JSON 介绍

Submitted by **xzy** on 2008, May 15, 6:05 PM. AJAX 技术

介绍

我们知道 AJAX 技术能够使得每一次请求更加迅捷，对于每一次请求返回的不是整个页面，也仅仅是所需要返回的数据。通常 AJAX 通过返回 XML 格式的数据，然后再通过客户端复杂的 JavaScript 脚本解析和渲染这些 XML 格式的数据。
JSON（读 Jason）是为了能够使得数据格式成为一种标准，更简单的被 JavaScript 解析。

优点

1、轻量级的数据交换格式
2、人们读写更加容易
3、易于机器的解析和生成
4、能够通过 JavaScript 中 eval()函数解析 JSON
5、JSON 支持多语言。包括：ActionScript, C, C#, ColdFusion, E, Java, JavaScript, ML, Objective CAML, Perl, PHP, Python, Rebol, Ruby, and Lua.

语法

JSON 语法是一种用于传输和生成数据的协定，很类似于 C 家族的语言，所以很容易被 C 家族的语言所解析。
对象：对象包含再{}之间
属性：采用 Key-Value 对来表示。属性之间使用逗号分开。 ***string*** : ***value***
数组：数组存放再[]之间 [***elements***]
元素：元素之间用逗号分开
值：值可以是字符串，数字，对象，数组，true，false，null

例子：
JSON

```
{"menu": {
  "id": "file",
  "value": "File:",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"} ]
  }
}}
```

JSON 详细教程 2008-12-30 11:15

这是个人在最近使用 json 时做的总结，拿出来给没接触过的小弟们晒晒，适用与没接触过 json 的人员，其中 json2。js 请到 json 官网下载。

```
<script type="text/javascript" src="json2.js"></script>
<script>
//直接声明 json 数据结构
var myJSONObject = {"bindings": [
                {"ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.*"},
                {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex": "^delete.*"},
                {"ircEvent": "PRIVMSG", "method": "randomURI", "regex": "^random.*"}
        ]
};
//声明字符串,可对比一下 json 文本与我们正常文本的区别
var         normalstring='[{persons:[{name:"jordan",sex:"m",age:"40"},
{name:"bryant",sex:"m",age:"28"},
{name:"McGrady",sex:"m",age:"27"} ]}]';
var     jsontext='[{"persons":[{"name":"jordan","sex":"m","age":"40"},
{"name":"bryant","sex":"m","age":"28"},
{"name":"McGrady","sex":"m","age":"27"} ]}]';

//调用 eval 函数转换为 json 对象，
var myE = eval(normalstring);
document.writeln(myE+'<br><br>');
//将 json 对象转换为字符串
var text = JSON.stringify(myE);
//对比转换后的 json 文本与声明的文本区别
document.writeln('转换后的 json 文本：'+text+'<br><br>声明的 json 格式文本'+jsontext+'<br><br>声明的普通格式文本'+normalstring+'<br><br>');

//当安全比较重要的时候使用 JSON 解析就好一些。JSON 解析只会识别 JSON 文本并且它更安全,下面调用 json 的 parse 函数对文本数据转换生成 json 数据结构
var myData = JSON.parse(jsontext);

document.writeln(myData+'<br><br>');

//下面是对 json 对象的增删查改操作

//声明 json 对象

var                 jsonObj2={persons:[{name:"jordan",sex:"m",age:"40"},
```

```
{name:"bryant",sex:"m",age:"28"}, {name:"McGrady",sex:"m",age:"27"} ]};

var persons=jsonObj2.persons;
var str="";

var person={name:"yaoMing",sex:"m",age:"26"};
//以下为 json 对象的操作，去掉注释可以查看操作结果
//jsonObj2.persons.push(person);//数组最后加一条记录
//jsonObj2.persons.pop();//删除最后一项
//jsonObj2.persons.shift();//删除第一项
jsonObj2.persons.unshift(person);//数组最前面加一条记录 只要适合 Javascript 的方法都是可以用在 JSON 对象的数组中的！所以还有另外的方法 splice( )进行 crud 操作！ //删除
//jsonObj2.persons.splice(0,2);//开始位置,删除个数
//替换不删除
var self={name:"tom",sex:"m",age:"24"};
var brother={name:"Mike",sex:"m",age:"29"};
jsonObj2.persons.splice(1,0,self,brother,self);//开始位置,删除个数,插入对象
//替换并删除
//jsonObj2.persons.splice(0,1,self,brother);//开始位置,删除个数,插入对象

for(var i=0;i<persons.length;i++){ var cur_person=persons[i];
str+=cur_person.name+"'sex is "+cur_person.sex+" and age is "+cur_person.age+"<br><br>"; }
document.writeln(str);
//转换为 json 文本
var myjsonobj = JSON.stringify(jsonObj2);
document.writeln(myjsonobj);
</script>
```

