JSON中文版

极客学院出版

前言

本教程会帮助我们了解 JSON 以及如何在各种编程语言，比如 PHP，PERL，Python，Ruby，Java等等编程语言中使用它。

JSON 或者 JavaScript 对象表示法是一种轻量级的基于文本的开放标准，被设计用于可读的数据交换。JSON 格式最初由 Douglas Crockford 提出，使用 RFC 4627 描述。JSON 的官方网络媒体类型是 `application/json` 。JSON 的文件名扩展是 .json。

适用人群

本教程旨在帮助初学者了解 JavaScript 对象表示法（ JSON ）开发数据交换格式的基本功能。完成本教程之后，你会发现自己处于在JavaScript，AJAX，Perl中使用 JSON 的水平为中等，然后你可以自己走向下一个水平。

学习前提

在学习本教程之前，你应该对 Web 应用程序如何通过 HTTP 进行工作有一个基本的了解，并且我们假设你已经掌握了基本的 JavaScript 知识。

更新日期	更新内容
2015-05-05	第一版发布

目录

1 JSON 基础

JSON 教程

JSON 或者 JavaScript 对象表示法是一种轻量级的基于文本的开放标准，被设计用于可读的数据交换。约定使用 JSON 的程序包括 C，C++，Java，Python，Perl 等等。

- JSON 是 JavaScript Object Notation 的缩写。
- 这个格式由 Douglas Crockford 提出。
- 被设计用于可读的数据交换。
- 它是从 JavaScript 脚本语言中演变而来。
- 文件名扩展是 .json。
- JSON 的网络媒体类型是 application/json。
- 统一标示符类型（Uniform Type Identifier）是 public.json。

JSON 使用范围

- 用于编写基于 JavaScript 应用程序，包括浏览器扩展和网站。
- JSON 格式可以用于通过网络连接序列化和传输结构化数据。
- 主要用于在服务器和 Web 应用程序之间传输数据。
- Web 服务和 APIs 可以使用 JSON 格式提供公用数据。
- 还可以用于现代编程语言中。

JSON 特点

- JSON 容易阅读和编写。
- 它是一种轻量级的基于文本的交换格式。
- 语言无关。

JSON 简单示例

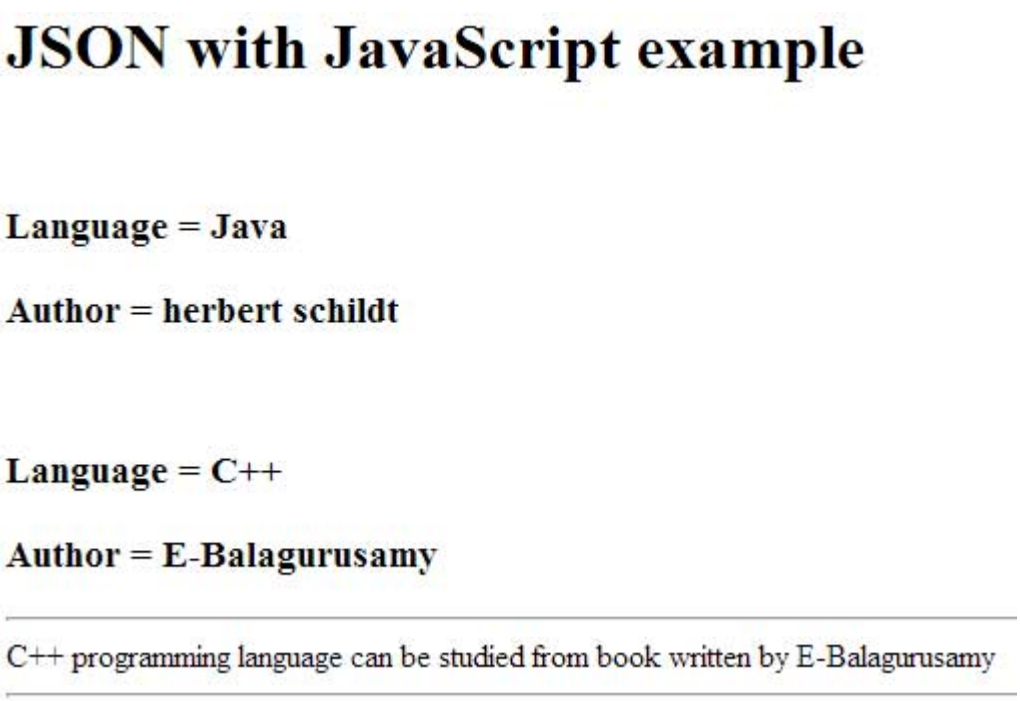
鉴于书籍数据有语言和版本信息，下面的例子展示了使用 JSON 存储书籍信息：

```
{
   "book": [
      {
         "id":"01",
         "language": "Java",
         "edition": "third",
         "author": "Herbert Schildt"
      },
      {
         "id":"07",
         "language": "C++",
         "edition": "second"
         "author": "E.Balagurusamy"
   }]
}
```

