第11章 JSON

如果阅读过第 3 章应该知道，向服务器发出 Ajax 请求时，可以以两种不同的方式从服务器响应中检索数据。一种是使用 XMLHttpRequest 对象的 responseXML 属性访问 XML 格式的数据；一种是使用 XMLHttpRequest 对象的 responseText 属性访问字符串格式的数据。当前，XML 是进行数据传输的标准语言，但是使用 XML 的缺点之一是很难对它进行解析并提取要添加到页面的数据。

Douglas Crockford 创建了另一种数据传输格式，我们称之为 JavaScript 对象表示法(JSON)。使用 JSON 进行数据传输的优势之一是 JSON 实际上就是 JavaScript，它是基于 ECMAScript 第 3 版中 JavaScript 对象字面量语法子集的一种文本格式。这表示可以使用 responseText 从服务器中检索 JSON 数据，然后再使用 JavaScript 的 eval()方法将 JSON 字符串转换为 JavaScript 对象。那么，使用附加 JavaScript 就可以很容易地从该对象中提取数据，而不需要处理 DOM。

另外，也有针对大部分编程语言(包括 C++、C#、ColdFusion、Java、Perl、PHP 和 Python)的 JSON 库。这些库能将上述语言格式化数据转换成 JSON 格式。更多详细内容，请参见 JSON 站点：www.json.org。

本章将详细介绍 JSON 的使用，包括 JSON 的基本语法、使用 JSON 和 XML 进行数据传输的比较和使用 XMLHttpRequest 对象检索并提取 JSON 数据的详细过程。

本章主要内容：

- JSON 语法(包括对象字面量、数组字面量和数据类型)。
- 数据传输格式(可读性、速度和数据提取)。
- 使用 JSON 数据(包括创建对 JSON 数据的 Ajax 请求，解析服务器响应，使用 JSON 解析器，在 Web 页面上显示响应的数据)。
- 在 PHP 中使用 JSON。

11.1 JSON 语法

JSON 由两个数据结构组成：

- **对象**——名/值对的无序集合。
- **数组**——值的有序集合。

JSON 没有变量或其他控制结构。JSON 只用于数据传输。

JSON 语法是基于对象字面量和数组字面量的 JavaScript 语法。当使用字面量时，将包括数据本身，但不包括生成数据的表达式。例如，在下列代码块中，赋予变量 x 的值是字面量(15)，而赋予变量 y 的值是表达式(3*x)，该表达式必须在赋值给 y 之前被求值。

```
var x = 15;
var y = 3 * x;
```

11.1.1 数据类型

JSON 数据结构包含下列数据类型：

- 字符串
- 数字
- 布尔值(true/false)
- null
- 对象
- 数组

JSON 字符串必须使用双引号括起来。它们使用标准的 JavaScript 转义序列。因此在下列字符的前面要添加一个反斜线：

- " (引号)
- b (空格)
- n (新行)
- f (换页)
- r (回车)
- t (水平定位)
- u (为 Unicode 字符增加 4 个数位)
- \ (反斜线符号)
- / (正斜杠符号)

例如，若要表示字符串 I feel "funny"，必须对单词 funny 外的双引号加反斜线：

```
"I feel \"funny\""
```

11.1.2 对象字面量

使用对象构造函数或对象字面量可以定义JavaScript对象。若要使用对象构造函数定义新对象，可以使用具有new关键字的Object构造函数。

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

然后，使用点符号为对象添加属性。

```
member.name = "Jobo";
member.address = "325 Smith Rd";
member.isRegistered = true;
```

也可以使用数组语法为对象添加属性。

```
member["name"] = "Jobo";
member["address"] = "325 Smith Rd";
member["isRegistered"] = true;
```

通过使用对象字面量，可以更有效率地创建同一对象。

```
var member =
{name: "Jobo",
address: "325 Smith Road",
isRegistered: true
};
```

JSON 不使用构造函数，只使用字面量。下列代码显示 JSON 文本形式的 member 对象：

```
{"name": "Jobo",
 "address": "325 Smith Road",
 "isRegistered": true
}
```