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。易于人阅读和编写。同时也易于机器解析和生成。它基于 JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999 的一个子集。 JSON 采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯（包括 C, C++, C#, Java, JavaScript, Perl, Python 等）。这些特性使 JSON 成为理想的数据交换语言。

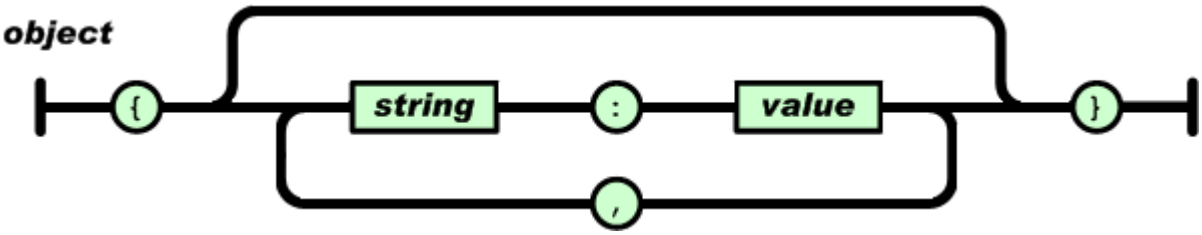
JSON 建构于两种结构：

- “名称/值”对的集合（A collection of name/value pairs）。不同的语言中，它被理解为*对象（object）*，纪录（record），结构（struct），字典（dictionary），哈希表（hash table），有键列表（keyed list），或者关联数组 （associative array）。
- 值的有序列表（An ordered list of values）。在大部分语言中，它被理解为数组（array）。

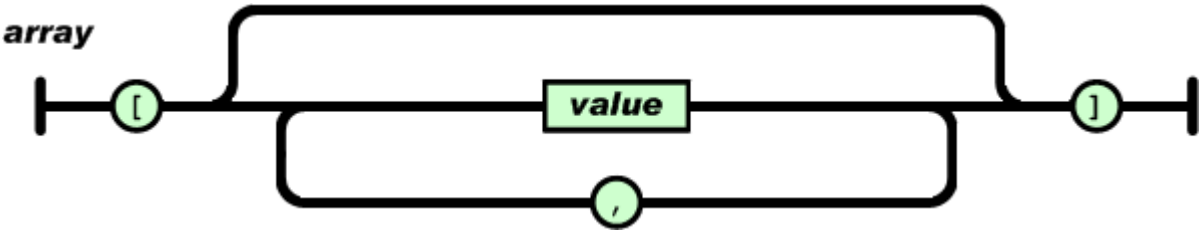
这些都是常见的数据结构。事实上大部分现代计算机语言都以某种形式支持它们。这使得一种数据格式在同样基于这些结构的编程语言之间交换成为可能。

