

Thomas Wenzlaff

Microsoft JScript und MS XML für den Hobby-Programmierer

Stand 21.03.2007



Hinweise zum Dokument

Wer strukturiert und objektorientiert programmieren möchte, sollte sich z.B. mit Javascript/JScript beschäftigen.

Schwerpunkte dieser Dokumentation sind:

Programmierung mit Javascript/JScript unter dem Betriebssystem "Microsoft Windows 32-Bit"
und mit JScript verbundene **programmtechnische Verwendungen** der Produkte
"Microsoft Internet Explorer" unter und ab der Version 6.x. (vor allem ab 6.x),
"Microsoft XML" exemplarisch Version 3

Dieses vorliegende Dokument dient nicht als Lehrmaterial, sondern ist eine stark strukturierte Systematik. Sämtliche Beispiele dienen ausschließlich dem Verständnis und als Anregungen zur Programmierung. Da sich das Dokument an den Hobby-Programmierer richtet, befindet sich in diesem Dokument nur eine eingeschränkte, aber ausreichende Syntaxbeschreibung.

Autor: Thomas Wenzlaff
www.twseite.de

Stand des Dokumentes: 21.03.2007

Rechtewahrung, Haftung und Verbesserungsvorschläge:

Das vorliegende Dokument richtet sich ausschließlich an den Hobby-Programmierer.

Die Informationen in diesem Produkt werden ohne Rücksicht auf einen eventuellen Patentschutz verwendet. Warennamen, Hardware- und Softwarebezeichnungen, Softwarekomponenten sowie Algorithmen werden ohne Gewährleistung der freien Verwendbarkeit benutzt, gehören dem jeweiligen Eigentümer, oder sollten als solche betrachtet werden.

Der Autor kann für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen. Der missbräuchliche Einsatz dieser Dokumentation - vor allem bei sicherheitsrelevanter Programmierung - wird vom Autor grundsätzlich abgelehnt.

Für Verbesserungsvorschläge und Hinweise ist der Autor dankbar.

Warnung zur Zweckentfremdung von Javascript/JScript:

Es sei darauf hingewiesen, dass mit Javascript/JScript sicherheitsrelevante Aktionen programmierbar sind, die das Sicherheitsbedürfnis des Users betreffen und/oder rechtlich relevant werden können. Ein Problem kann die Unkenntnis des Benutzers sein, wenn o.g. Aktionen nicht für den User transparent gestaltet wurden.

Eine radikale Maßnahme gegen das Risiko ist z.B. die Abschaltung von Scripten (wie Javascript), ActiveX und Cookies (z.B. im Browser selbst oder anhand einer Firewall-Software). Die Abschaltung von Javascript durch den User macht die Anwendung dieser Dokumentation durch einen Programmierer hinfällig. Dem User sind daher dringend eine Antivirus- **und** eine Firewall-Software zu empfehlen, die **beide** auch während einer Netzwerksitzung im Internet aktiv sind. Z.B. erfolgen die Blockierung von nervenden Portscans durch Mitbenutzer jeder Art im Netzwerk und die Abwehr von immer häufiger eingesetzten Trojanern und Scriptviren (Scriptviren innerhalb eines HTML-Dokumentes oder einer HTML-Email werden durch die o.g. empfohlenen Softwares gefiltert).

Hinweise zur Kodierung von Quelltext in diesem Dokument:

Die im Dokument verwendete Rechtschreibung und Grammatik sowie das Layout des Dokumentes entsprechen nicht den aktuellen Rechtsnormen. In diesem Dokument werden auch aus Gründen der Transparenz DV-Fachbegriffe sowie Sprachstile verwendet, die nicht nur in der Sprachpflege umstritten sind. Der Ersatz deutscher Begriffe durch die aus der DV-Fachsprache ist nicht zu vermeiden. Konsequenterweise, hätte nur die englische Sprache verwendet werden dürfen, da die meisten Softwareentwicklungen und tendenziell neuen Produkte nicht aus dem deutschsprachigen Raum stammen.

Die in diesem Dokument verwendeten Einrückungen und Einschachtelungen von HTML-Elementen wie Tags, Scripten usw. dienen vorrangig der Transparenz z.B. von Quelltexten, Kommentaren und Syntax-Beschreibungen.



