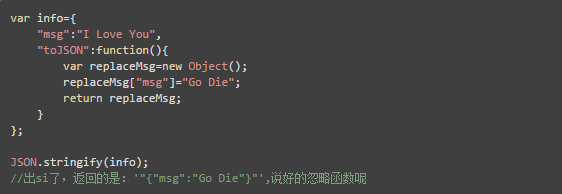
你可以注意到上面例子最后一组输出看上去没有key，其实这个key是一个空字符串，而最后的object是最后解析完成对象，因为到了最上层，已经没有真正的属性了。

3

影响 JSON.stringify 的神奇函数

影响 JSON.stringify 的神奇函数 —— object.toJSON

如果你在一个JS对象上实现了toJSON方法，那么调用JSON.stringify去序列化这个JS对象时，JSON.stringify会把这个对象的toJSON方法返回的值作为参数去进行序列化。



这个函数就是这样子的。

其实Date类型可以直接传给JSON.stringify做参数，其中的道理就是，Date类型内置了toJSON方法。

小结以及关于兼容性的问题

到这里终于把，JSON和JS中的JSON，梳理了一遍，也对里面的细节和注意点进行了一次遍历，知道JSON是一种语法上衍生于JS语言的一种轻量级的数据交换格式，也明白了JSON相对于一般的JS数据结构（尤其是对象）的差别，更进一步，仔细地讨论了JS中关于JSON处理的3个函数和细节。

不过遗憾的是，以上所用的3个函数，不兼容IE7以及IE7之前的浏览器。有关兼容性的讨论，留待之后吧。如果想直接在应用上解决兼容性，那么可以套用JSON官方的js，可以解决。