第 30 章 XML

学习要点：

1.IE 中的 XML
2.DOM2 中的 XML
3.跨浏览器处理 XML

主讲教师：李炎恢
合作网站：http://www.ibeifeng.com
讲师博客：http://hi.baidu.com/李炎恢

随着互联网的发展，Web 应用程序的丰富，开发人员越来越希望能够使用客户端来操作 XML 技术。而 XML 技术一度成为存储和传输结构化数据的标准。所以，本章就详细探讨一下 JavaScript 中使用 XML 的技术。

对于什么是 XML，干什么用的，这里就不在赘述了，在以往的 XHTML 或 PHP 课程都有涉及到，可以理解成一个微型的结构化的数据库，保存一些小型数据用的。

一. IE 中的 XML

在统一的正式规范出来以前，浏览器对于 XML 的解决方案各不相同。DOM2 级提出了动态创建 XML DOM 规范，DOM3 进一步增强了 XML DOM。所以，在不同的浏览器实现 XML 的处理是一件比较麻烦的事情。

1.创建 XMLDOM 对象

IE 浏览器是第一个原生支持 XML 的浏览器，而它是通过 ActiveX 对象实现的。这个对象，只有 IE 有，一般是 IE9 之前采用。微软当年为了开发人员方便的处理 XML，创建了 MSXML 库，但却没有让 Web 开发人员通过浏览器访问相同的对象。

```
var xmlDom = new ActiveXObject('MSXML2.DOMDocument');
```

ActiveXObject 类型

XML 版本字符串	说明
Microsoft.XmlDom	最初随同 IE 发布，不建议使用
MSXML2.DOMDocument	脚本处理而更新的版本，仅在特殊情况作为备份用
MSXML2.DOMDocument.3.0	在 JavaScript 中使用，这是最低的建议版本
MSXML2.DOMDocument.4.0	脚本处理时并不可靠，使用这个版本导致安全警告
MSXML2.DOMDocument.5.0	脚本处理时并不可靠，使用这个版本导致安全警告
MSXML2.DOMDocument.6.0	脚本能够可靠处理的最新版本

PS：在这六个版本中微软只推荐三种：

1.MSXML2.DOMDocument.6.0 最可靠最新的版本
2.MSXML2.DOMDocument.3.0 兼容性较好的版本

3.MSXML2.DOMDocument　　仅针对 IE5.5 之前的版本

PS：这三个版本在不同的 windows 平台和浏览器下会有不同的支持，那么为了实现兼容，我们应该考虑这样操作：从 6.0->3.0->备用版本这条路线进行实现。

```
function createXMLDOM() {
    var version = [
                                'MSXML2.DOMDocument.6.0',
                                'MSXML2.DOMDocument.3.0',
                                'MSXML2.DOMDocument'
    ];
    for (var i = 0; i < version.length; i ++) {
        try {
            var xmlDom = new ActiveXObject(version[i]);
            return xmlDom;
        } catch (e) {
            //跳过
        }
    }
    throw new Error('您的系统或浏览器不支持 MSXML！');      //循环后抛出错误
}
```

2.载入 XML

如果已经获取了 XMLDOM 对象，那么可以使用 loadXML()和 load()这两个方法可以分别载入 XML 字符串或 XML 文件。

```
xmlDom.loadXML('<root version="1.0"><user>Lee</user></root>');
alert(xmlDom.xml);
```

PS：loadXML 参数直接就是 XML 字符串，如果想效果更好，可以添加换行符\n。.xml 属性可以序列化 XML，获取整个 XML 字符串。

```
xmlDom.load('test.xml');                    //载入一个 XML 文件
alert(xmlDom.xml);
```

当你已经可以加载了 XML，那么你就可以用之前学习的 DOM 来获取 XML 数据，比如标签内的某个文本。

```
var user = xmlDom.getElementsByTagName('user')[0];      //获取<user>节点
alert(user.tagName);                                    //获取<user>元素标签
alert(user.firstChild.nodeValue);                       //获取<user>里的值 Lee
```

DOM 不单单可以获取 XML 节点，也可以创建。

```
var email= xmlDom.createElement('email');
xmlDom.documentElement.appendChild(email);
```

3.同步及异步

load()方法是用于服务器端载入 XML 的，并且限制在同一台服务器上的 XML 文件。那

么在载入的时候有两种模式：同步和异步。

所谓同步：就是在加载 XML 完成之前，代码不会继续执行，直到完全加载了 XML 再返回。好处就是简单方便、坏处就是如果加载的数据停止响应或延迟太久，浏览器会一直堵塞从而造成假死状态。

```
xmlDom.async = false;                    //设置同步，false，可以用 PHP 测试假死
```