理解上述程序之后我们来看另外一个例子，让我们把下面的代码保存为 json.htm：

```
<html>
<head>
<title>JSON example</title>
<script language="javascript" >

   var object1 = { "language" : "Java", "author"  : "herbert schildt" };
   document.write("<h1>JSON with JavaScript example</h1>");
   document.write("<br>");
   document.write("<h3>Language = " + object1.language+"</h3>");
   document.write("<h3>Author = " + object1.author+"</h3>");

   var object2 = { "language" : "C++", "author"  : "E−Balagurusamy" };
   document.write("<br>");
   document.write("<h3>Language = " + object2.language+"</h3>");
   document.write("<h3>Author = " + object2.author+"</h3>");

   document.write("<hr />");
   document.write(object2.language + " programming language can be studied " +
   "from book written by " + object2.author);
   document.write("<hr />");

</script>
</head>
<body>
</body>
</html>
```

现在尝试使用 IE 或者其他任何启用了 JavaScript 的浏览器打开这个页面，它会生成如下所示结果：



图片 1.1 json example demo

你可以参考 JSON 对象 来了解更多关于 JSON 对象的信息。

JSON 语法

我们来快速浏览一下 JSON 的基本语法。JSON 的语法基本上可以视为 JavaScript 语法的一个子集，包括以下内容：

- 数据使用名/值对表示。
- 使用大括号保存对象，每个名称后面跟着一个 ':'（冒号），名/值对使用 ,（逗号）分割。
- 使用方括号保存数组，数组值使用 ,（逗号）分割。

下面是一个简单的示例：

```
{
   "book": [
      {
         "id":"01",
         "language": "Java",
         "edition": "third",
         "author": "Herbert Schildt"
      },
      {
         "id":"07",
         "language": "C++",
         "edition": "second"
         "author": "E.Balagurusamy"
   }]
}
```

JSON 支持以下两种数据结构：

- **名/值对集合：** 这一数据结构由不同的编程语言支持。
- **有序的值列表：** 包括数组，列表，向量或序列等等。

JSON 数据类型

JSON 格式支持以下数据类型：

类型	描述
数字型（Number）	JavaScript 中的双精度浮点型格式
字符串型（String）	双引号包裹的 Unicode 字符和反斜杠转义字符
布尔型（Boolean）	true 或 false
数组（Array）	有序的值序列
值（Value）	可以是字符串，数字，true 或 false，null 等等
对象（Object）	无序的键:值对集合
空格（Whitespace）	可用于任意符号对之间
null	空

数字型

- JavaScript 中的双精度浮点型格式，取决于实现。
- 不能使用八进制和十六进制格式。
- 在数字中不能使用 NaN 和 Infinity。

下表展示了数字类型：

类型	描述
整形（Integer）	数字1-9，0和正负数
分数（Fraction）	分数，比如 .3，.9
指数（Exponent）	指数，比如 e，e+，e-，E，E+，E-

语法：

```
var json-object-name = { string : number_value, .....}
```

示例：

下面的示例展示了数字类型，其值不应该使用引号包裹：

```
var obj = {marks: 97}
```

字符串型

- 零个或多个双引号包裹的 Unicode 字符以及反斜杠转义序列。
- 字符就是只有一个字符的字符串，长度为 1。

下表展示了字符串类型：

类型	描述
"	双引号
\	反斜线
/	斜线
b	退格符
f	换页符
n	换行符
r	回车符
t	水平制表符
u	四位十六进制数字

语法：

```
var json-object-name = { string : "string value", .....}
```

示例：

下面的示例展示了字符串数据类型：

```
var obj = {name: 'Amit'}
```

布尔型

它包含 true 和 false 两个值。

语法：

```
var json-object-name = { string : true/false, .....}
```

示例：

```
var obj = {name: 'Amit', marks: 97, distinction: true}
```

数组

- 它是一个有序的值集合。
- 使用方括号闭合，这意味着数组以 [开始，以] 结尾。
- 值使用 , (逗号) 分割。
- 数组索引可以以 0 或 1 开始。
- 当键名是连续的整数时应该使用数组。

语法：

```
[ value, .....]
```

示例：

下面的示例展示了一个包含多个对象的数组：

```
{
   "books": [
      { "language":"Java" , "edition":"second" },
      { "language":"C++" , "lastName":"fifth" },
      { "language":"C" , "lastName":"third" }
   ]
}
```