说明：

以上 JSON 对象定义的结尾没有分号。

如果有一个以上的成员对象，则在 JSON 中，可以将其表示为一个包含拥有两个对象的数组的对象。

```
{"member": [
{"name": "Jobo",
 "address": "325 Smith Road",
 "isRegistered": true
},
{"name": "Rico",
 "address": "30 Ocean Drive",
 "isRegistered": false
}
]
}
```

注意：

有关 JavaScript 对象的更多详细信息，请参见第 2 章。

11.1.3　数组字面量

使用构造函数或数组字面量还可以创建JavaScript数组。若要使用构造函数定义新数组，可以使用具有new关键字的Array构造函数。

```
var myArray = new Array();
```

然后，可以使用方括号和表示位置的索引值将成员添加到数组中。

```
myArray[0] = 1218;
myArray[1] = "Crawford"
myArray[2] = "Drive";
```

使用数组字面量创建相同的对象更有效率。

```
var myArray = [1218, "Crawford", "Drive"];
```

下列代码显示 JSON 文本形式的数组：

```
[1218, "Crawford", "Drive"]
```

11.1.4 使用 JSON 解析器

可以使用 JSON 解析器从对象和数组中创建 JSON 文本或者从 JSON 文本中创建对象和数组。JSON 站点 www.json.org/json.js 上提供有 JSON 解析器。通过将下列代码加入到页面的头部分中，可以复制这一 JavaScript 文件并在页面中引用它：

```
<script type="text/javascript" src="json.js"></script>
```

上述代码定义了 2 个函数：

- toJSONString()
- parseJSON()

其中，toJSONString()方法被添加到JavaScript Object和Array定义中，该方法能将JavaScript对象或数组转换成JSON文本。不必将对象或数组转换为字面量就能使用该方法。例如，下面的成员对象代码使用了Object构造函数：

```
<script type="text/javascript">
var member = new Object();
member.name = "Jobo";
member.address = "325 Smith Rd";
member.isRegistered = true;
member = member.toJSONString();
alert("The member object as a JSON data structure: \n" + member);
</script>
```

在 IE 7 的警告对话框中，显示了该转换结果，如图 11-1 所示。

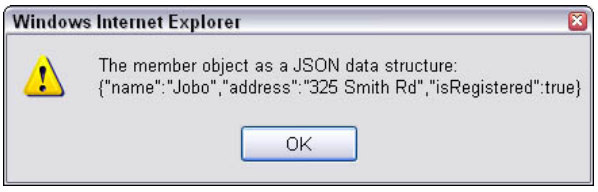

图 11-1 IE 7 中的警告对话框

将parseJSON()方法添加到JavaScript String定义中。该方法能从JSON文本中创建对象或数组。在本章后面的“Ajax和JSON”小节中将会使用该方法。

11.2　数据传输格式

在 Ajax 应用程序中，JSON 或 XML 都可以用作数据传输的格式。选择数据传输格式的关键是看要传输的数据类型。XML 的结构要比 JSON 的复杂得多。几乎所有的数据类型都可以使用 XML 格式进行传输。现在，很多桌面应用程序(例如，Microsoft Word、Excel 和 Access)都允许将数据作为 XML 导入和导出；然而，JSON 的简单数据结构肯定是 Ajax 应用程序中传输数据所需要的。使用 JSON 具有以下几种优势。

- JSON就是 JavaScript—— 使用JavaScript的eval()方法可以很容易地将 JSON转变为JavaScript对象和数组。然后，可以使用JavaScript从服务器响应中提取数据。如果我们已经熟练使用JavaScript，那么使用JSON就会非常容易。
- JSON是类型化的—— JSON对象已经具有JavaScript数据类型：字符串、数字、Boolean、null、数组或对象。通过使用DTD或XML 模式可以囊括XML内容的数据类型，以定义XML文档的结构，不过，JavaScript数据类型不是XML的内置功能。
- JSON 可以解析为 JavaScript——JSON 以文本格式表示 JavaScript。使用 JavaScript 本身，可以将 JSON 文本转换成 JavaScript。为增加安全，可以对该转换使用 JSON 解析器。返回作为 XML 的数据就意味着我们需要解析 XML。通常，这需要 DOM 方法和 DOM 操作。如前几章所介绍，这会变得非常复杂，即使是简单数据。