所谓异步：就是在加载 XML 时，JavaScript 会把任务丢给浏览器内部后台去处理，不会造成堵塞，但要配合 readystatechange 事件使用，所以，通常我们都使用异步方式。

```
xmlDom.async = true;                     //设置异步，默认
```

通过异步加载，我们发现获取不到 XML 的信息。原因是，它并没有完全加载 XML 就返回了，也就是说，在浏览器内部加载一点，返回一点，加载一点，返回一点。这个时候，我们需要判断是否完全加载，并且可以使用了，再进行获取输出。

XML DOM 中 readystatechange 事件

就绪状态	说明
1	DOM 正在加载
2	DOM 已经加载完数据
3	DOM 已经可以使用，但某些部分还无法访问
4	DOM 已经完全可以
PS：readyState 可以获取就绪状态值	

```
var xmlDom = createXMLDOM();
xmlDom.async = true;                     //异步，可以不写
xmlDom.onreadystatechange = function () {
    if (xmlDom.readyState == 4) {        //完全加载了，再去获取 XML
        alert(xmlDom.xml);
    }
}
xmlDom.load('test.xml');                 //放在后面重点体现异步的作用
```

PS：可以通过 readyState 来了解事件的执行次数，将 load()方法放到最后不会因为代码的顺序而导致没有加载。并且 load()方法必须放在 onreadystatechange 之后，才能保证就绪状态变化时调用该事件处理程序，因为要先触发。用 PHP 来测试，在浏览器内部执行时，是否能操作，是否会假死。

PS：不能够使用 this，不能够用 IE 的事件处理函数，原因是 ActiveX 控件为了预防安全性问题。

PS：虽然可以通过 XML DOM 文档加载 XML 文件，但公认的还是 XMLHttpRequest 对象比较好。这方面内容，我们在 Ajax 章节详细了解。

4.解析错误

在加载 XML 时，无论使用 loadXML()或 load()方法，都有可能遇到 XML 格式不正确的情况。为了解决这个问题，微软的 XML DOM 提供了 parseError 属性。

parseError 属性对象

属性	说明
errorCode	发生的错误类型的数字代号
filepos	发生错误文件中的位置
line	错误行号
linepos	遇到错误行号那一行上的字符的位置
reason	错误的解释信息

```
if (xmlDom.parseError == 0) {
    alert(xmlDom.xml);
} else {
    throw new Error('错误行号：' + xmlDom.parseError.line +
                    '\n 错误代号：' + xmlDom.parseError.errorCode +
                    '\n 错误解释：' + xmlDom.parseError.reason);
}
```

二．DOM2 中的 XML

IE 可以实现了对 XML 字符串或 XML 文件的读取，其他浏览器也各自实现了对 XML 处理功能。DOM2 级在 document.implementaion 中引入了 createDocument()方法。IE9、Firefox、Opera、Chrome 和 Safari 都支持这个方法。

1.创建 XMLDOM 对象

```
var xmlDom = document.implementation.createDocument('','root',null);    //创建 xmlDom
var user = xmlDom.createElement('user');                                 //创建 user 元素
xmlDom.getElementsByTagName('root')[0].appendChild(user);                //添加到 root 下
var value = xmlDom.createTextNode('Lee');                                //创建文本
xmlDom.getElementsByTagName('user')[0].appendChild(value);               //添加到 user 下
alert(xmlDom.getElementsByTagName('root')[0].tagName);
alert(xmlDom.getElementsByTagName('user')[0].tagName);
alert(xmlDom.getElementsByTagName('user')[0].firstChild.nodeValue);
```

PS：由于 DOM2 中不支持 loadXML()方法，所以，无法简易的直接创建 XML 字符串。所以，只能采用以上的做法。

PS：createDocument()方法需要传递三个参数，命名空间，根标签名和文档声明，由于 JavaScript 管理命名空间比较困难，所以留空即可。文档声明一般根本用不到，直接 null 即可。命名空间和文档声明留空，表示创建 XMLDOM 对象不需要命名空间和文档声明。

PS：命名空间的用途是防止太多的重名而进行的分类，文档类型表明此文档符合哪种规范，而这里创建 XMLDOM 不需要使用这两个参数，所以留空即可。

2.载入 XML

DOM2 只支持 load()方法，载入一个同一台服务器的外部 XML 文件。当然，DOM2 也有 async 属性，来表面同步或异步，默认异步。

```
//同步情况下
var xmlDom = document.implementation.createDocument('','root',null);
xmlDom.async = false;
xmlDom.load('test.xml');
alert(xmlDom.getElementsByTagName('user')[0].tagName);

//异步情况下
var xmlDom = document.implementation.createDocument('','root',null);
xmlDom.async = true;
addEvent(xmlDom, 'load', function () {                    //异步直接用 onload 即可
    alert(this.getElementsByTagName('user')[0].tagName);
});
xmlDom.load('test.xml');
```

PS：不管在同步或异步来获取 load()方法只有 Mozilla 的 Firefox 才能支持，只不过新版的 Opera 也是支持的，其他浏览器则不支持。

3.DOMParser 类型

由于 DOM2 没有 loadXML()方法直接解析 XML 字符串，所以提供了 DOMParser 类型来创建 XML DOM 对象。IE9、Safari、Chrome 和 Opera 都支持这个类型。

```
var xmlParser = new DOMParser();                          //创建 DOMParser 对象
var xmlStr = '<user>Lee</user></root>';     //XML 字符串
var xmlDom = xmlParser.parseFromString(xmlStr, 'text/xml');   //创建 XML DOM 对象
alert(xmlDom.getElementsByTagName('user')[0].tagName);     //获取 user 元素标签名
```

PS：XML DOM 对象是通过 DOMParser 对象中的 parseFromString 方法来创建的，两个参数：XML 字符串和内容类型 text/xml。

4.XMLSerializer 类型

由于 DOM2 没有序列化 XML 的属性，所以提供了 XMLSerializer 类型来帮助序列化 XML 字符串。IE9、Safari、Chrome 和 Opera 都支持这个类型。

```
var serializer = new XMLSerializer();              //创建 XMLSerializer 对象
var xml = serializer.serializeToString(xmlDom);  //序列化 XML
alert(xml);
```

5.解析错误

在 DOM2 级处理 XML 发生错误时，并没有提供特有的对象来捕获错误，而是直接生成另一个错误的 XML 文档，通过这个文档可以获取错误信息。

```
var errors = xmlDom.getElementsByTagName('parsererror');
if (errors.length > 0) {
    throw new Error('XML 格式有误：' + errors[0].textContent);
```

```
}
```

PS：errors[0].firstChild.nodeValue 也可以使用 errors[0].textContent 来代替。

三．跨浏览器处理 XML

如果要实现跨浏览器就要思考几个个问题：1.load()只有 IE、Firefox、Opera 支持，所以无法跨浏览器；2.获取 XML DOM 对象顺序问题，先判断先进的 DOM2 的，然后再去判断落后的 IE；3.针对不同的 IE 和 DOM2 级要使用不同的序列化。4.针对不同的报错进行不同的报错机制。

```
//首先，我们需要跨浏览器获取 XML DOM
function getXMLDOM(xmlStr) {
    var xmlDom = null;

    if (typeof window.DOMParser != 'undefined') {          //W3C
        xmlDom = (new DOMParser()).parseFromString(xmlStr, 'text/xml');
        var errors = xmlDom.getElementsByTagName('parsererror');
        if (errors.length > 0) {
            throw new Error('XML 解析错误：' + errors[0].firstChild.nodeValue);
        }
    } else if (typeof window.ActiveXObject != 'undefined') {    //IE
        var version = [
                            'MSXML2.DOMDocument.6.0',
                            'MSXML2.DOMDocument.3.0',
                            'MSXML2.DOMDocument'
        ];
        for (var i = 0; i < version.length; i ++) {
            try {
                xmlDom = new ActiveXObject(version[i]);
            } catch (e) {
                //跳过
            }
        }
        xmlDom.loadXML(xmlStr);
        if (xmlDom.parseError != 0) {
            throw new Error('XML 解析错误：' + xmlDom.parseError.reason);
        }
    } else {
        throw new Error('您所使用的系统或浏览器不支持 XML DOM！');
    }

    return xmlDom;
}
```

```
//其次，我们还必须跨浏览器序列化 XML
function serializeXML(xmlDom) {
    var xml = '';
    if (typeof XMLSerializer != 'undefined') {
        xml = (new XMLSerializer()).serializeToString(xmlDom);
    } else if (typeof xmlDom.xml != 'undefined') {
        xml = xmlDom.xml;
    } else {
        throw new Error('无法解析 XML！');
    }
    return xml;
}
```

PS：由于兼容性序列化过程有一定的差异，可能返回的结果字符串可能会有一些不同。至于 load()加载 XML 文件则因为只有部分浏览器支持而无法跨浏览器。

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(yc60.com)联合提供：

本次主讲老师：李炎恢

我的博客：hi.baidu.com/李炎恢/

我的邮件：yc60.com@gmail.com