JSON: **J**ava**S**cript **O**bject **N**otation(JavaScript 对象表示法)

JSON 是存储和交换文本信息的语法。类似 XML。

JSON 比 XML 更小、更快，更易解析。

* JSON 独立于语言：JSON 使用 Javascript语法来描述数据对象，但是 JSON 仍然独立于语言和平台。JSON 解析器和 JSON 库支持许多不同的编程语言。 目前非常多的动态（PHP，JSP，.NET）编程语言都支持JSON。

## JSON - 转换为 JavaScript 对象

JSON 文本格式在语法上与创建 JavaScript 对象的代码相同。

由于这种相似性，无需解析器，JavaScript 程序能够使用内建的 eval() 函数，用 JSON 数据来生成原生的 JavaScript 对象。

## 与 XML 相同之处

* JSON 是纯文本
* JSON 具有"自我描述性"（人类可读）
* JSON 具有层级结构（值中存在值）
* JSON 可通过 JavaScript 进行解析
* JSON 数据可使用 AJAX 进行传输

## 与 XML 不同之处

* 没有结束标签
* 更短
* 读写的速度更快
* 能够使用内建的 JavaScript eval() 方法进行解析
* 使用数组
* 不使用保留字

## 为什么使用 JSON？

对于 AJAX 应用程序来说，JSON 比 XML 更快更易使用：

#### 使用 XML

* 读取 XML 文档
* 使用 XML DOM 来循环遍历文档
* 读取值并存储在变量中

#### 使用 JSON

* 读取 JSON 字符串
* 用 eval() 处理 JSON 字符串

## JSON 语法规则

JSON 语法是 JavaScript 对象表示语法的子集。

* 数据在名称/值对中
* 数据由逗号分隔
* 大括号保存对象
* 中括号保存数组

## JSON 名称/值对

JSON 数据的书写格式是：名称/值对。

名称/值对包括字段名称（在双引号中），后面写一个冒号，然后是值：

"name" : "菜鸟教程"

这很容易理解，等价于这条 JavaScript 语句：

name = "菜鸟教程"

## JSON 值

JSON 值可以是：

* 数字（整数或浮点数）
* 字符串（在双引号中）
* 逻辑值（true 或 false）
* 数组（在中括号中）
* 对象（在大括号中）
* null

## JSON 数组

JSON 数组在中括号中书写：

数组可包含多个对象：

{ "sites": [ { "name":"菜鸟教程" , "url":"www.runoob.com" }, { "name":"google" , "url":"www.google.com" }, { "name":"微博" , "url":"www.weibo.com" } ] }

在上面的例子中，对象 "sites" 是包含三个对象的数组。每个对象代表一条关于某个网站（name、url）的记录。

## JSON 布尔值

JSON 布尔值可以是 true 或者 false：

{ "flag":true }

## JSON null

JSON 可以设置 null 值：

{ "runoob":null }

## JSON 使用 JavaScript 语法

因为 JSON 使用 JavaScript 语法，所以无需额外的软件就能处理 JavaScript 中的 JSON。

通过 JavaScript，您可以创建一个对象数组，并像这样进行赋值：

## 实例

var sites = [ { "name":"runoob" , "url":"www.runoob.com" }, { "name":"google" , "url":"www.google.com" }, { "name":"微博" , "url":"www.weibo.com" } ];

可以像这样访问 JavaScript 对象数组中的第一项（索引从 0 开始）：

sites[0].name;

## 对象语法

## 实例

{ "name":"runoob", "alexa":10000, "site":null }

JSON 对象使用在大括号({})中书写。

对象可以包含多个 **key/value（键/值）**对。

key 必须是字符串，value 可以是合法的 JSON 数据类型（字符串, 数字, 对象, 数组, 布尔值或 null）。

key 和 value 中使用冒号(:)分割。

每个 key/value 对使用逗号(,)分割。

你也可以使用中括号（[]）来访问对象的值：

## 实例

var myObj, x; myObj = { "name":"runoob", "alexa":10000, "site":null }; x = myObj["name"];

## 嵌套 JSON 对象

JSON 对象中可以包含另外一个 JSON 对象：

## 实例

myObj = { "name":"runoob", "alexa":10000, "sites": { "site1":"www.runoob.com", "site2":"m.runoob.com", "site3":"c.runoob.com" } }

你可以使用点号(.)或者中括号([])来访问嵌套的 JSON 对象。

## 实例

x = myObj.sites.site1; // 或者 x = myObj.sites["site1"];

## 删除对象属性

我们可以使用 **delete** 关键字来删除 JSON 对象的属性：

## 实例

delete myObj.sites.site1;