JSON 具有以下这些形式：

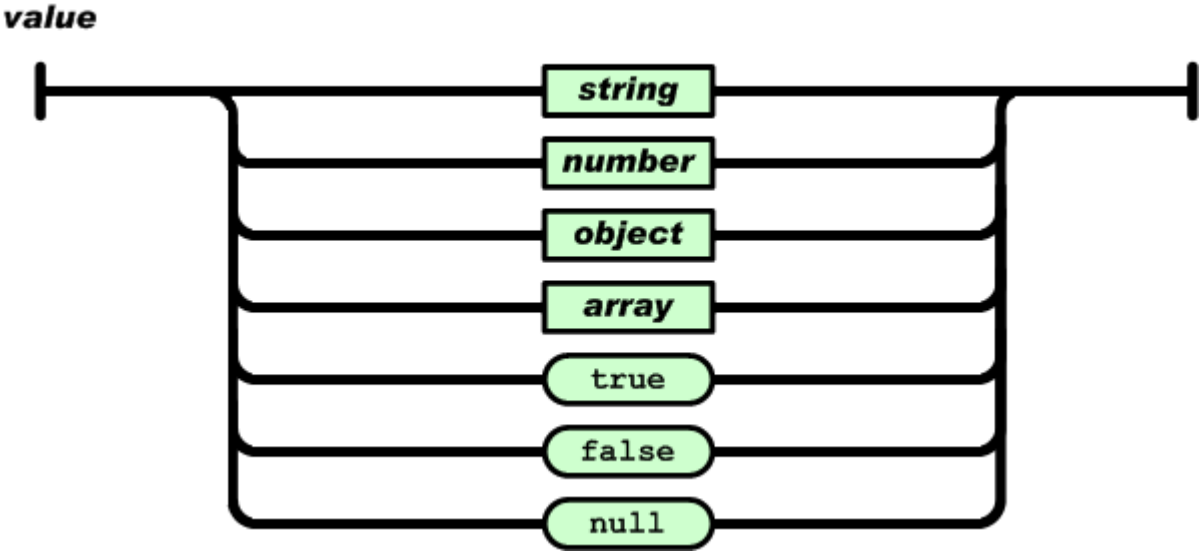
对象是一个无序的“‘名称/值’对”集合。一个对象以“{”（左括号）开始，“}”（右括号)结束。每个“名称”后跟一个“:”（冒号)；“‘名称/值’ 对”之间使用“,”（逗号）分隔。



