第十章　JSON 的应用

本章单词

1. collection: _____　　2. dictionary: _____
3. record: _____　　4. associative: _____
5. pairs: _____　　6. Natation: _____

JSON 即 JavaScript Object Natation，它是一种轻量级的数据交换格式，非常适合于服务器与 JavaScript 的交互。本文将快速讲解 JSON 格式，并通过代码示例演示如何分别在客户端和服务器端进行 JSON 格式数据的处理。

尽管有许多宣传关于 XML 如何拥有跨平台，跨语言的优势，然而，除非应用于 Web Services，否则，在普通的 Web 应用中，开发者经常为 XML 的解析伤透了脑筋，无论是服务器端生成或处理 XML，还是客户端用 JavaScript 解析 XML，都常常导致复杂的代码，极低的开发效率。实际上，对于大多数 Web 应用来说，他们根本不需要复杂的 XML 来传输数据，XML 的扩展性很少具有优势，许多 AJAX 应用甚至直接返回 HTML 片段来构建动态 Web 页面。和返回 XML 并解析它相比，返回 HTML 片段大大降低了系统的复杂性，但同时缺少了一定的灵活性。

现在， JSON 为 Web 应用开发者提供了另一种数据交换格式。让我们来看看 JSON 到底是什么，同 XML 或 HTML 片段相比，JSON 提供了更好的简单性和灵活性。

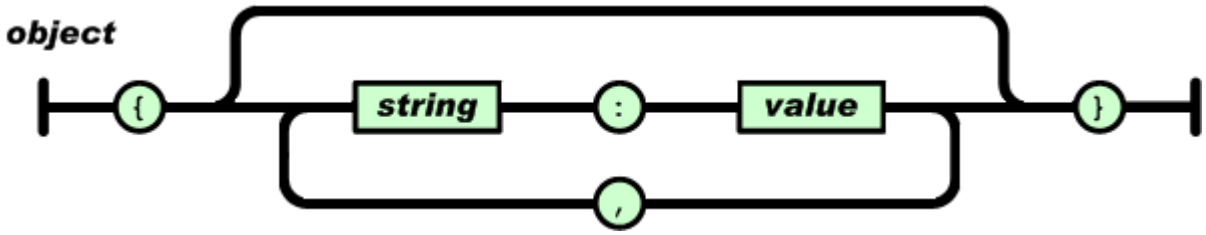
10.1 JSON 的形式

JSON 建构于两种结构：

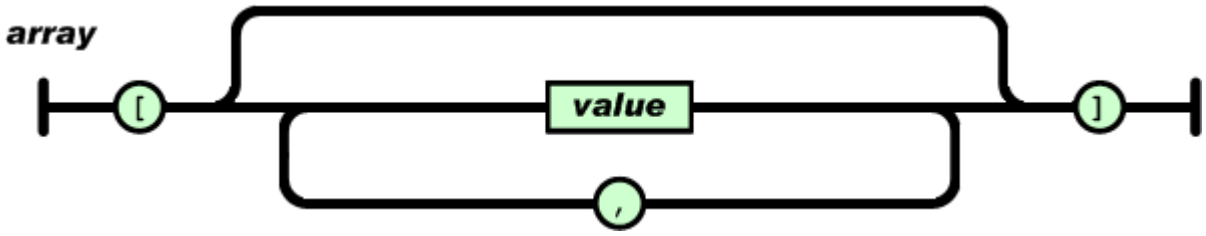
- “名称/值”对的集合（A collection of name/value pairs）。不同的语言中，它被理解为对象（object），纪录（record），结构（struct），字典（dictionary），哈希表（hash table），有键列表（keyed list），或者关联数组 （associative array）。
- 值的有序列表（An ordered list of values）。在大部分语言中，它被理解为数组（array）。