对象

- 它是一个无序的名/值对集合。
- 对象使用大括号闭合，以 '{' 开始，以 '}' 结尾。
- 每个名称后面都跟随一个 ':' (冒号)，名/值对使用 , (逗号) 分割。
- 键名必须是字符串，并且不能同名。
- 当键名是任意字符串时应该使用对象。

语法：

```
{ string : value, .....}
```

示例：

下面的例子展示了对象：

```
{
   "id": "011A",
   "language": "JAVA",
   "price": 500,
}
```

空格

可以在任意一对符号之间插入。可以添加用来让代码更可读。下面的例子展示了使用空格和不使用空格的声明：

语法：

```
{string:"   ",....}
```

示例：

```
var i= "   sachin";
var j = "  saurav"
```

null

意味着空类型。

语法：

```
null
```

示例：

```
var i = null;

if(i==1) {
   document.write("<h1>value is 1</h1>");
} else {
   document.write("<h1>value is null</h1>");
}
```

JSON 值

包括：

- 数字（整型和浮点型）
- 字符串
- 布尔值
- 数组
- 对象
- null

语法：

```
String | Number | Object | Array | TRUE | FALSE | NULL
```

示例：

```
var i =1;
var j = "sachin";
var k = null;
```

JSON 对象

创建简单的对象

JSON 对象可以使用 JavaScript 创建。我们来看看使用 JavaScript 创建 JSON 对象的各种方式：

- 创建一个空对象：

```
var JSONObj = {};
```

- 创建一个新对象：

```
var JSONObj = new Object();
```

- 创建一个 **bookname** 属性值为字符串，**price**属性值为数字的对象。可以通过使用 '.' 运算符访问属性。

```
var JSONObj = { "bookname ":"VB BLACK BOOK", "price":500 };
```

这里有一个例子，展示了在 JavaScript 中使用 JSON 创建对象，可以将下面的代码保存为 *json_object.htm*：

```
<html>
<head>
<title>Creating Object JSON with JavaScript</title>
<script language="javascript" >

   var JSONObj = { "name" : "tutorialspoint.com", "year"  : 2005 };
   document.write("<h1>JSON with JavaScript example</h1>");
   document.write("<br>");
   document.write("<h3>Website Name="+JSONObj.name+"</h3>");
   document.write("<h3>Year="+JSONObj.year+"</h3>");

</script>
</head>
<body>
</body>
</html>
```

现在让我们尝试使用 IE 或者任何其他启用了 JavaScript 的浏览器打开这个页面，这会生成如下所示结果：

图片 1.2 json object

创建数组对象

下面的例子展示了在 JavaScript 中使用 JSON 创建数组对象，可以将下面的代码保存为 json_array_object.htm：

```
<html>
<head>
<title>Creation of array object in javascript using JSON</title>
<script language="javascript" >

document.writeln("<h2>JSON array object</h2>");

var books = {
   "Pascal" : [
      { "Name"  : "Pascal Made Simple", "price" : 700 },
      { "Name"  : "Guide to Pascal", "price" : 400 }
   ],
   "Scala"  : [
      { "Name"  : "Scala for the Impatient", "price" : 1000 },
      { "Name"  : "Scala in Depth", "price" : 1300 }
   ]
}

var i = 0
document.writeln("<table border='2'><tr>");
for(i=0;i<books.Pascal.length;i++)
{
   document.writeln("<td>");
   document.writeln("<table border='1' width=100 >");
   document.writeln("<tr><td><b>Name</b></td><td width=50>"
   + books.Pascal[i].Name+"</td></tr>");
   document.writeln("<tr><td><b>Price</b></td><td width=50>"
```

```
   + books.Pascal[i].price +"</td></tr>");
   document.writeln("</table>");
   document.writeln("</td>");
}

for(i=0;i<books.Scala.length;i++)
{
   document.writeln("<td>");
   document.writeln("<table border='1' width=100 >");
   document.writeln("<tr><td><b>Name</b></td><td width=50>"
   + books.Scala[i].Name+"</td></tr>");
   document.writeln("<tr><td><b>Price</b></td><td width=50>"
   + books.Scala[i].price+"</td></tr>");
   document.writeln("</table>");
   document.writeln("</td>");
}
document.writeln("</tr></table>");
</script>
</head>
<body>
</body>
</html>
```

现在让我们尝试使用 IE 或者任意其他启用了 JavaScript 的浏览器打开这个页面，和会生成如下所示结果：

JSON array object

Name	Pascal Made Simple	**Name**	Guide to Pascal	**Name**	Scala for the Impatient	**Name**	Scala in Depth
Price	700	**Price**	400	**Price**	1000	**Price**	1300