Inhaltsverzeichnis

1. Übersicht.....	4
1.1. Microsoft XML-Versionen.....	4
1.2. Microsoft HTML-DOM mit XML	4
1.3. Microsoft XML-DOM	4
1.4. Microsoft XML im JScript.....	4
2. HTML-DOM und XML-Datenverarbeitung	5
2.1. Objekt XML im HTML-DOM (Nutzung von XML-Dateninseln, Data Islands).....	5
2.1.1. XLM-Dateninseln (Data Islands)	5
2.1.2. XLM-Objekt (XML-Dateninsteln per HTML-DOM)	5
2.2. HTML-DOM Eigenschaft .XMLDocument der Objekte doument und XML	7
2.3. XML-DOM Eigenschaft .documentElement	8
2.4. Objekt XMLHttpRequest im HTML-DOM (ab IE 7 XML-Datenaustausch)	8
2.5. Microsoft Ajax für ASP.Net Framework.....	13
3. XML-DOM (Übersicht).....	14
3.1. Knotenarten (IXMLDOMNodeType).....	15
3.2. Eigenschaften (Attribute) - Übersicht	17
3.3. Methoden - Übersicht	25
3.4. Events - Übersicht	30
4. Index.....	31



1. Übersicht

XML-Daten per Script sind ansprechbar per

DOM (XML-DOM und / oder HTML-DOM (Schnittstellen zwischen DOM vorhanden))
mit oder ohne Active-X-Control.

Für das HTML-DOM gilt:

Einerseits wird die vorhandene Einbindung von XML-Dateninseln (siehe XML-Objekt) ab XML-Version 6.0 und zukünftig nicht mehr unterstützt (z.Z. nur noch von auslaufenden MS XML-Versionen), andererseits wurde mit dem Internet Explorer 7 das XMLHttpRequest-Objekt zum Datentransfer von XML-Daten eingeführt (XMLHttpRequest-Objekt gibt es schon länger bei Browsern der Konkurrenz und wird gern in Ajax-orientierten Softwares genutzt).

1.1. Microsoft XML-Versionen

Microsoft XML umfasst am Beispiel von Version 6 vom 07.12.2006 folgende Komponenten:

XML 1.0 (DOM & SAX2 APIs)
XML Schema (XSD) 1.0
XPath 1.0
XSLT 1.0.

Microsoft XML ab Version 6.0 unterstützt XML-Dateninseln nicht mehr.

Vorgänger-Versionen von 6.0 (die XML-Dateninseln unterstützten) werden nur noch gewartet und laufen aus.

MS XML-Versionen sind parallel installierbar - dazu Microsoft:

'MSXML6 wird parallel zu MSXML 3 und MSXML4 installiert und beeinträchtigt vorhandene Anwendungen, die MSXML3 und MSXML4 verwenden, nicht (versionsunabhängige ProgIDs verweisen nicht auf MSXML6). Entwickler müssen zu neuen "60" ProgIDs übergehen, um MSXML6 in ihren Anwendungen zu nutzen. Weitere Informationen finden Sie im MSXML6 SDK (bald erhältlich).'

MS XML-Versionen sollten je ihr Software Developer Kit (SDK) erhalten.

Microsoft-Tools zu XML-Versionen
sind spezifisch zur XML-Version
können das Net-Framework benötigen.

1.2. Microsoft HTML-DOM mit XML

Microsoft HTML-DOM bietet

XML-Objekt
XMLHttpRequest-Objekt

XML-Dateninsel im HTML-Code
ab IE 7 Datentransfer synchron, asynchron

an.

XML-Dateninseln

werden nur noch von nicht mehr weiter entwickelten XML-Versionen unterstützt (unter Version 6.0)
sind ab XML-Version 6.0 und zukünftig nicht mehr verfügbar.

1.3. Microsoft XML-DOM

Microsoft XML-DOM wird weiter unten in Auszügen beschrieben.

1.4. Microsoft XML im JScript

Der Zugriff auf XML-Daten erfolgt per HTML-DOM, XML-DOM, mit oder ohne Active-X-Control.



2. HTML-DOM und XML-Datenverarbeitung

HTML-DOM

hat eigene Komponenten für XML-Datenverarbeitung

kann über Schnittstellen das XML-DOM benutzen.

z.B. per HTML-DOM-Eigenschaft .XMLDocument

Schnittstelle zum XML-Code: Root des XML-Daten-Knoten (XML-Objektes)

wird unterstützt von

Objekt document

Objekt XML

z.B. per XML-DOM-Eigenschaft .documentElement

Je nach XML-Version ist die XML-Datenverarbeitung möglich.

MS XML 6 schließt die Nutzung von XML-Dateninseln aus (siehe Objekt XML)

Vorgänger von MS XML 6 unterstützen zwar XML-Dateninseln, aber diese XML-Versionen laufen aus.