数组是值（value）的有序集合。一个数组以“[”（左中括号）开始，“]”（右中括号）结束。值之间使用“,”（逗号）分隔。

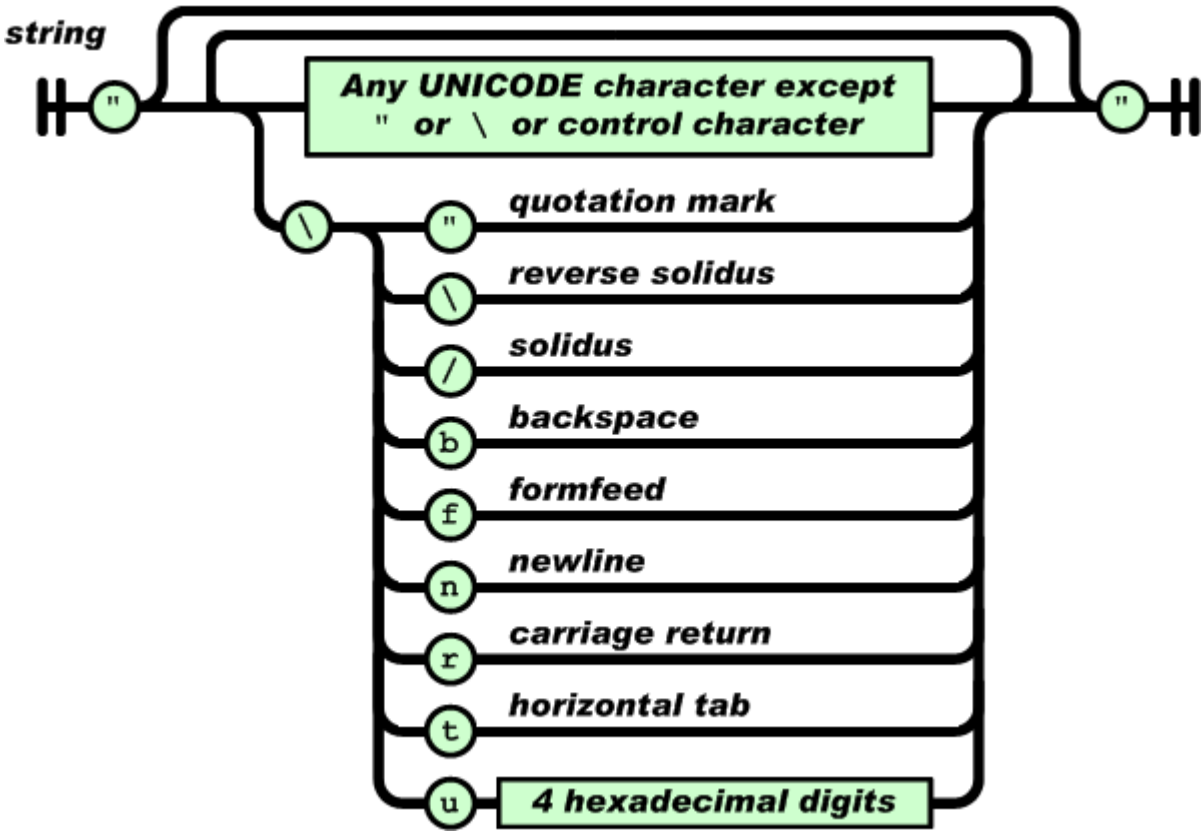


值（*value*）可以是双引号括起来的字符串（*string*）、数值(number)、true、false、 null、对象（object）或者数组（array）。这些结构可以嵌套。

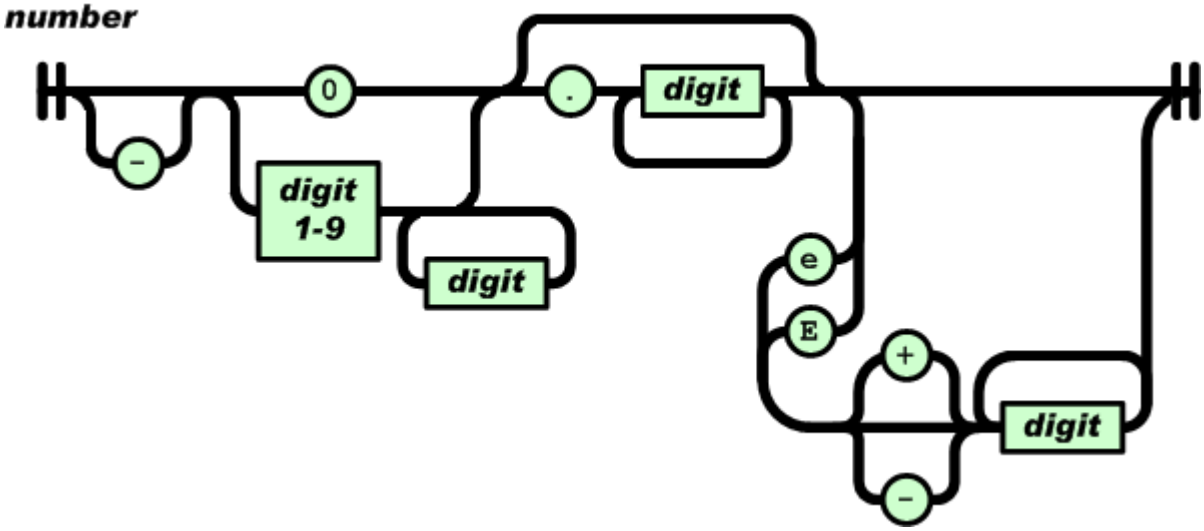


字符串（*string*）是由双引号包围的任意数量 Unicode 字符的集合，使用反斜线转义。一个字符（character）即一个单独的字符串（character string）。

字符串（*string*）与 C 或者 Java 的字符串非常相似。



数值（*number*）也与 C 或者 Java 的数值非常相似。除去未曾使用的八进制与十六进制格式。除去一些编码细节。



空白可以加入到任何符号之间。 以下描述了完整的语言。

JavaScript 中的 JSON

- 作者：Douglas Crockford
- 原文网址：http://www.json.org/js.html
- 译者：可爱的猴子

JavaScript 这种编程语言首要的目的是为 Netscape Navigator 提供一种页面脚本语言。它仍被普遍的认为是 Java 的一个子集，但事实并非如此。它是一种语法类似 c 语言并且支持面向对象的 Scheme-like 语言。JavaScript 使用了 ECMAScript 语言规范第三版进行了标准化。