图片 1.3 json array object

JSON 模式（Schema）

JSON 模式是一种基于 JSON 格式定义 JSON 数据结构的规范。它被写在 IETF 草案下并于 2011 年到期。JSON 模式：

- 描述现有数据格式。
- 干净的人类和机器可读的文档。
- 完整的结构验证，有利于自动化测试。
- 完整的结构验证，可用于验证客户端提交的数据。

JSON 模式验证库

目前有好几个验证器可用于不同的编程语言。但是目前最完整和兼容 JSON 模式的验证器是 JSV。

语言	程序库
C	WJElement (LGPLv3)
Java	json-schema-validator (LGPLv3)
.NET	Json.NET (MIT)
ActionScript 3	Frigga (MIT)
Haskell	aeson-schema (MIT)
Python	Jsonschema
Ruby	autoparse (ASL 2.0); ruby-jsonschema (MIT)
PHP	php-json-schema (MIT). json-schema (Berkeley)
JavaScript	Orderly (BSD); JSV; json-schema; Matic (MIT); Dojo; Persevere (modified BSD or AFL 2.0); schema.js.

JSON 模式示例

下面是一个基本的 JSON 模式，其中涵盖了一个经典的产品目录说明：

```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "title": "Product",
   "description": "A product from Acme's catalog",
   "type": "object",
```

```
  "properties": {
    "id": {
      "description": "The unique identifier for a product",
      "type": "integer"
    },
    "name": {
      "description": "Name of the product",
      "type": "string"
    },
    "price": {
      "type": "number",
      "minimum": 0,
      "exclusiveMinimum": true
    }
  },
  "required": ["id", "name", "price"]
}
```

我们来看一下可以用于这一模式中的各种重要关键字：

关键字	描述
$schema	$schema 关键字状态，表示这个模式与 v4 规范草案书写一致。
title	用它给我们的模式提供了标题。
description	关于模式的描述。
type	type 关键字在我们的 JSON 数据上定义了第一个约束：必须是一个 JSON 对象。
properties	定义各种键和他们的值类型，以及用于 JSON 文件中的最小值和最大值。
required	存放必要属性列表。
minimum	给值设置的约束条件，表示可以接受的最小值。
exclusiveMinimum	如果存在 "exclusiveMinimum" 并且具有布尔值 true，如果它严格意义上大于 "minimum" 的值则实例有效。
maximum	给值设置的约束条件，表示可以接受的最大值。
exclusiveMaximum	如果存在 "exclusiveMinimum" 并且具有布尔值 true，如果它严格意义上小于 "maximum" 的值则实例有效。
multipleOf	如果通过这个关键字的值分割实例的结果是一个数字则表示紧靠 "multipleOf" 的数字实例是有效的。
maxLength	字符串实例字符的最大长度数值。
minLength	字符串实例字符的最小长度数值。
pattern	如果正则表达式匹配实例成功则字符串实例被认为是有效的。

可以在 http://json-schema.org 上检出可用于定义 JSON 模式的完整关键字列表。上面的模式可用于测试下面给出的 JSON 代码的有效性：

```
[
  {
    "id": 2,
    "name": "An ice sculpture",
    "price": 12.50,
  },
  {
    "id": 3,
    "name": "A blue mouse",
    "price": 25.50,
  }
]
```

JSON 与 XML 对比

JSON 和 XML 都是人类可读的格式并且与语言无关。在现实环境中它们都支持创建，读取和解码。我们可以基于以下因素来比较 JSON 和 XML：

冗余度

XML 比 JSON 冗余，因此对我们来说编写 JSON 会更快。

数组用法

XML 被用来描述结构化数据，不包含数组；而 JSON 包含数组。

解析

可以使用 JavaScript 的 eval 方法解析 JSON。当针对 JSON 应用这个方法时，eval 返回描述的对象。

示例

下面分别展示了一个 XML 和 JSON 示例：

JSON：

```
{
   "company": Volkswagen,
   "name": "Vento",
   "price": 800000
}
```

XML：

```
<car>
  <company>Volkswagen</company>
  <name>Vento</name>
  <price>800000</price>
</car>
```

2

JSON 示例

在 PHP 中使用 JSON

本教程将会教我们如何使用 PHP 编程语言编码和解码 JSON 对象。让我们先来准备环境以便针对 JSON 进行 PHP 编程。

环境

从 PHP 5.2.0 开始默认捆绑了 JSON 扩展并被编译到 PHP 中。

JSON 函数

函数	程序库
json_encode	返回一个值的 JSON 表示形式。
json_decode	解码为一个 JSON 字符串。
json_last_error	返回最后一次出现的错误。

