JSON （JavaScript Object Notation）一种简单的数据格式，比xml更轻巧。 JSON 是 JavaScript 原生格式，这意味着在 JavaScript 中处理 JSON 数据不需要任何特殊的 API 或工具包。

JSON的规则很简单： 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。具体细节参考<http://www.json.org/json-zh.html>

**举个简单的例子：**

**js 代码**

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);

}

这表示一个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

**js 代码**

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);

user.username = "Tom";

alert(user.username);

}

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。

**js 代码**

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JSON字符到Object

**js 代码**

function myEval() {

var str = '{ "name": "Violet", "occupation": "character" }';

var obj = eval('(' + str + ')');

alert(obj.toJSONString());

}

或者使用parseJSON()方法

**js 代码**

function myEval() {

var str = '{ "name": "Violet", "occupation": "character" }';

var obj = str.parseJSON();

alert(obj.toJSONString());

}

下面使用prototype写一个JSON的ajax例子。

先写一个servlet (我的是servlet.ajax.JSONTest1.java)就写一句话

**java 代码**

response.getWriter().print("{ \"name\": \"Violet\", \"occupation\": \"character\" }");

再在页面中写一个ajax的请求

**js 代码**

function sendRequest() {

var url = "/MyWebApp/JSONTest1";

var mailAjax = new Ajax.Request(

url,

{

method: 'get',

onComplete: jsonResponse

}

);

}

function jsonResponse(originalRequest) {

alert(originalRequest.responseText);

var myobj = originalRequest.responseText.parseJSON();

alert(myobj.name);

}

prototype-1.5.1.js中提供了JSON的方法，String.evalJSON(), 可以不使用json.js, 修改上面的方法

**js 代码**

function jsonResponse(originalRequest) {

alert(originalRequest.responseText);

var myobj = originalRequest.responseText.evalJSON(true);

alert(myobj.name);

}

JSON还提供了java的jar包 <http://www.json.org/java/index.html> API也很简单，下面举个例子

在javascript中填加请求参数

**js 代码**

function sendRequest() {

var carr = new Car("Dodge", "Coronet R/T", 1968, "yellow");

var pars = "car=" + carr.toJSONString();

var url = "/MyWebApp/JSONTest1";

var mailAjax = new Ajax.Request(

url,

{

method: 'get',

parameters: pars,

onComplete: jsonResponse

}

);

}

使用JSON请求字符串就可以简单的生成JSONObject并进行解析,修改servlet添加JSON的处理(要使用json.jar)

**java 代码**

private void doService(HttpServletRequest request, HttpServletResponse response) throws IOException {

String s3 = request.getParameter("car");

try {

JSONObject jsonObj = new JSONObject(s3);

System.out.println(jsonObj.getString("model"));

System.out.println(jsonObj.getInt("year"));

} catch (JSONException e) {

e.printStackTrace();

}

response.getWriter().print("{ \"name\": \"Violet\", \"occupation\": \"character\" }");

}

同样可以使用JSONObject生成JSON字符串，修改servlet

**java 代码**

private void doService(HttpServletRequest request, HttpServletResponse response) throws IOException {

String s3 = request.getParameter("car");

try {

JSONObject jsonObj = new JSONObject(s3);

System.out.println(jsonObj.getString("model"));

System.out.println(jsonObj.getInt("year"));

} catch (JSONException e) {

e.printStackTrace();

}

JSONObject resultJSON = new JSONObject();

try {

resultJSON.append("name", "Violet")

.append("occupation", "developer")

.append("age", new Integer(22));

System.out.println(resultJSON.toString());

} catch (JSONException e) {

e.printStackTrace();

}

response.getWriter().print(resultJSON.toString());

}

js 代码

function jsonResponse(originalRequest) {

alert(originalRequest.responseText);

var myobj = originalRequest.responseText.evalJSON(true);

alert(myobj.name);

alert(myobj.age);

}

**参考**

* <http://www.json.org/js.html>
* <http://www.blogjava.net/Jkallen/archive/2006/03/28/37905.html>
* <http://www.json.org/>
* <http://www.prototypejs.org/learn/json>
* <http://www.json.org/java/index.html>
* <http://www.ibm.com/developerworks/cn/web/wa-ajaxintro10/index.html>