JSON 具有以下这些形式：

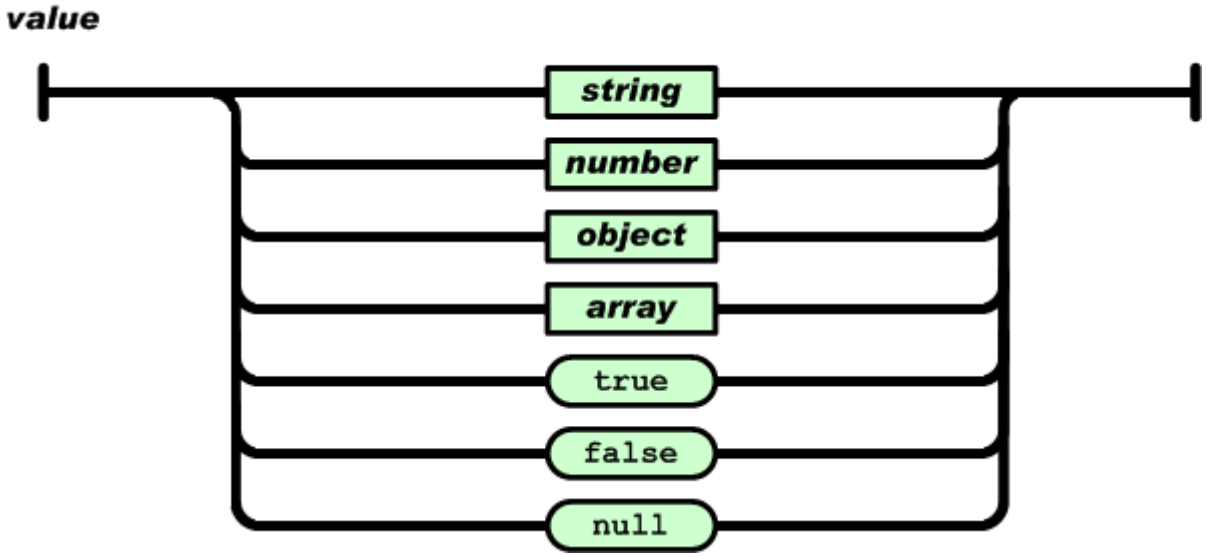
对象是一个无序的“‘名称/值’对”集合。一个对象以“{”（左括号）开始，“}”（右括号）结束。每个“名称”后跟一个“:”（冒号）；“‘名称/值’ 对”之间使用“,”（逗号）分隔。