使用 PHP 编码 JSON（json_encode）

PHP 的 json_encode() 函数用于在 PHP 中编码 JSON。编码成功时这个函数返回给定值的 JSON 表示形式，失败则返回 FALSE。

语法：

```
string json_encode ( $value [, $options = 0 ] )
```

参数：

- value: 要编码的值。这个函数只能用于 UTF-8 编码的数据。
- options: 这个可选值是一个由 JSON_HEX_QUOT, JSON_HEX_TAG, JSON_HEX_AMP, JSON_HEX_APOS, JSON_NUMERIC_CHECK,JSON_PRETTY_PRINT, JSON_UNESCAPED_SLASHES, JSON_FORCE_OBJECT 组成的位掩码。

示例：

下面的例子展示了如何使用 PHP 将一个数组转换为 JSON：

```
<?php
   $arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
   echo json_encode($arr);
?>
```

执行时会生成如下结果：

```
{"a":1,"b":2,"c":3,"d":4,"e":5}
```

下面的例子展示了 PHP 对象也可以被转换为 JSON：

```
<?php
   class Emp {
      public $name = "";
      public $hobbies  = "";
      public $birthdate = "";
   }
   $e = new Emp();
   $e->name = "sachin";
   $e->hobbies  = "sports";
   $e->birthdate = date('m/d/Y h:i:s a', strtotime("8/5/1974 12:20:03"));

   echo json_encode($e);
?>
```

执行时会生成如下所示结果：

```
{"name":"sachin","hobbies":"sports","birthdate":"08\/05\/1974 12:20:03 pm"}
```

使用 PHP 解码 JSON（json_decode）

PHP 的 json-decode() 函数用于在 PHP 中解码 JSON。这个函数返回从 JSON 解码到适当 PHP 类型的值。

语法：

```
mixed json_decode ($json [,$assoc = false [, $depth = 512 [, $options = 0 ]]])
```

参数：

- json_string: 编码的字符串，必须是 UTF-8 编码的数据。
- assoc: 它是一个布尔类型的参数，当设置为 TRUE 时，返回的对象将会被转换为关联数组。
- depth: 它是一个整型参数，用于指定递归深度。
- options: 它是一个 JSON 解码的整型位掩码。支持 JSON_BIGINT_AS_STRING。

示例：

下面例子展示了如何使用 PHP 解码 JSON 对象：

```
<?php
   $json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

   var_dump(json_decode($json));
   var_dump(json_decode($json, true));
?>
```

执行时生成如下所示结果：

```
object(stdClass)#1 (5) {
   ["a"] => int(1)
   ["b"] => int(2)
   ["c"] => int(3)
   ["d"] => int(4)
   ["e"] => int(5)
}

array(5) {
   ["a"] => int(1)
   ["b"] => int(2)
   ["c"] => int(3)
   ["d"] => int(4)
   ["e"] => int(5)
}
```

在 Perl 中使用 JSON

本教程将会教我们如何使用 Perl 编程语言编码和解码 JSON 对象。让我们先来准备环境以便针对 JSON 进行 Perl 编程。

环境

在开始使用 Perl 编码和解码 JSON 之前，我们需要安装 JSON 模块，可以从 CPAN 中获取。下载 JSON-2.53.tar.gz 或者其他任意最新版本之后请按照下列步骤操作：

```
$tar xvfz JSON-2.53.tar.gz
$cd JSON-2.53
$perl Makefile.PL
$make
$make install
```

JSON 函数

函数	程序库
encode_json	将给定的 Perl 数据结构转为 UTF-8 编码的二进制字符串。
decode_json	解码 JSON 字符串。
to_json	将给定的 Perl 数据结构转换为 JSON 字符串。
from_json	接受一个 JSON 字符串并试图解析它，返回结果引用。
convert_blessed	给这个函数传递 true，则 Perl 可以使用这个对象类的 TO_JSON 方法将对象转换为 JSON。

使用 Perl 编码 JSON（encode_json）

Perl 的 encode_json() 函数可以将给定的 Perl 数据结构转换为 UTF-8 编码的二进制字符串。

语法：

```
$json_text = encode_json ($perl_scalar );
or
$json_text = JSON->new->utf8->encode($perl_scalar);
```

示例：

下面的例子展示了使用 Perl 将数组转换为 JSON：

```
 #!/usr/bin/perl

use JSON;
my %rec_hash = ('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
my $json = encode_json \%rec_hash;
print "$json\n";
```