**使用JSON**

JSON也就是JavaScript Object Notation，是一个描述数据的轻量级语法。JSON的优雅是因为它是JavaScript语言的一个子集。接下来你将看到它为什么如此重要。首先，来比较一下JSON和XML语法。

JSON和XML都使用结构化方法描述数据。例如一个地址簿应用程序可以提供用来产生XML格式的地址卡的web服务：

<?xml version='1.0' encoding='UTF-8'?>

<card>

<fullname>Sean Kelly</fullname>

<org>SK Consulting</org>

<emailaddrs>

<address type='work'>kelly@5icool.org</address>

<address type='home' pref='1'>kelly@5icool.org</address>

</emailaddrs>

<telephones>

<tel type='work' pref='1'>+1 214 555 1212</tel>

<tel type='fax'>+1 214 555 1213</tel>

<tel type='mobile'>+1 214 555 1214</tel>

</telephones>

<addresses>

<address type='work' format='us'>1234 Main St

Springfield, TX 78080-1216</address>

<address type='home' format='us'>5678 Main St

Springfield, TX 78080-1316</address>

</addresses>

<urls>

<address type='work'>http://www.5icool.org/</address>

<address type='home'>http://www.5icool.org/</address>

</urls>

</card>

使用JSON, 形式如下:

{

"fullname": "Sean Kelly",

"org": "SK Consulting",

"emailaddrs": [

{"type": "work", "value": ""},

{"type": "home", "pref": 1, "value": ""}

],

"telephones": [

{"type": "work", "pref": 1, "value": "+1 214 555 1212"},

{"type": "fax", "value": "+1 214 555 1213"},

{"type": "mobile", "value": "+1 214 555 1214"}

],

"addresses": [

{"type": "work", "format": "us",

"value": "1234 Main StnSpringfield, TX 78080-1216"},

{"type": "home", "format": "us",

"value": "5678 Main StnSpringfield, TX 78080-1316"}

],

"urls": [

{"type": "work", "value": "http://www.5icool.org/"},

{"type": "home", "value": "http://www.5icool.org/"}

]

}

如你所看到的，JSON有结构化的嵌套数据元素，这一点和XML相似。JSON也是基于文本的，XML也是如此。两者都使用Unicode。 JSON和XML都很容易阅读。主观上，JSON更清晰，冗余更少。JSON WEB站点严格地描述了JSON语法，目前就是这样的。它确实是一个简单的小语言！ XML确实适合标记文档，但是JSON是数据交互的理想格式。每个JSON文档描述了一个这样一个对象，该对象包含有：嵌套对象、数组、字符串、数字、布 尔值或空值。

在这些地址卡例子代码中，JSON版本是更轻量级的，只占用了682字节的空间，而XML版本需要744字节空间。尽管这不是一个可观的节省。而实际的好处则来自解析过程。

**XML对比JSON：地位丧失**

通过使用XMLHttpRequest对象，可以从你的基于AJAX的应用程序取得XML和JSON文件。典型的，交互代码如下：

var req = new XMLHttpRequest();

req.open("GET", "http://www.5icool.org/addr?cardID=32", /\*async\*/true);

req.onreadystatechange = myHandler;

req.send(/\*no params\*/null);

作为WEB服务器响应，你提供的处理器函数（myHandler函数）被多次调用，为你提供提前终止事务，更新进度条等机会。通常的，只有在web请求完成以后才起作用：那时，你就可以使用返回的数据了。

为了处理XML版本的地址卡数据，myHandler的代码如下：

function myHandler() {

if (req.readyState == 4 /\*complete\*/) {

// Update address field in a form with first street address

var addrField = document.getElementById('addr');

var root = req.responseXML;

var addrsElem = root.getElementsByTagName('addresses')[0];

var firstAddr = addrsElem.getElementsByTagName('address')[0];

var addrText = fistAddr.firstChild;

var addrValue = addrText.nodeValue;

addrField.value = addrValue;