人和机器可以很容易地读取 XML 和 JSON。下列代码显示的是第 5 章中的 XML 文件 classes.xml。

```
<?xml version="1.0"?>
<classes>
  <class>
    <classID>CS115</classID>
    <department>ComputerScience</department>
    <credits req="yes">3</credits>
    <instructor>Adams</instructor>
    <title>Programming Concepts</title>
  </class>
  <class>
    <classID semester="fall">CS205</classID>
    <department>ComputerScience</department>
    <credits req="yes">3</credits>
    <instructor>Dykes</instructor>
    <title>JavaScript</title>
  </class>
  <class>
    <classID semester="fall">CS255</classID>
    <department>ComputerScience</department>
    <credits req="yes">3</credits>
    <instructor>Brunner</instructor>
    <title>Java</title>
  </class>
```

```
</classes>
```

下列代码是以 JSON 格式显示的相同数据(classes.txt)。

```
{"class1": [
   {
    "classID": "CS115",
    "department": "Computer Science",
    "credits": 3,
    "req": "yes",
    "instructor" : "Adams",
    "title" : "Programming Concepts"
   },
   {
    "classID": "CS205",
    "semester": "fall",
    "department": "Computer Science",
    "credits": 3,
    "req": "yes",
    "instructor" : "Dykes",
    "title" : "JavaScript"
   },
   {
    "classID": "CS255",
    "semester": "fall",
    "department": "Computer Science",
    "credits": 3,
    "req": "yes",
    "instructor" : "Brunner",
    "title" : "Java"
    }
   ]
}
```

说明：

因为在 JavaScript 中，class 是一个保留字，所以上述代码中的对象名称已从 class 改为 class 1。而对于 JavaScript 语言的未来扩充，保留字将被取消。

XML 使用元素、属性、实体和其他结构。JSON 不是文档格式，因此它不需要这些附加结构。因为 JSON 数据只包括“名-值”对(对象)或值(数组)，所以 JSON 数据比同等的 XML 数据占用更少的空间。例如，classes.xml 占用 690 字节，而 classes.txt 占用 647 字节。虽然在这里看来是微不足道的差异，但在大型应用中，XML 和 JSON 数据之间的字节差异可能导致明显的速度差异。

说明：

有关 XML 和 JSON 之间比较的更多详细资料，请参见 www.json.org/xml.html 上的 *JSON: The Fat-Free Alternative to XML* 一文。

11.3　Ajax 和 JSON

与创建文本数据的 Ajax 请求一样，可以使用 XMLHttpRequest 对象创建 JSON 数据的请求。

11.3.1　创建请求

如果直接请求服务器上一个 JSON 文件中的 JSON 数据(换句话说，并非使用服务器端语言(例如，PHP)作为媒介获得数据)，则可以利用文件名来请求 JSON 文件。

```
request.open("GET", "classes.txt", true);
```