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。

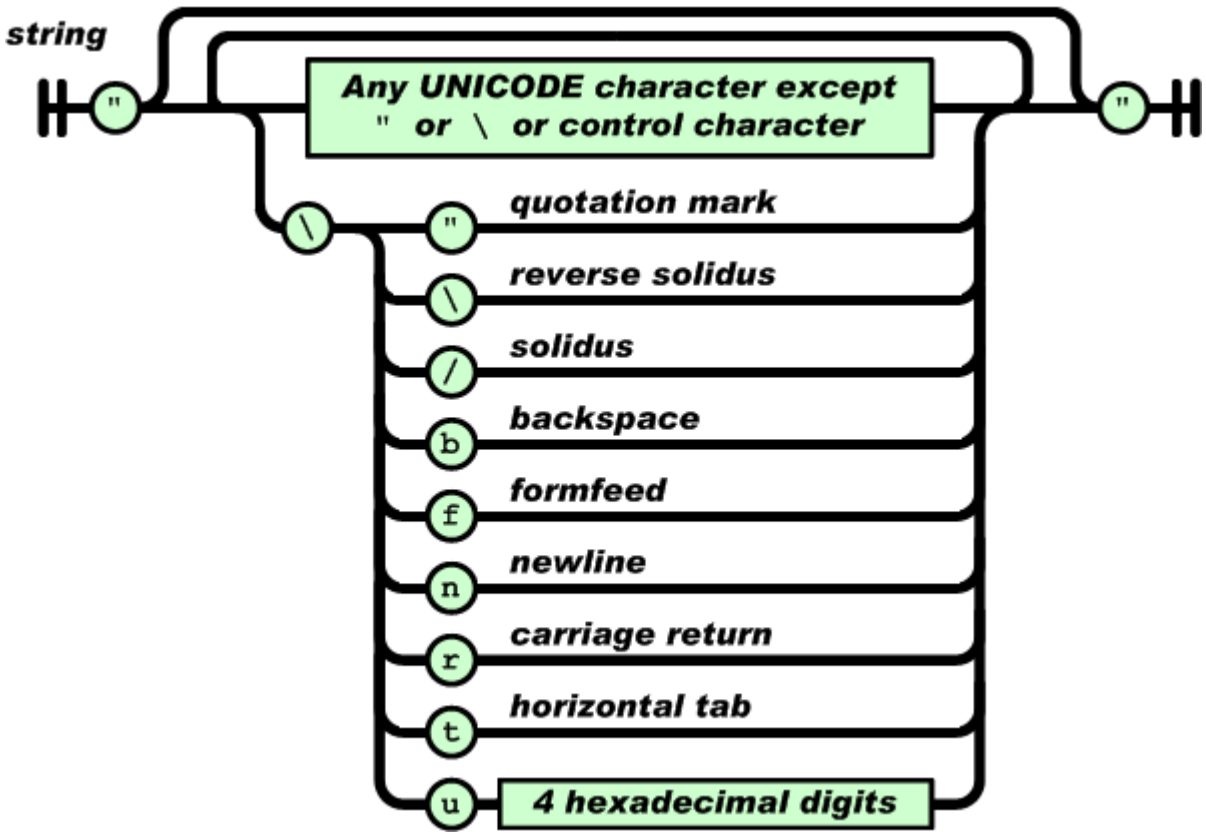


值(value)可以是双引号括起来的字符串(string)、数值(number)、true、false、 null、对象(object)或者数组（array)。这些结构可以嵌套。

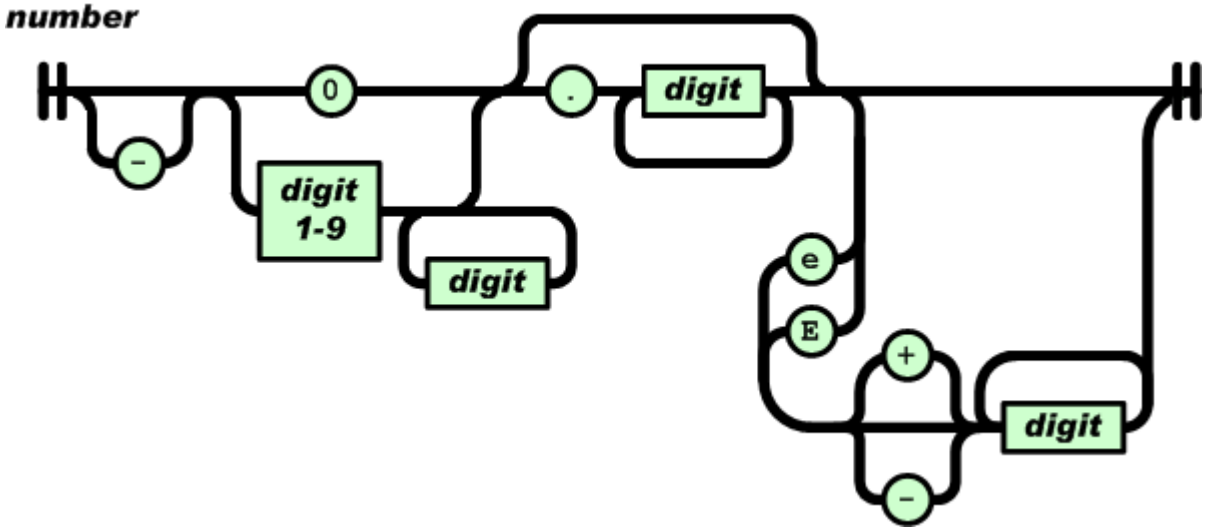


字符串（string）是由双引号包围的任意数量 Unicode 字符的集合，使用反斜线转义。一个字符（character）即一个单独的字符串（character string)。