执行时生成如下所示结果：

```
{"e":5,"c":3,"a":1,"b":2,"d":4}
```

下面的例子展示了如何将 Perl 对象转换为 JSON：

```
 #!/usr/bin/perl

package Emp;
sub new
{
   my $class = shift;
   my $self = {
   name => shift,
      hobbies => shift,
      birthdate => shift,
   };
   bless $self, $class;
   return $self;
}
sub TO_JSON { return { %{ shift() } }; }

package main;
use JSON;
my $JSON = JSON->new->utf8;
$JSON->convert_blessed(1);

$e = new Emp( "sachin", "sports", "8/5/1974 12:20:03 pm");
$json = $JSON->encode($e);
print "$json\n";
```

执行时生成如下所示结果：

```
{"birthdate":"8/5/1974 12:20:03 pm","name":"sachin","hobbies":"sports"}
```

使用 Perl 解码 JSON（decode_json）

Perl 的 decode_json() 函数用于在 Perl 中解码 JSON。这个函数返回从 JSON 解码到适当 Perl 类型的值。

语法：

```
$perl_scalar = decode_json $json_text
or
$perl_scalar = JSON->new->utf8->decode($json_text)
```

示例：

下面的例子展示了如何使用 Perl 解码 JSON 对象。如果你的机器上没有 Data::Dumper 模块那么你需要安装这个模块。

```
 #!/usr/bin/perl

use JSON;
use Data::Dumper;

$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

$text = decode_json($json);
print Dumper($text);
```

执行时生成如下所示结果：

```
$VAR1 = {
    'e' => 5,
    'c' => 3,
    'a' => 1,
    'b' => 2,
    'd' => 4
};
```

在 Python 中使用 JSON

本教程将会教我们如何使用 Python 编程语言编码和解码 JSON。让我们先来准备环境以便针对 JSON 进行 Python 编程。

环境

在我们使用 Python 编码和解码 JSON 之前，我们需要安装一个可用 JSON 模块。对于本教程请按照如下方式下载和安装 Demjson：

```
$tar xvfz demjson-1.6.tar.gz
$cd demjson-1.6
$python setup.py install
```

JSON 函数

函数	程序库
encode	将 Python 对象编码为 JSON 字符串表示。
decode	将 JSON 编码的字符串解码为 Python 对象。

使用 Python 编码 JSON（encode）

Python 的 encode() 函数用于将 Python 对象编码为 JSON 字符串表示。

语法：

```
demjson.encode(self, obj, nest_level=0)
```

示例：

下面的例子展示了使用 Python 将数组转换为 JSON：

```
 #!/usr/bin/python

import demjson

data = [ { 'a' : 1, 'b' : 2, 'c' : 3, 'd' : 4, 'e' : 5 } ]
```

```
json = demjson.encode(data)
print json
```

执行时会生成如下所示结果：

```
[{"a":1,"b":2,"c":3,"d":4,"e":5}]
```

使用 Python 解码 JSON（decode）

Python 可以使用 demjson.decode() 函数处理 JSON 解码。这个函数返回从 JSON 解码到适当 Python 类型的值。

语法：

```
demjson.decode(self, txt)
```

示例：

下面的例子展示了如何使用 Python 解码 JSON 对象。

```
 #!/usr/bin/python

import demjson

json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

text = demjson.decode(json)
print text
```

执行时生成如下所示结果：

```
{u'a': 1, u'c': 3, u'b': 2, u'e': 5, u'd': 4}
```

在 Ruby 中使用 JSON

本教程将会教我们如何使用 Ruby 编程语言编码和解码 JSON。让我们先来准备环境以便针对 JSON 进行 Ruby 编程。

环境

在我们使用 Ruby 编码和解码 JSON 之前，我们需要安装一个可用于 Ruby 的 JSON 模块。你可能需要安装 Ruby gem，如果你使用的是最新版的 Ruby，那么你必须在你的机器上安装 gem，安装好 gem 之后请按照下面的步骤操作：

```
$gem install json
```

使用 Ruby 解析 JSON

下面的例子展示了前 2 个键持有字符串值，最后 3 个键持有字符串数组。我们把下面的内容保存为叫做 input.json 的文件。

```
{
   "President": "Alan Isaac",
   "CEO": "David Richardson",
   "India": [
      "Sachin Tendulkar",
      "Virender Sehwag",
      "Gautam Gambhir"
   ],
   "Srilanka": [
      "Lasith Malinga",
      "Angelo Mathews",
      "Kumar Sangakkara"
   ],
   "England": [
      "Alastair Cook",
      "Jonathan Trott",
      "Kevin Pietersen"
   ]
}
```

下面是用于解析上述 JSON 文档的 Ruby 程序：

```
 #!/usr/bin/ruby

require 'rubygems'
require 'json'
require 'pp'

json = File.read('input.json')
obj = JSON.parse(json)

pp obj
```