在这种情况下，classes.txt 是 JSON 数据文件的名称，request 是创建用来存放 XMLHttpRequest对象的变量。

图 11-2 显示了 Firebug 控制台中的服务器响应。如图 11-3 所示，响应的内容类型是 text/plain。

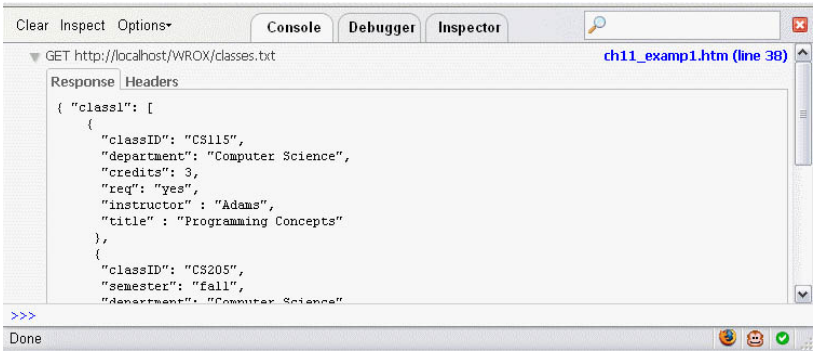

图 11-2　Firebug 控制台中的服务器响应

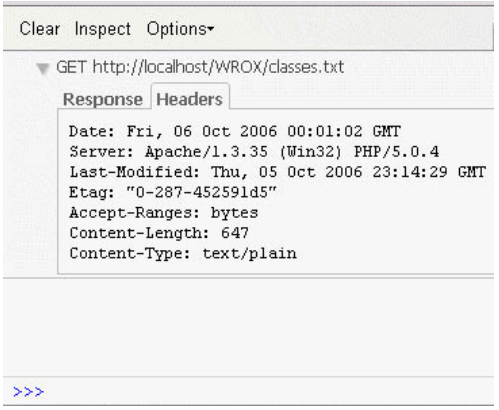

图 11-3　响应的内容类型是 text/plain

说明：

有关使用 Firefox 的 Firebug 插件的更多信息，请参见第 6 章。

如果直接从服务器上的 XML 文件请求 XML 数据，则同样可以利用文件名来请求 XML 文件。

```
request.open("GET", "classes.xml", true);
```

图 11-4 显示了 Firebug 控制台中的服务器响应。如图 11-5 所示，因为使用的是 Apache Web 服务器，所以该响应的内容类型是 application/xml。如果 web 服务器是 IIS，则该响应的内容类型是 text/xml。

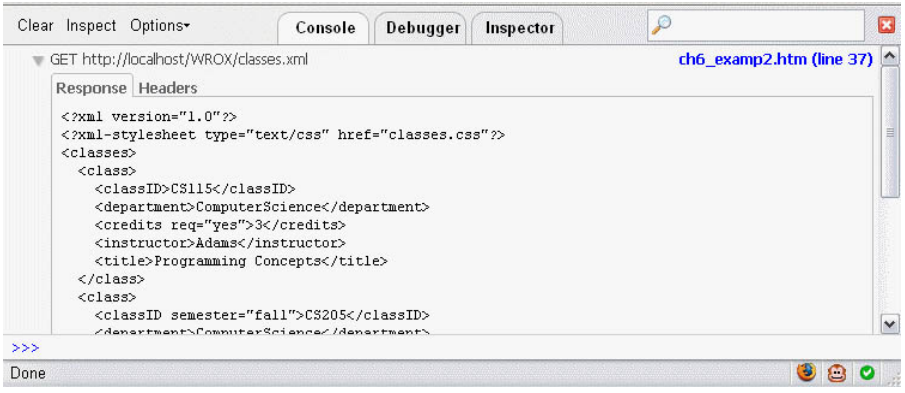

图 11-4　Firebug 控制台中的服务器响应再次出现

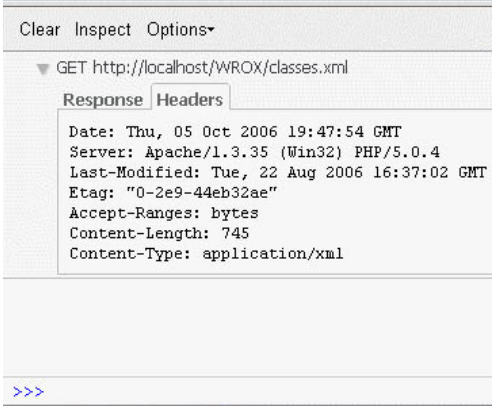

图 11-5　该响应的内容类型是 application/xml

11.3.2　解析响应

一旦接收服务器中的 JSON 数据，就可以采用两种不同的方式解析该响应。可以使用 JavaScript 的内置函数 eval()，或者为了进一步的安全，使用 JSON 解析器代替。

1. 使用 eval()

eval()方法可以把 JavaScript 字符串当作参数，还可以将该字符串转换成对象，或作为

命令运行。例如，使用 eval()创建可以存放当前日期和时间的新变量。

```
eval("var myDate = new Date();");
```

如果使用 XMLHttpRequest 对象的 responseText 属性请求 JSON 数据，那么使用 eval()将 JSON 文本字符串转换成 JavaScript 对象。

```
var jsonResp = request.responseText;
jsonResp = eval("(" + jsonResp + ")");
```