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。



10.2 JSON 的定义与使用

比如，后台载入一些用户的基本信息，如果写成 XML，如下：

```
<content>
<user>
    <username>andy</username>
    <age>20</age>
    <info>
        <tel>123456</tel>
```

```
            <cellphone>98765</tel>
        </info>
        <address>
            <city>Beijing</city>
            <postcode>222333</postcode>
        </address>
        <address>
            <city>newyork</city>
            <postcode>555666</postcode>
        </address>
    </user>
    </content>
```

而写成 JSON 呢：

```
    function showJSON() {
       var user =
        {
          "username":"andy",
          "age":20,
          "info": { "tel": "123456", "cellphone": "98765"},
          "address":
              [
                 {"city":"beijing","postcode":"222333"},
                 {"city":"newyork","postcode":"555666"}
              ]
        }

        alert(user.username);
        alert(user.age);
        alert(user.info.cellphone);
        alert(user.address[0].city);
        alert(user.address[0].postcode);
    }
```

简单的不只是表达上，最重要的是可以丢弃让人晕头转向的 DOM 解析了。因为只要符合 JavaScript 的声明规范，JavaScrip 会自动帮你解析好 的。

上面的 JSON 表示一个 user 对象，拥有 username, age, info, address 等属性

同样也可以用 JSON 来简单的修改数据，修改上面的例子

```
    user.username = "Tom";
```

JSON 提供了 json.js 包，下载 http://www.json.org/json.js 后，将其引入然后就可以简单的使用 object.toJSONString()转换成 JSON 数据。

```
    function showCar() {
       var carr = new Car("Dodge", "Coronet R/T", 1968, "yellow");
        alert(carr.toJSONString());
    }

    function Car(make, model, year, color)        {
```

```
        this.make    =    make;
        this.model    =    model;
        this.year    =    year;
        this.color    =    color;
    }
```

可以使用 eval 来转换字符到 JSON Object

```
    function myEval() {
        var str = '{ "name": "Violet", "occupation": "character" }';
        var obj = eval('(' + str + ')');
         alert(obj.toJSONString());
    }
```

或者使用 parseJSON()方法

```
    function myEval() {
        var str = '{ "name": "Violet", "occupation": "character" }';
        var obj = str.parseJSON();
        alert(obj.toJSONString());
    }
```

10.3 JSON 与 XML 对比

✧ 可读性

JSON 和 XML 的可读性可谓不相上下，一边是建议的语法，一边是规范的标签形式，很难分出胜负。

✧ 可扩展性

XML 天生有很好的扩展性，JSON 当然也有，没有什么是 XML 能扩展，JSON 不能的。

✧ 编码难度

XML 有丰富的编码工具，比如 Dom4j、JDom 等，JSON 也有 json.org 提供的工具，但是 JSON 的编码明显比 XML 容易许多，即使不借助工具也能写出 JSON 的代码，可是要写好 XML 就不太容易了。

✧ 解码难度

XML 的解析得考虑子节点父节点，让人头昏眼花，而 JSON 的解析难度几乎为 0。

✧ 流行度

XML 已经被业界广泛的使用，而 JSON 才刚刚开始，但是在 Ajax 这个特定的领域，未来的发展一定是 XML 让位于 JSON。到时 Ajax 应该变成 Ajaj(Asynchronous JavaScript and JSON)了。