**[尝试一下 »](https://www.runoob.com/try/try.php?filename=tryjson_object_delete" \t "https://www.runoob.com/json/_blank)**

你可以使用中括号([])来删除 JSON 对象的属性：

## 实例

delete myObj.sites["site1"]

删除数值的元素，数组的大小不变，代码如下：

delete 运算符并不是彻底删除元素，而是删除它的值，但仍会保留空间。

运算符 delete 只是将该值置为 undefined，而不会影响数组长度，即将其变为稀疏数组（《JS权威指南》7.5节）。

# JSON.parse()

JSON 通常用于与服务端交换数据。

在接收服务器数据时一般是字符串。

我们可以使用 JSON.parse() 方法将数据转换为 JavaScript 对象。

### **语法**

JS**参数说明：**

* **text:**必需， 一个有效的 JSON 字符串。
* **reviver:** 可选，一个转换结果的函数， 将为对象的每个成员调用此函数。

ON.parse(text[, reviver])

## JSON 解析实例

例如我们从服务器接收了以下数据：

{ "name":"runoob", "alexa":10000, "site":"www.runoob.com" }

我们使用 JSON.parse() 方法处理以上数据，将其转换为 JavaScript 对象：

var obj = JSON.parse('{ "name":"runoob", "alexa":10000, "site":"www.runoob.com" }');

## 从服务端接收 JSON 数据

我们可以使用 AJAX 从服务器请求 JSON 数据，并解析为 JavaScript 对象。

## 异常

### **解析数据**

JSON 不能存储 Date 对象。

如果你需要存储 Date 对象，需要将其转换为字符串。

之后再将字符串转换为 Date 对象。

JSON 不允许包含函数，但你可以将函数作为字符串存储，之后再将字符串转换为函数。

对于服务器返回的JSON字符串，如果 jQuery 异步请求没做类型说明，或者以字符串方式接受，那么需要做一次对象化处理，方式不是太麻烦，就是将该字符串放于 eval()中执行一次。这种方式也适合以普通 javascipt 方式获取 json 对象，以下举例说明：

var u = eval('('+user+')');

为什么要 eval 这里要添加 **('('+user+')')** 呢？

原因在于：eval 本身的问题。 由于 json 是以 **{}** 的方式来开始以及结束的，在 js 中，它会被当成一个语句块来处理，所以必须强制性的将它转换成一种表达式。

加上圆括号的目的是迫使 eval 函数在处理 JavaScript 代码的时候强制将括号内的表达式（expression）转化为对象，而不是作为语句（statement）来执行。举一个例子，例如对象字面量 {}，如若不加外层的括号，那么 eval 会将大括号识别为 javascript 代码块的开始和结束标记，那么{}将会被认为是执行了一句空语句。所以下面两个执行结果是不同的：

alert(eval("{}"); // return undefined

alert(eval("({})");// return object[Object]

## 解析函数

JSON 不允许包含函数，JSON.stringify() 会删除 JavaScript 对象的函数，包括 key 和 value。

我们可以在执行 JSON.stringify() 函数前将函数转换为字符串来避免以上问题的发生：

## 把 JSON 文本转换为 JavaScript 对象

JSON 最常见的用法之一，是从 web 服务器上读取 JSON 数据（作为文件或作为 HttpRequest），将 JSON 数据转换为 JavaScript 对象，然后在网页中使用该数据。

为了更简单地为您讲解，我们使用字符串作为输入进行演示（而不是文件）。

eval() 函数可编译并执行任何 JavaScript 代码。这隐藏了一个潜在的安全问题。

使用 JSON 解析器将 JSON 转换为 JavaScript 对象是更安全的做法。JSON 解析器只能识别 JSON 文本，而不会编译脚本。

在浏览器中，这提供了原生的 JSON 支持，而且 JSON 解析器的速度更快。

较新的浏览器和最新的 ECMAScript (JavaScript) 标准中均包含了原生的对 JSON 的支持。

Jsonp(JSON with Padding) 是 json 的一种"使用模式"，可以让网页从别的域名（网站）那获取资料，即跨域读取数据。

为什么我们从不同的域（网站）访问数据需要一个特殊的技术(JSONP )呢？这是因为同源策略。

同源策略，它是由Netscape提出的一个著名的安全策略，现在所有支持JavaScript 的浏览器都会使用这个策略。

## JSONP 应用

### **1. 服务端JSONP格式数据**

如客户想访问 : http://www.runoob.com/try/ajax/jsonp.php?jsoncallback=callbackFunction。

假设客户期望返回JSON数据：["customername1","customername2"]。

真正返回到客户端的数据显示为: callbackFunction(["customername1","customername2"])。