执行时生成如下所示结果：

```
{
   "President"=>"Alan Isaac",
   "CEO"=>"David Richardson",

   "India"=>
      ["Sachin Tendulkar", "Virender Sehwag", "Gautam Gambhir"],

   "Srilanka"=>
      ["Lasith Malinga ", "Angelo Mathews", "Kumar Sangakkara"],

   "England"=>
      ["Alastair Cook", "Jonathan Trott", "Kevin Pietersen"]
}
```

在 Java 中使用 JSON

本教程将会教我们如何使用 Java 编程语言编码和解码 JSON。让我们先来准备环境以便针对 JSON 进行 Java 编程。

环境

在我们使用 Java 编码和解码 JSON 之前，我们需要安装一个可用的 JSON 模块。对于这个教程请下载和安装 JSON.simple，然后把 **jsonsimple-1.1.1.jar** 文件的路径添加到环境变量 CLASSPATH 中。

JSON 和 Java 实体映射

JSON.simple 实体映射从左侧到右侧为解码或解析，实体映射从右侧到左侧为编码。

JSON	Java
string	java.lang.String
number	java.lang.Number
true | false	java.lang.Boolean
null	null
array	java.util.List
object	java.util.Map

解码时，*java.util.List* 的默认具体类是 *org.json.simple.JSONArray*，*java.util.Map* 的默认具体类是 *org.simple.JSONObject*。

在 Java 中编码 JSON

下面这个简单的示例展示了使用 java.util.HashMap 的子类 JSONObject 编码一个 JSON 对象。这里并没有提供顺序。如果你需要严格的元素顺序，请使用 JSONValue.toJSONString(map) 方法的有序映射实现，比如 java.util.LinkedHashMap。

```
import org.json.simple.JSONObject;

class JsonEncodeDemo
{
   public static void main(String[] args)
```

```
   {
      JSONObject obj = new JSONObject();

      obj.put("name", "foo");
      obj.put("num", new Integer(100));
      obj.put("balance", new Double(1000.21));
      obj.put("is_vip", new Boolean(true));

      System.out.print(obj);
   }
}
```

编译和执行上面的程序时，会生成如下所示结果：

```
{"balance": 1000.21, "num":100, "is_vip":true, "name":"foo"}
```

下面是另一个示例，使用 Java JSONObject 展示了 JSON 对象流：

```
java
import org.json.simple.JSONObject;
class JsonEncodeDemo
{
   public static void main(String[] args)
   {
      JSONObject obj = new JSONObject();

      obj.put("name","foo");
      obj.put("num",new Integer(100));
      obj.put("balance",new Double(1000.21));
      obj.put("is_vip",new Boolean(true));

      StringWriter out = new StringWriter();
      obj.writeJSONString(out);
      String jsonText = out.toString();
      System.out.print(jsonText);
   }
}
```

编译和执行上面的程序时，会生成如下所示结果：

```
{"balance": 1000.21, "num":100, "is_vip":true, "name":"foo"}
```

在 Java 中解码 JSON

下面的例子使用了 JSONObject 和 JSONArray，其中 JSONObject 就是 java.util.Map，JSONArray 就是 java.util.List，因此我们可以使用 Map 或 List 的标准操作访问它们。

```
import org.json.simple.JSONObject;
import org.json.simple.JSONArray;
import org.json.simple.parser.ParseException;
import org.json.simple.parser.JSONParser;

class JsonDecodeDemo
{
   public static void main(String[] args)
   {
      JSONParser parser=new JSONParser();
      String s = "[0,{\"1\":{\"2\":{\"3\":{\"4\":[5,{\"6\":7}]}}}}]";
         try{
         Object obj = parser.parse(s);
         JSONArray array = (JSONArray)obj;
         System.out.println("The 2nd element of array");
         System.out.println(array.get(1));
         System.out.println();
         JSONObject obj2 = (JSONObject)array.get(1);
         System.out.println("Field \"1\"");
         System.out.println(obj2.get("1"));

         s = "{}";
         obj = parser.parse(s);
         System.out.println(obj);

         s= "[5,]";
         obj = parser.parse(s);
         System.out.println(obj);

         s= "[5,,2]";
         obj = parser.parse(s);
         System.out.println(obj);
      }catch(ParseException pe){
         System.out.println("position: " + pe.getPosition());
         System.out.println(pe);
      }
   }
}
```

编译和执行上面的程序时，会生成如下所示结果：

```
The 2nd element of array
{"1":{"2":{"3":{"4":[5,{"6":7}]}}}}

Field "1"
{"2":{"3":{"4":[5,{"6":7}]}}}
{}
[5]
[5,2]
```