本章练习

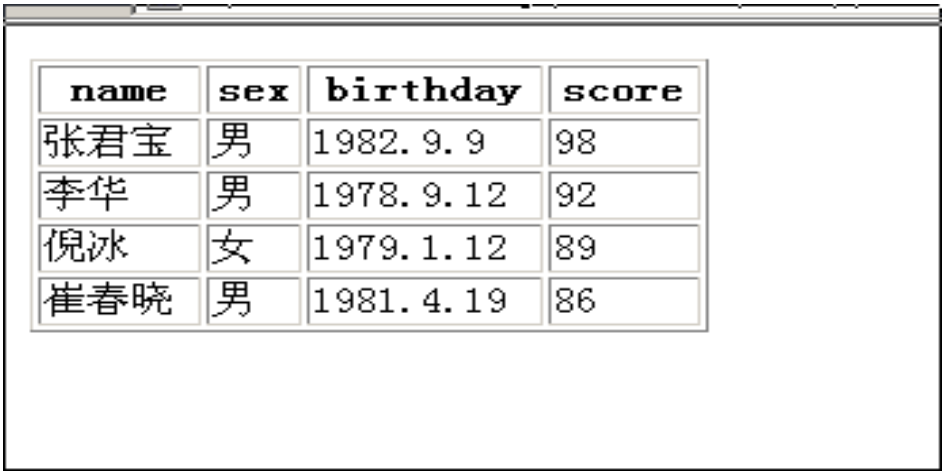
一、请把下面的 XML 文件转换成 JSON 对象

```
<country>
    <province> 江苏
        <city>苏州</city>
        <city>南京</city>
    </province>
</country>
```

二、现有 student.xml 文件内容如下:

```
<?xml version="1.0" encoding="utf-8"?>
<roster>
 <student ID="101">
      <name>张君宝</name>
      <sex>男</sex>
      <birthday>1982.9.9</birthday>
      <score>98</score>
 </student>
 <student ID="102">
      <name>李华</name>
      <sex>男</sex>
      <birthday>1978.9.12</birthday>
      <score>92</score>
 </student>
 <student ID="103">
      <name>倪冰</name>
      <sex>女</sex>
      <birthday>1979.1.12</birthday>
      <score>89</score>
 </student>
 <student ID="104">
      <name>崔春晓</name>
      <sex>男</sex>
      <birthday>1981.4.19</birthday>
      <score>86</score>
 </student>
</roster>
```

将上面的 xml 文件使用 json 形式表示出来，然后使用 js 读取出 json 中的数据生成下面的表格信息:



name	sex	birthday	score
张君宝	男	1982.9.9	98
李华	男	1978.9.12	92
倪冰	女	1979.1.12	89
崔春晓	男	1981.4.19	86

三、现有 list.xml 文件内容如下:

```
<?xml version="1.0" encoding="utf-8"?>
<list>
 <province id="P001">浙江</province>
 <province id="P002">江苏</province>
 <province id="P003">湖南</province>
 <province id="P004">湖北</province>
 <province id="P005">四川</province>
 <province id="P006">江西</province>
 <province id="P007">广东</province>
 <city provinceid="P001">所有地区|杭州|宁波|金华|温州|绍兴|嘉兴|衢州|湖州|台州|舟山|丽水</city>
 <city  provinceid="P002">所有地区|南京|镇江|常州|无锡|苏州|扬州|连云港|南通|宿迁|徐州|淮阴|泰州|盐城</city>
 <city  provinceid="P003">所有地区|长沙|株洲|湘潭|常德|郴州|怀化|邵阳|湘西|益阳|永州|衡阳|娄底|张家界|岳阳</city>
 <city  provinceid="P004">所有地区|武汉|宜昌|恩施|黄冈|荆门|十堰|鄂州|黄石|荆州|襄樊|咸宁|潜江|随州|天门|仙桃|孝感|神农架</city>
 <city  provinceid="P005">所有地区|成都|宜宾|德阳|广安|资阳|凉山|绵阳|南充|南充泸州|遂宁|巴中|阿坝|达州|甘孜|广元|自贡|眉山|攀枝花|乐山|内江|雅安
 </city>
 <city provinceid="P006">所有地区|南昌|上饶|萍乡|九江|赣州|景德镇|鹰潭|新余|抚州|吉安|宜春</city>
 <city  provinceid="P007">所有地区|广州|深圳|珠海|中山|东莞|惠州|潮州|佛山|河源|江门|茂名|清远|汕头|韶关|湛江|揭阳|梅州|汕尾|阳江|云浮|肇庆</city>
</list>
```

将上面的 list.xml 文件的内容使用 json 形式表示出来，然后利用 dom 操作并将省份和城市动态添加到下拉列表中，如下图: