**第32章 JSON**

学习要点：

1. JSON语法
2. 解析和序列化

前两章我们探讨了XML的结构化数据，但开发人员还是觉得这种微型的数据结构还是过于烦琐、冗长。为了解决这个问题，JSON的结构化数据出现了。JSON是JavaScript的一个严格的子集，利用JavaScript中的一些模式来表示结构化数据。

1. **JSON语法**

JSON和XML类型，都是一种结构化的数据表示方式。所以，JSON并不是JavaScript独有的数据格式，其他很多语言都可以对JSON进行解析和序列化。

JSON的语法可以表示三种类型的值：

1. 简单值：可以在JSON中表示字符串、数值、布尔值和null。但JSON不支持JavaScript中的特殊值undefined。
2. 对象：顾名思义。
3. 数组：顾名思义。

**简单值**

1. "Lee" 这两个量就是JSON的表示方法，一个是JSON数值，一个是JSON字符串。布尔值和null也是有效的形式。但实际运用中要结合对象或数组。

**对象**

JavaScript对象字面量表示法：

var box = {

name : 'Lee',

age : 100

};

而JSON中的对象表示法需要加上双引号，并且不存在赋值运算和分号：

{

"name" : "Lee", //使用双引号，否则转换会出错

"age" : 100

}

**数组**

JavaScript数组字面量表示法：

var box = [100, 'Lee', true];

而JSON中的数组表示法同样没有变量赋值和分号：

[100, "Lee", true]

一般比较常用的一种复杂形式是数组结合对象的形式：

[

{

"title" : "a",

"num" : 1

},

{

"title" : "b",

"num" : 2

},

{

"title" : "c",

"num" : 3

}

]

PS：一般情况下，我们可以把JSON结构数据保存到一个文本文件里，然后通过XMLHttpRequest对象去加载它，得到这串结构数据字符串(XMLHttpRequest对象将在Aajx章节中详细探讨)。所以，我们可以模拟这种过程。

模拟加载JSON文本文件的数据，并且赋值给变量。

var box = '[{"name" : "a","age" : 1},{"name" : "b","age" : 2}]';

PS；上面这短代码模拟了var box = load('demo.json');赋值过程。因为通过load加载的文本文件，不管内容是什么，都必须是字符串。所以两边要加上双引号。

其实JSON就是比普通数组多了两边的双引号，普通数组如下：

var box = [{name : 'a', age : 1},{name : 'b', age : 2}];

**二．解析和序列化**

如果是载入的JSON文件，我们需要对其进行使用，那么就必须对JSON字符串解析成原生的JavaScript值。当然，如果是原生的JavaScript对象或数组，也可以转换成JSON字符串。

对于讲JSON字符串解析为JavaScript原生值，早期采用的是eval()函数。但这种方法既不安全，可能会执行一些恶意代码。

var box = '[{"name" : "a","age" : 1},{"name" : "b","age" : 2}]';

alert(box); //JSON字符串

var json = eval(box); //使用eval()函数解析

alert(json); //得到JavaScript原生值

ECMAScript5对解析JSON的行为进行规范，定义了全局对象JSON。支持这个对象的浏览器有IE8+、Firefox3.5+、Safari4+、Chrome和Opera10.5+。不支持的浏览器也可以通过一个开源库json.js来模拟执行。JSON对象提供了两个方法，一个是将原生JavaScript值转换为JSON字符串：stringify()；另一个是将JSON字符串转换为JavaScript原生值：parse()。

var box = '[{"name" : "a","age" : 1},{"name" : "b","age" : 2}]'; //特别注意，键要用双引号

alert(box);

var json = JSON.parse(box); //不是双引号，会报错

alert(json);

var box = [{name : 'a', age : 1},{name : 'b', age : 2}]; //JavaScript原生值

var json = JSON.stringify(box); //转换成JSON字符串

alert(json); //自动双引号

在序列化JSON的过程中，stringify()方法还提供了第二个参数。第一个参数可以是一个数组，也可以是一个函数，用于过滤结果。第二个参数则表示是否在JSON字符串中保留缩进。

var box = [{name : 'a', age : 1, height : 177},{name : 'b', age : 2, height : 188}];

var json = JSON.stringify(box, ['name', 'age'], 4);

alert(json);

PS：如果不需要保留缩进，则不填即可；如果不需要过滤结果，但又要保留缩进，则讲过滤结果的参数设置为null。如果采用函数，可以进行复杂的过滤。

var box = [{name : 'a', age : 1, height : 177},{name : 'b', age : 2, height : 188}];

var json = JSON.stringify(box, function (key, value) {

switch (key) {

case 'name' :

return 'Mr. ' + value;

case 'age' :

return value + '岁';

default :

return value;

}

}, 4);

alert(json);

PS：保留缩进除了是普通的数字，也可以是字符。

还有一种方法可以自定义过滤一些数据，使用toJSON()方法，可以将某一组对象里指定返回某个值。

var box = [{name : 'a', age : 1, height : 177, toJSON : function () {

return this.name;

}},{name : 'b',age : 2, height : 188, toJSON : function () {

return this.name;

}}];

var json = JSON.stringify(box);

alert(json);

PS：由此可见序列化也有执行顺序，首先先执行toJSON()方法；如果应用了第二个过滤参数，则执行这个方法；然后执行序列化过程，比如将键值对组成合法的JSON字符串，比如加上双引号。如果提供了缩进，再执行缩进操作。

解析JSON字符串方法parse()也可以接受第二个参数，这样可以在还原出JavaScript值的时候替换成自己想要的值。

var box = '[{"name" : "a","age" : 1},{"name" : "b","age" : 2}]';

var json = JSON.parse(box, function (key, value) {

if (key == 'name') {

return 'Mr. ' + value;

} else {

return value;

}

});

alert(json[0].name);