注意：

因为JSON字符串常包含花括号(例如，JavaScript的for或if语句使用这些花括号)，所以我们用圆括号来括住JSON字符串，以表明它是一个求值表达式，而不是一个要运行的命令。

2. 使用 parseJSON()

如果 Web 服务器既提供 JSON 数据也提供请求页面，则适合选用 eval()方法。如果涉及安全，则适合使用 JSON 解析器。JSON 解析器只作用于 JSON 文本，并且不执行其他 JavaScript。

在这种情况下，可以使用 responseText，但要使用 parseJSON()方法将 JSON 文本字符串转换成 JavaScript 对象。

```
var jsonResp = request.responseText;
jsonResp = jsonResp.parseJSON();
```

为了访问parseJSON()函数，还需要添加引用json.js文件的script标记。

```
<script type="text/javascript" src="json.js"></script>
```

11.3.3 将 JSON 数据添加到页面

一旦将 JSON 数据转换为 JavaScript 对象，就可以使用 JavaScript 从该对象中提取数据。例如，如果变量 jsonResp 包含 JavaScript 数组，则可以使用 for 循环遍历数组的成员。

```
for (i=0; i < jsonResp.class1.length; i++) {
...
```

不需要使用 DOM 从响应中提取数据，不过可能要使用 DOM 方法(例如 createElement())将响应数据动态地添加到页面，如下面的示例所示。

试一试 为 JSON 数据创建 XMLHttpRequest

在本示例中，将为 classes.txt 文件(classes.xml 文件的 JSON 数据版本)创建并打开一个 XMLHttpRequest。然后，使用 DOM 方法将该数据动态地添加到页面并显示。将下列代码输入文本编辑器，并将其保存为 ch11_examp1.htm。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Checking Courses</title>
<script type="text/javascript">
  function getDoc()
  {
    var request;
    if (window.XMLHttpRequest) {
      request = new XMLHttpRequest();
    }
    else if (window.ActiveXObject) {
      request = new ActiveXObject("Microsoft.XMLHTTP");
    }
    if(request) {
      request.open("GET", "classes.txt", true);
      request.onreadystatechange = function()

      if ((request.readyState == 4) && (request.status == 200)) {
        var jsonResp = request.responseText;
        jsonResp = eval("(" + jsonResp + ")");
        findClass(jsonResp);
      }

    }

    request.send(null);
  }

    function findClass(jsonResp) {
      for (i=0; i < jsonResp.class1.length; i++) {
        var title = jsonResp.class1[i].title;
        var req = jsonResp.class1[i].req;
        var myEl = document.createElement('p');
        var newText = title + " is the name of a course in the Computer Science
department.";
        var myTx = document.createTextNode(newText);
        myEl.appendChild(myTx);
        var course = document.getElementById('title');
        course.appendChild(myEl);

          if (req == 'yes') {
            var addlText = " This is a required course.";
            var addlText2 = document.createTextNode(addlText);
            myEl.appendChild(addlText2);
          }

          else {
            var addlText = " This is not a required course.";
            var addlText2 = document.createTextNode(addlText);
            myEl.appendChild(addlText2);
          }
```

```
        }

      }
</script>
</head>
<body>

<h1>Checking courses</h1>
  <form>
    <input type = "button" id="reqDoc" value = "Check courses" />
  </form>
<script type="text/javascript">
var myDoc = document.getElementById('reqDoc');
myDoc.onclick = getDoc;
</script>
<div id="title"></div>
</body>
</html>
```

要正常运行，必须将 ch11_examp1.htm 和 classes.txt 这两个文件置于运行 Ajax 请求的服务器上。将这两个文件复制到 Web 服务器的根文件夹中 BegAjax 文件夹下的 Chapter11 子文件夹(例如，如果使用 IIS，则根文件夹是 wwwroot 文件夹；如果使用 Apache，则根文件夹是 htdocs 文件夹)。

示例的说明

该页面包含一个窗体，该窗体内有一个标签为 Check Courses 的按钮。当用户单击该按钮时，getDoc()函数被调用。