2.1. Objekt XML im HTML-DOM (Nutzung von XML-Dateninseln, Data Islands)

2.1.1. XLM-Dateninseln (Data Islands)

Dateninseln (Data Islands)

werden ab MSXML 6 und zukünftig nicht mehr unterstützt

sind als XML-Objekt im HTML-DOM implementiert.

2.1.2. XLM-Objekt (XML-Dateninseln per HTML-DOM)

Das XML-Objekt referenziert eine XML-Dateninsel (XML-Code) im HTML-Code bzw. per Script-Objekt (XML-Daten-Knoten). Ab IE 5 sind Dateninseln les- und schreibbar.

Einbindung von XML-Code in HTML-Code:

direkt kodiert als nicht verschachtelbare XML-Objekte
per SRC (im XML-Objekt oder SCRIPT-Objekt)

Beispiele:

```
<SCRIPT ID="XMLID" LANGUAGE="XML">
  <XMLDATA>
    <DATA>TEXT</DATA>
  </XMLDATA>
</SCRIPT>
```

```
<XML ID="XMLID"
  LANGUAGE="XML"
  TYPE="text/xml"
  SRC="http://localhost/xmlFile.xml">
</XML>
```

Schnittstelle zur Root des XML-Daten-Knoten:

HTML-DOM-Eigenschaft .XMLDocument der Objekte document und XML.

Beispiele für XML-Objekt:

```
function returnXMLData()
{return document.all("XMLID").XMLDocument.nodeValue;}
```

```
function returnXMLData()
{return XMLID.documentElement.text;}
```

```
function returnIslandRootName()
{var islandRoot = document.all("SCRIPT").XMLDocument;
return islandRoot.nodeName;}
```



```
}
```

Schnittstelle zu den Inhalten des XML-Daten-Knoten:

XML-DOM-Eigenschaft .documentElement

Beispiel:

```
function returnXMLData()
{return XMLID.documentElement.text;}
```

Attribute:

canHaveHTML

id ID

isContentEditable

isDisabled

isMultiLine

parentElement

readyState Stringwert

korrespondiert mit readyState des XML-DOM-Document-Objektes (dort aber numerisch)

Eigenschaft wird im HTML- und im XML-DOM synchron belegt, aber mit divergenten Werten:

readyState per HTML-DOM	'complete'	Objekt komplett geladen und initialisiert
readyState per XML-DOM	4	Objekt komplett geladen und initialisiert

Man fragt readyState also im HTML- oder im XML-DOM ab.

Beispiel:

```
<XML ID="ID_DatenInsel">
<METADATA>
  <AUTHOR>Test Autor</AUTHOR>
  <GENERATOR>Visual Notepad</GENERATOR>
  <PAGETYPE>Reference</PAGETYPE>
  <ABSTRACT>Definiert eine Dateninsel</ABSTRACT>
</METADATA>
</XML>
```

```
// HTML-DOM
if (ID_DatenInsel.readyState == "complete")
{window.alert ("The XML document is ready.");}
```

```
// XML-DOM
if (ID_DatenInsel.XMLDocument.readyState == 4)
{window.alert ("The XML document is ready.");}
```

recordset

scopeName

src SRC

tagUrn

XMLDocument (Eigenschaft vom Objekt document und XML-Objekt):

Referenz auf Root des XML-Baumes laut XML-DOM (auf XML-Dokument)

```
[var Zeiger =] object.XMLDocument
```

Beispiel:

```
<XML ID="ID_DatenInsel">
<METADATA>
  <AUTHOR>Test Autor</AUTHOR>
```



```

    <GENERATOR>Visual Notepad</GENERATOR>
    <PAGETYPE>Reference</PAGETYPE>
    <ABSTRACT>Definiert eine Dateninsel</ABSTRACT>
  </METADATA>
</XML>

var XML_Dateninsel_Root= ID_DatenInsel.XMLDocument;
var XML_Dateninsel_Knoten=
    XML_Dateninsel_Root.selectSingleNode("METADATA/ABSTRACT");
    // XML-Pfad im Dokument
var XML_Dateninsel_Knoten_Textinhalt= XML_Dateninsel_Knoten.text;
    // Text zwischen ABSTRACT-Tags
alert(XML_Dateninsel_Knoten_Textinhalt);

```

Methoden:

addBehavior
 componentFromPoint
 fireEvent
 getAttributeNode Referenz auf HTML-DOM-Attribut zum Objekt
 namedRecordset
 normalize
 removeAttributeNode HTML-DOM-Attribut entfernen
 removeBehavior
 setAttributeNode HTML-DOM-Attribut setzen

Events:

ondataavailable
 ondatasetchanged
 ondatasetcomplete
 onreadystatechange
 onrowenter
 onrowexit
 onrowsdelete
 onrowsinserted

