ie和firefox与DOM

(2012-07-17 20:21:18)

       最近都跟IE和Firefox结下了“不解之缘”，这两大浏览器Spansor之争还真是令人抓狂啊！Firefox的界面比较漂亮，对CSS的支持友好。但是到了Window的平台上与本地文件文件系统的交互就远远不如Microsoft自己的产品IE了。刚刚才处理好了Firefox与IE之间在CSS上的兼容，现在新问题又来了。本地文件的操作，Firefox要不是得不到full path就是load本地xml文件受限。没办法谁叫这个两套系统一个支持Firefox，另一个一定要支持IE呢！于是获得的启示是“钢铁就是这样炼成的！”。

       “多说无益”还是有实际解决问题的行动才行。

【基本的XML DOM操作方法】

－、元素查找问题

1. document.all[name]

  (1)现有问题：Firefox不支持document.all[name]

  (2)解决方法：使用getElementsByName(name)，getElementById(id)等来替代。

2. 集合类对象问题

  (1)现有问题：IE中对许多集合类对象取用时可以用 ()，但在Firefox只能用[]。

      如：IE中可以使用document.forms("formName")来返回名字为"formName"的Form，但在Firefox却行不通。

   (2)解决方法：使用[]，上例中可以改为document.forms["formName"]

3. HTML元素的ID在JavaScript可见

  (1)现有问题：IE中HTML元素中的ID可以作为document的下属对象变量名直接使用。在Firefox中不能。

  (2)解决方法：使用getElementById("idName")代替idName作为对象变量使用。

4. eval(idName)取得对象

  (1)现有问题：在IE中，利用eval(idName)可以取得ID为idName的HTML对象，在Firefox中不能。

  (2)解决方法：用 getElementById(idName) 代替 eval(idName)。

5. 变量名与某HTML对象ID相同

  (1)现有问题：在Firefox中，因为对象ID不作为HTML对象的名称，所以可以使用与HTML对象id相同的变量名，IE中不能。

  (2)解决方法：在声明变量时，一律加上var，以避免歧义，这样在IE中亦可正常运行。此外，最好不要取与HTML对象id相同的变量名，以减少错误。

  注：3、4和5都属于同一类的问题。

6. Frame

  (1)现有问题：在IE中可以用window.top.frameId和window.top.frameName来得到该Frame所代表的Window，Firefox中只能用window.top.frameName。

  (2)解决方法：将Frame的Id和Name设置成相同，使用window.top.frameName来访问Frame。

二、DOM操作

1. 设置元素的文本内容。

  (1)现有问题：IE使用innerText，而Firefox使用textContent来设置元素文本内容。

  (2)解决方法：如果文本内容不包含"<"和">"等特殊字符，可以使用innerHTML。否则，可以使用：

        var child = elem.firstChild;

            if (child != null) elem.removeChild(child);

            elem.a(document\_createTextNode(content));

2. parentElement，parent.children

  (1)现有问题：IE可以使用parentElement得父结点，parent.children得结点的所有子结点。Firefox不支持。

  (2)解决方法：使用parentNode和parent.childNodes。

3. 对childNodes的解释。

  (1)现有问题：IE和Firefox中对childNodes的解释不同，IE不会包含空白文本结点，而Firefox会包含。

  (2)解决方法：使用childNodes过滤文本结点，如下：

      var children = elem.childNodes;

          for (i = 0; i < children.length; i++) {

            if (children[i].nodeType != 3) { // 过滤文本结点

              // ...

            }

          }

4. 对document.getElementsByName的解释。

  (1)现有问题：IE中getElementsByName只会检查<input>和<img>元素，而在Firefox下会检查所有元素。

  (2)解决方法：不要使用getElementsByName检查除<input>和<img>之外的元素，如果要获得单个元素，尽量使用getElementById。

5. 对document.getElementByIdx\_x\_x的解释。

  (1)现有问题：IE中getElementById不仅检查Id属性，也会检查Name属性，当Name属性匹配参数时也会返回该元素。而在Firefox中只会检查Id属性。

  (2)解决方法：尽量保持Id和Name相同，不要让一个元素name属性和另一个元素的id属性相同。

三、事件

1. event.x与event.y问题

  (1)现有问题：在IE中，event对象有x,y 属性，Firefox中没有。

  (2)解决方法：在Firefox中，与event.x 等效的是 event.pageX。可以使用：

      mX = event.x ? event.x : event.pageX;

2. window.event

  (1)现有问题：使用window.event无法在Firefox上运行

  (2)解决方法：

         原代码(可在IE中运行)：

            <input type="button" name="someButton" value="提交" onclick="javascript:gotoSubmit()"/>

            ...

            <script language="javascript">

                function gotoSubmit() {

                    ...

                    alert(window.event);    // use window.event

                    ...

                }

            </script>

        新代码(可在IE和Firefox中运行)：

            <input type="button" name="someButton" value="提交" onclick="javascript:gotoSubmit(event)"/>

            ...

            <script language="javascript">

                function gotoSubmit(evt) {

                    evt = evt ? evt : (window.event ? window.event : null);

                    ...

                    alert(evt);             // use evt

                    ...

                }

            </script>

3. attachEvent和addEventListener

  (1)现有问题：IE中使用attachEvent来添加事件，Firefox中使用addEventListener。

  (2)解决方法：如下，注意事件参数的区别，一个是click，一个是onclick。

        if (document.attachEvent) document.attachEvent("click", clickHandler,false);

        else document.addEventListener("onclick",clickHandler);

四、语法

1. const

  (1)现有问题：在IE中不能使用const关键字。如const constVar = 32;在IE中这是语法错误。

  (2)解决方法：不使用const，以var代替。

2. 多余的逗号

  (1)现有问题：firefox中对象文字常量容许多余的逗号，在IE中不允许。下面语句在IE中非法。

      var obj = { 'key' : 'aaa', }

  (2)解决方法：去掉多余逗号。

五、XML

1. 创建XMLHttpRequest

  (1)现有问题：Firefox使用XMLHttpRequest，IE使用ActiveXObject。

  (2)解决方法：

      if (window.XMLHttpRequest) {

          req = new XMLHttpRequest();

      } else if (window.ActiveXObject) {

          req = new ActiveXObject("Microsoft.XMLHTTP");

      }

2. 创建DOM

  (1)现有问题：Firefox和IE创建DOM的方式不同。

  (2)解决方法：

        function createXmlDom() {

          var oXmlDom;

          if (Window.ActiveXObject) { // IE

            oXmlDom = new ActiveXObject("Microsoft.XmlDom");

          } else {  // Firefox

            oXmlDom = document.implementation.createDocument("", "", null);

          }

        }

3. 加载XML

  (1)现有问题：如果要加载外部文件IE和Firefox都可以用:

          oXmlDom.async=false;      // 这在Firefox中是必须的

          oXmlDom.load("test.xml");

     但是它们加载xml字符串的方式不一样，IE中直接可以使用oXmlDom.loadXML("<root><child/></root>")，而Firefox要使用DOMParser:

        var oParser = new DOMParser();

          var oXmlDom = oParser.parseFromString("<root/>", "text/xml");

  (2)解决方法：比较好的方法是给Firefox产生的XMLDom加上loadXML方法：

        if (isFirefox) { // 需要浏览器检测

          Document.prototype.loadXML = function(sXml) {

            var oParser = new DOMParser();

            var oXmlDom = oParser.parseFromString(sXml, "text/xml");

            while (this.firstChild) this.removeChild(this.firstChild);

            for (var i = 0; i < oXmlDom.childNodes.length; i++) {

              var oNewNode = this.importNode(oXmlDom.childNodes[i], true);

              this.a(oNewNode);

            }

          }

        }

      这样在IE和Firefox就可以调用loadXML方法了。

4. XPath支持

  (1)现有问题：IE中可以直接用XmlDOM的selectNodes来根据XPath表示式来选择结点，Firefox则比较复杂，需要使用XPathEvaluator。

     IE:

        var lstNodes = oXmlDom.documentElement.selectNodes("employee/name");

        for (var i = 0; i < lstNodes.length; i++) {

          alert(lstNodes[i].firstChild.nodeValue);

        }

     Firefox:

          var oEvaluator = new XPathEvaluator();

          var oResult = oEvaluator.evaluate("employee/name", oXmlDom.documentElement, null, XPathResult.ORDERED\_NODE\_ITERATOR\_TYPE, null);

          var oElement = oResult.iterateNext();

          while (oElement) {

            alert(oElement.firstChild.nodeValue);

            oElement = oResult.iterateNext();