JSON 是 JavaScript 面向对象语法的一个子集。由于 JSON 是 JavaScript 的一个子集，因此它可清晰的运用于此语言中。

```
var myJSONObject = {"bindings": [
        {"ircEvent": "PRIVMSG", "method": "newURI", "regex":
"^http://.*"},
        {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex":
"^delete.*"},
        {"ircEvent": "PRIVMSG", "method": "randomURI", "regex":
"^random.*"}
    ]
};
```

上面的示例，创建了一个包括单独成员”bindings”的对象，此成员包括一个含有三个对象（”ircEvent”, “method”, 与 “regex”）的数组

成员可以通过.或者下标操作符检索。

```
myJSONObject.bindings[0].method    // "newURI"
```

为了将 JSON 文本转换为对象，可以使用 eval()函数。eval()函数调用 JavaScript 编辑器。由于 JSON 是 JavaScript 的子集，因此编译器将正确的解析文本并产生对象结构。文本必须括在括号中避免产生 JavaScript 的语法歧义。

```
var myObject = eval('(' + myJSONtext + ')');
```

eval 函数非常快速。它可以编译执行任何 JavaScript 程序，因此产生了安全性问题。当使用可信任与完善的源代码时才可以使用 eval 函数。这样可以更安全的使用 JSON 解析器。使用 XMLHttpRequest 的 web 应用，页面之间的通讯只允许同源，因此是可以信任的。但这却不是完善的。如果服务器没有严谨的 JSON 编码，或者没有严格的输入验证，那么可能传送包括危险脚本的无效 JSON 文本。eval 函数将执行恶意的脚本。

使用 JSON 解析器可以防止此类事件。JSON 解析器只能辨识 JSON 文本，拒绝所有脚本。提供了本地 JSON 支持的浏览器的 JSON 解析器将远快于 eval 函数。预计未来的 ECMAScript 标准将支持本地 JSON。

```
var myObject = JSON.parse(myJSONtext, reviver);
```

一个替换函数(reviver function)做为可选参数被最终结果的每一级的键(key)与值（value）调用。 每个值都将被替换函数的值代替。这可以用来将一般的类改变成伪类的实例，或者将日期字符串转变为日期对象。

```
myData = JSON.parse(text, function (key, value) {
    var type;
    if (value && typeof value === 'object') {
        type = value.type;
        if (typeof type === 'string' && typeof window[type] ===
'function') {
            return new (window[type])(value);
        }
    }
    return value;
});
```

JSON stringifier 进行反向操作，可以把 JavaScript 数据结构转换为 JSON 文本。JSON 不支持循环数据结构，因此应小心不要为 JSON stringifier 提供循环结构。

```
var myJSONText = JSON.stringify(myObject, replacer);
```

如果 stringify 函数发现一个带有 toJSON 方法的对象，它将执行此方法，并且返回产生的值。这样一个对象就可以决定自己的 JSON 表现。

stringifier 方法可以携带一个可选的字符串数组。这些字符串被用于选择包括在 JSON 文本中的属性。

stringifier 方法可以携带一个可选的替代(replacer)函数。它将在结构中每个值的 toJSON 方法（如果有的话）后面执行。它将每个键与值做为参数传递，当然对象要包含这个键。值将被 stringified 返回。

如果没有提供数组或替代函数，一个用于忽略被集成的属性的可选替代函数将被提供。如果想要所有被继承的属性，可以提供一个简单的替换函数：

```
var myJSONText = JSON.stringify(myObject, function (key, value) {
    return value;
});
```

值在 JSON 中不代表任何内容，函数与未定义（undefined）被排除在外。

不能确定的数量将被替换为 null。为了替代其它的值，可以像下面一样使用替换（replacer）函数

```
function replacer(key, value) {
    if (typeof value === 'number' && !isFinite(value)) {
        return String(value);
    }
    return value;
}
```

开放源代码的 JSON 解析器与 JSON stringifier 可以使用。通过 minified 可以小于 2.5K。