```
<form>
<input type = "button" id="reqDoc" value = "Make request">
</form>
```

采用传统事件注册来注册单击事件，而不是在 input 标记中加入 onclick 属性。

```
<script type="text/javascript">
var myDoc = document.getElementById('reqDoc');
myDoc.onclick = getDoc;
</script>
```

getDoc()函数用一个if语句来检验浏览器是否直接支持XMLHttpRequest。如果支持，则在名为request的变量中创建并存储一个新的XMLHttpRequest对象。

```
    if (window.XMLHttpRequest) {
      request = new XMLHttpRequest();
    }
```

如果浏览器是IE 5 或IE 6，则在request变量中创建并存储新的ActiveXObject。

```
      else if (window.ActiveXObject) {
        request = new ActiveXObject("Microsoft.XMLHTTP");
```

```
        }
```

如果已经创建对象，则使用XMLHttpRequest对象的open方法来请求JSON文档classes.txt中的数据。该方法有 3 个参数：用于请求的HTTP方法("GET")、文档的URL("classes.txt")和指出调用为异步的布尔值true。

```
    if(request) {
      request.open("GET", "classes.txt", true);
```

下面声明 request 变量，即使它最初没有赋值。在表达式中使用未声明的变量会导致错误。

```
var request;
```

当 XMLHttpRequest 对象的 readyState 属性发生改变时(在数据开始下载时)，调用 anonymous 函数。

```
      request.onreadystatechange = function()
        {
        if ((request.readyState == 4) && (request.status == 200)) {
```

如果 readyState 属性等于 4，则下载完成。作为附加检查，status 属性等于 200 也表示请求被成功处理。

```
var jsonResp = request.responseText;
jsonResp = eval("(" + jsonResp + ")");
findClass(jsonResp);
```

一旦下载完成，responseText 属性就在名为 jsonResp 的变量中存储 JSON 数据。接着，eval()方法将 JSON 数据转换成 JavaScript 对象。该对象被当作参数发送给 findClass()函数。

```
function findClass(jsonResp) {
      for (i=0; i < jsonResp.class1.length; i++) {
```

findClass()函数使用 for 循环，该 for 循环会遍历 class1 数组，直到提取完该数组中所有成员的数据为止。该数组的长度被用在 for 循环的测试条件中：

```
i < jsonResp.class1.length;
```

从该数组中提取 title 和 req 数据。

```
        var title = jsonResp.class1[i].title;
        var req = jsonResp.class1[i].req;
```

使用 createElement、createTextNode 和 appendChild 将 title 的值添加到现有页面中。

```
     var myEl = document.createElement('p');
     var newText = title + " is the name of a course in the Computer Science
department.";
```

```
        var myTx = document.createTextNode(newText);
        myEl.appendChild(myTx);
        var course = document.getElementById('title');
        course.appendChild(myEl);
```

用一个为 id 赋值 title 的表达式将该值添加到 div 元素中，该 id 值由 getElementById('title') 标识。

```
    <div id="title"></div>
```

在 if/else 语句中使用 req 的值，并将附加文本节点添加到 myE1 元素中。

```
          if (req == 'yes') {
            var addlText = " This is a required course.";
            var addlText2 = document.createTextNode(addlText);
            myEl.appendChild(addlText2);
          }
          else {
        var addlText = " This is not a required course.";
            var addlText2 = document.createTextNode(addlText);
            myEl.appendChild(addlText2);
          }
```

如果使用的是 JSON 解析器，而不是 eval()，则只需要改变该代码中的两行代码即可。在页面的头部添加下列代码行，以便能访问 parseJSON()函数。同时，务必将 json.js 文件复制到与 ch11_examp1.htm 同样的位置。

```
    <script type="text/javascript" src="json.js"></script>
```

使用下列代码代替调用 eval()方法的代码行：