JSON 与 Ajax

AJAX 就是异步 JavaScript 和 XML，它是一组用于客户端的相互关联的 Web 开发技术，以创建异步 Web 应用程序。遵循 AJAX 模型，Web 应用程序可以以异步的方式发送数据以及从服务器上检索数据，而不影响现有页面的显示行为。

许多开发人员都在客户端和服务器之间使用 JSON 传递 AJAX 更新。实时更新体育成绩的站点就可以视为一个 AJAX 例子。如果这些成绩要更新到站点上，那么必须要把它们存储到服务器上便于需要时网页能取回这些成绩。这里我们可以使用 JSON 格式的数据。

任何使用 AJAX 更新的数据都可以使用 JSON 格式存储在 Web 服务器上。使用 AJAX，那么 JavaScript 就可以在必要时取回这些 JSON 文件，解析它们，然后做以下两件事情：

- 把它们显示到网页上之前将解析的值存储到变量中便于进一步处理。
- 直接分配数据给网页中的 DOM 元素，那么它就会显示在站点上。

示例

下面的代码展示了 JSON 和 AJAX，请把它们保存为 ajax.htm 文件。这里的加载函数 loadJSON() 将会使用异步的方式上传 JSON 数据。

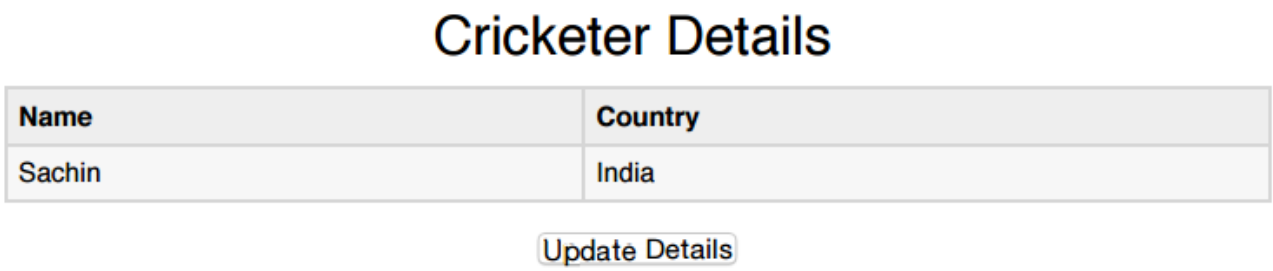
```
<head>
<meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
<script type="application/javascript">
function loadJSON()
{
   var data_file = "http://www.tutorialspoint.com/json/data.json";
   var http_request = new XMLHttpRequest();
   try{
      // Opera 8.0+, Firefox, Chrome, Safari
      http_request = new XMLHttpRequest();
   }catch (e){
      // IE 浏览器处理
      try{
         http_request = new ActiveXObject("Msxml2.XMLHTTP");
      }catch (e) {
         try{
            http_request = new ActiveXObject("Microsoft.XMLHTTP");
         }catch (e){
```

```
            // 错误处理
            alert("Your browser broke!");
            return false;
         }
      }
   }
   http_request.onreadystatechange = function(){
      if (http_request.readyState == 4 )
      {
         // 使用 JSON.parse 解析 JSON 数据
         var jsonObj = JSON.parse(http_request.responseText);
         // jsonObj 变量现在包含数组结构，可以通过 jsonObj.name 和 jsonObj.country 的方式访问
         document.getElementById("Name").innerHTML = jsonObj.name;
         document.getElementById("Country").innerHTML = jsonObj.country;
      }
   }
   http_request.open("GET", data_file, true);
   http_request.send();
}
</script>
<title>tutorialspoint.com JSON</title>
</head>
<body>
<h1>Cricketer Details</h1>
<table class="src">
<tr><th>Name</th><th>Country</th></tr>
<tr><td><div id="Name">Sachin</div></td>
<td><div id="Country">India</div></td></tr>
</table>
<div class="central">
<button type="button" onclick="loadJSON()">Update Details </button>
</body>
</html>
```

下面就是包含 JSON 格式数据的输入文件 data.json，当我们点击 Update Detail 按钮时会以异步的方式上传它。这个文件已经保存到 http://www.tutorialspoint.com/json/ 上了。

```
{"name": "brett", "country": "Australia"}
```

上面的 HTML 代码会生成如下所示屏幕显示，这里你可以进行 AJAX 实战：



图片 2.1 ajax in action

当我们点击 Update Detail 按钮时，应该会得到如下所示的结果，你也可以自己尝试 JSON 和 AJAX，提供你自己的浏览器支持的 JavaScript。

图片 2.2 ajax in action