          }

  (2)解决方法：比较好的方法给Firefox的Element添加selectNodes方法。

        if (isFirefox) { // 需要浏览器检测

            Element.prototype.selectNodes = function(sXPath) {

            var oEvaluator = new XPathEvaluator();

              var oResult = oEvaluator.evaluate(sXPath, this, null, XPathResult.ORDERED\_NODE\_ITERATOR\_TYPE, null);

              var aNodes = new Array();

              if (oResult != null) {

                var oElement = oResult.iterateNext();

                while (oElement) {

                  aNodes.push(oElement);

                  oElement = oResult.iterateNext();

                }

              }

              return aNodes;

           }

      }

   这样在IE和Firefox中就都可以调用selectNodes方法了。

5. XSLT支持

  (1)现有问题：IE中可以使用XmlDOM的transferNode方法将其转换成html，而Firefox需要使用XSLTProcessor。

  IE:

        oXmlDom.load("employee.xml");

        oXslDom.load("employee.xslt");

        var sResult=oXmlDom.transformNode(oXslDom);

    Firefox:

        var oProcessor = new XSLTProcessor();

        oProcessor.importStylesheet(oXslDom);

        var oResultDom = oProcessor.transformToDocument(oXmlDom);

        var oSerializer = new XMLSerializer();

        var sXml = oSerializer.serializeToString(oResultDom, "text/xml");

        alert(sXml);

  (2)解决方法：比较好的方法给Firefox的Node添加transferNode方法。

        if (isFirefox) { // 需要浏览器检测

          Node.prototype.transformNode = function(oXslDom) {

          var oProcessor = new XSLTProcessor();

            oProcessor.importStylesheet(oXslDom);

            var oResultDom = oProcessor.transformToDocument(oXmlDom);

            var oSerializer = new XMLSerializer();

            var sXml = oSerializer.serializeToString(oResultDom, "text/xml");

            return sXml;

          }

        }

   这样在IE和Firefox中就都可以调用transferNode方法了。

【Firefox下完整的读写文件的方法】

 <script language="JavaScript">

             //保存为本地文件

            function save(path,content) {

                try {

                    netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");

                } catch (e) {

                    alert("Permission to save file was denied.");

                }

                var file = Components.classes["@mozilla.org/file/local;1"]

                    .createInstance(Components.interfaces.nsILocalFile);

                file.initWithPath( path );

                if ( file.exists() == false ) {

                    //alert( "Creating file... " );

                    file.create( Components.interfaces.nsIFile.NORMAL\_FILE\_TYPE, 420 );

                }

                var outputStream = Components.classes["@mozilla.org/network/file-output-stream;1"]

                    .createInstance( Components.interfaces.nsIFileOutputStream );

                outputStream.init( file, 0x04 | 0x08 | 0x20, 420, 0 );

                 var converter = Components.classes["@mozilla.org/intl/scriptableunicodeconverter"]

                                .createInstance(Components.interfaces.nsIScriptableUnicodeConverter);

                converter.charset = 'UTF-8';

                var convSource = converter.ConvertFromUnicode(content);

                var result = outputStream.write( convSource, convSource.length );

                outputStream.close();

            }

            //读取本地文件

            function read(path) {

                try {

                    netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");

                } catch (e) {

                    alert("Permission to read file was denied.");

                }

                var file = Components.classes["@mozilla.org/file/local;1"]

                    .createInstance(Components.interfaces.nsILocalFile);

                file.initWithPath( path );

                if ( file.exists() == false ) {

                    alert("File does not exist");

                }

                var is = Components.classes["@mozilla.org/network/file-input-stream;1"]

                    .createInstance( Components.interfaces.nsIFileInputStream );

                is.init( file,0x01, 00004, null);

                var sis = Components.classes["@mozilla.org/scriptableinputstream;1"]

                    .createInstance( Components.interfaces.nsIScriptableInputStream );

                sis.init( is );

                var converter = Components.classes["@mozilla.org/intl/scriptableunicodeconverter"]

                                      .createInstance(Components.interfaces.nsIScriptableUnicodeConverter);

               converter.charset = "UTF-8";

                var output = converter.ConvertToUnicode(sis.read( sis.available() ));

                return output;

            }

                  save("d:\\Yes.txt","这下ok了吧");

     alert(read("d:\\Yes.txt"));

        </script>