```
    jsonResp = jsonResp.parseJSON();
```

其他代码则完全相同，并且该代码以同样的方式运行。

11.4　在 PHP 中使用 JSON

很多服务器端架构都可以使用 JSON，其中有 PHP、Java、C#、Ruby、Python、Perl 和 ColdFusion。可参见 JSON 主页 www.json.org 上的完整列表。

JSON-PHP 是能使 PHP 处理 JSON 数据的 PHP 库。可以在站点 http://mike.teczno.com/JSON/JSON.phps 免费下载它。若要将 JSON-PHP 用于 PHP 文件，请执行下列这些步骤。

(1) 复制 JSON.phps 文件，并将其保存为 JSON.php。

(2) 通过在 PHP 文件中添加下列代码行，确定加载该文件，并使 PHP 页面可以使用该文件：

```
require_once('JSON.php');
```

(3) 创建 Services_JSON()类的新实例，并将其赋予 PHP 变量。在 JSON.php 中定义 Services_JSON 类。

```
$myJSON = new Services_JSON();
```

(4) 使用 encode()方法将 PHP 对象转换成 JSON 格式。

```
$response = $myJSON -> encode($response);
```

(5) 将 JSON 数据发送给客户。

```
print($response);
```

例如，下列 PHP 代码(array.php)创建了具有三个成员的 PHP 数组，然后，将数组数据作为 JSON 发送到发出请求的浏览器：

```
<?php
require_once('JSON.php');
$myJSON = new Services_JSON();
$av1 = array(1, 3, 'x');
$response = $myJSON-> encode($av1);
echo ($response);
?>
```

使用下列代码对该 JSON 数据提出 Ajax 请求：

```
request.open("GET", "array.php", true);
```

通过具有 id 值为 display 的 div，使用 innerHTML 在页面上显示服务器响应。

```
    var jsonResp = request.responseText;
    jsonResp = eval("(" + jsonResp + ")");
    var display1 = document.getElementById('display');
    display1.innerHTML = jsonResp;
```

图 11-6 显示了 Firebug 控制台中的响应。

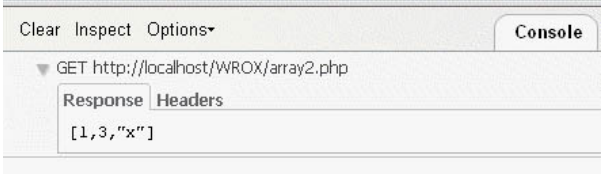

图 11-6　Firebug 控制台中的响应

有关使用 JSON-PHP 的更多信息，请参见 http://mike.teczno.com/JSON/doc/中的文件。

11.5 本章小结

JSON 是可以用来进行数据传输的 JavaScript 对象和数组字面量的子集。JSON 要比 XML 简单。因为 JSON 的文件大小要比同等的 XML 文档要小，所以它可以加快执行速度。

本章主要有以下几各方面。

- JSON 是使用两个结构进行数据传输的文本格式：包含名-值对的 JavaScript 对象字面量和包含值的 JavaScript 数组字面量。
- 与 XML 不同，JSON 只能用来传输数据，而不能用作文档格式。
- Douglas Crockford 的 JSON 解析器就是 JavaScript 文件(json.js)，该 JavaScript 文件(json.js)可以将方法添加到 JavaScript，以在 JSON 数据和 JavaScript 对象之间转换。
- responseText 属性可以用来返回 Ajax 请求中的 JSON 数据。
- 使用 eval()或 parseJSON()将字符串转换成 JavaScript 对象，就可以解析 JSON 数据。

在第 12 章中，我们将使用 Ajax 创建一个实际的 Ajax 应用程序——一个使用 MySQL 数据库和 PHP 创建可排序列表。

11.6 练习

下列问题的参考答案可以在附录 A 中找到。

1. 将下列 JavaScript 对象定义转换成对象字面量：

```
var myObject = new Object();
myObject.name = "Cessna";
myObject.model = "152";
myObject.year = "1984";
myObject.color1 = "white";
myObject.color2 = "blue";
```

2. 使用 JSON 解析器 json.js，将练习 1 中的 JavaScript 对象转换为 JSON 格式。