}

}

值得注意的是你不必解析XML文档：XMLHttpRequest对象自动地解析了，并使responseXML中的DOM树可用。通过使用 responseXML属性，可以调用getElementsByTagName方法查找文档的地址部分，你还可以使用第一个去找到它。然后，可以再次调 用getElementsByTagName在地址部分查找第一个地址元素。这就取得了文档的第一个DOM子节点，就是一个文本节点，并取得节点的值，这 就是你想要的街道地址。最后，可以在表单域中显示结果。

确实不是一个简单的工作，现在，使用JSON再试一下：

function myHandler() {

if (req.readyState == 4 /\*complete\*/) {

var addrField = document.getElementById('addr');

var card = eval('(' + req.responseText + ')');

addrField.value = card.addresses[0].value;

}

}

你所做的第一件事情就是解析JSON响应。但是，因为JSON是JavaScript的一个子集，你可以使用JavaScript自己的编译器来解析它， 通过调用eval函数。解析JSON仅需要一行！此外，操纵JSON中的对象就像操纵其他JavaScript对象一样。这显然要比通过DOM树来操纵简 单，例如：

card.addresses[0].value 是第一个街道地址, "1234 Main Stb &"

card.addresses[0].type 是地址类型, "work"

card.addresses[1] 是家庭地址对象

card.fullname 是card的名称, "Sean Kelly"

如果更仔细观察，你可能会发现XML格式中文档至少有一个跟元素，card。这在JSON里是不存在的，为什么？ 大概就是，如果你正在开发JavaScript来访问Web服务，你已经知道你想要得到的。然而，你可以在JSON中这么使用：

{"card": {"fullname": ...}}

使用这个技术，你的JSON文件总是以一个带有单一命名属性的对象开始，该属性标识了对象的种类。

**JSON是快速可靠的吗？**

JSON提供轻量的小文档，并且JSON在JavaScript更容易使用。XMLHttpRequest自动为你解析了XML文档，而你还要手工解析 JSON文件，但是解析JSON比解析XML更慢么？作者通过几千次的反复测试，使用XMLHttpRequest解析XML和解析JSON，结果是解析 JSON比XML要快10倍！当把AJAX当作桌面应用看待时，速度是最重要的因素，很明显，JSON更优秀。

当然，你不能总是控制服务器端来为AJAX程序产生数据。你还可以使用第三方服务器代替服务器提供XML格式的输出。并且，如果服务器恰好提供JSON，你可以确定你真的想使用它吗?

代码中值得注意的是，你将响应文本直接传入到eval中。如果你控制着服务器，就可以这么做。如果不是，一个恶意服务器可以使你的浏览器执行危险操作。在这样的情况下，你最好使用写在JavaScript中的代码来解析JSON。幸运地，这已经有了。

说到解析，Python爱好者可能注意到JSON不只是JavaScript的子集，它还是Python的一个子集。你可以在Python中直接执行JSON，或者使用安全JSON解析代替。JSON.org网站列举了许多常用JSON解析器

。

**服务器端的JSON**

到现在为止，你或许将焦点注意在运行在客户浏览器中的基于AJAX的web应用程序使用JSON。自然地，首先，JSON格式的数据必须在服务器端产生。 幸运地是，创建JSON或将其他存在的数据转换成JSON是相当简单的。一些WEB应用程序框架，例如TurboGears，自动包括对JSON输出的支 持。

此外商业WEB服务提供商也注意到了JSON。Yahoo最近创建了许多基于JSON的web服务。Yahoo的多种搜索服务，履行计 划，del.icio.us，还有高速公路交通服务也都支持JSON输出。毫无疑问，其他主要WEB服务提供商也将加入到对JSON的支持中。

**总结**

JSON的聪明在于它是JavaScript和Python的子集，使得它更易用，为AJAX提供高效的数据交互。它解析更快，比 XML更易使用。JSON正成为现在“Web 2.0”的最强音。每个开发者，无论是标准桌面应用程序或Web应用程序，越来越注意到了它的简单和便捷。我希望你能体会到在buzzword- compliant, Web-2.0-based, AJAX-enabled, 敏捷开发中应用到JSON的乐趣。