Styles:

behavior behavior
 ext-autospace textAutospace
 text-underline-position textUnderlinePosition

2.2. HTML-DOM Eigenschaft .XMLDocument der Objekte document und XML

Referenz auf Root des XML-Baumes laut XML-DOM (auf XML-Dokument-Root)

```
[var Zeiger =] object.XMLDocument
```

Beispiel:

```

<XML ID="ID_DatenInsel">
  <METADATA>
    <AUTHOR>Test Autor</AUTHOR>
    <GENERATOR>Visual Notepad</GENERATOR>
    <PAGETYPE>Reference</PAGETYPE>
    <ABSTRACT>Definiert eine Dateninsel</ABSTRACT>
  </METADATA>
</XML>

var XML_Dateninsel_Root= ID_DatenInsel.XMLDocument;
var XML_Dateninsel_Knoten=
    XML_Dateninsel_Root.selectSingleNode("METADATA/ABSTRACT");
    // XML-Pfad im Dokument
var XML_Dateninsel_Knoten_Textinhalt= XML_Dateninsel_Knoten.text;
    // Text zwischen ABSTRACT-Tags
alert(XML_Dateninsel_Knoten_Textinhalt);

```



2.3. XML-DOM Eigenschaft .documentElement

Zeiger auf Root des XML-Dokumentes

```
var Root_Zeiger = XML_dokument_Objekt.documentElement;
XML_dokument_Objekt.documentElement= Root_Zeiger;
```

lesen und schreiben

lesen: Zeiger auf IXMLDOMElement also auf XML-Dokument-Objekt (z.B. aus geladener XML-Daten)
null wenn Root nicht existent

schreiben:

Objekt wird als XML_dokument_objekt-Objekt an die Liste der Kinder-Dokument-Typ-Knoten angehängen (wird zum letzten Kind), wobei .parentNode den Zeiger des Knoten erhält für Einfügen an anderer Position: siehe XML-DOM-Funktion insertBefore

XML_dokument_Objekt kann auch belegt werden z.B. per

```
xml_html_dom_objekt.XMLDocument
document.XMLDocument
```

Beispiel mit Active-X-Control:

```
// XML-Dokument-Objekt erzeugen mit MSXML-Version 3.0
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
// Daten in das XML-Objekt laden
XML_DOM_Objekt.async = false;
XML_DOM_Objekt.load("xml_datei.xml");
// prüfen ob Daten ohne Fehler nach MSXML-Version 3.0
if (XML_DOM_Objekt.parseError.errorCode != 0)
{var XML_Datei_Fehler = XML_DOM_Objekt.parseError;
 alert("Datenfehler in xml_datei.xml erkannt: " + XML_Datei_Fehler.reason);
}
else
{// XML-Daten fehlerfrei
 // Root holen
 var XML_Daten_Root = XML_DOM_Objekt.documentElement;
 // Root abklappen und Textinhalte aller Kinder in der Root anzeigen
 for (var i=0; i<XML_Daten_Root.childNodes.length; i++)
 {alert(XML_Daten_Root.childNodes.item(i).childNodes.item(0).text);}
}
```

2.4. Objekt XMLHttpRequest im HTML-DOM (ab IE 7 XML-Datenaustausch)

Das Objekt dient dem Laden und Versenden von XML-Daten über Http-Request vom / zum Server der Webseite:
Asynchroner oder synchroner Zugriff (Request) ohne Active-X-Control.

Elternobjekt ist window.

XML-Daten können nur per XML-DOM clientseitig in renderbare HTML-Daten konvertiert werden. Alternativ ist Extensible Stylesheet Language Transformations (XSLT) nutzbar.

XML-Daten

- sind inzwischen trotz diverser Konventionen ein Austauschformat zwischen Anwendungen (nicht nur unter Windows) geworden. (daher auch oft kostenpflichtige Softwares)
- sind nicht gerade platzsparend erzeugbar, dafür als schlichte Textdatei
- sind eventuell Konkurrenz z.B. zu Adobe PDF
- sind im Syntax exakter zu verarbeiten als z.B. die (zulässigen) HTML-Tag-Verstümmelungen (in Programmierer-Faulheit) (allerdings ist das kein Vorteil, sondern das Ausmerzen von unnötigen Unzulässigkeiten und Sicherheitsproblemen)
- sind objektorientiert auswertbar

Beispiel 1:

```
var xmlHttp;
```




```
if (window.XMLHttpRequest) {xmlHttp = new XMLHttpRequest();} // IE7, Mozilla, Safari, etc.
else{if (window.ActiveXObject) {xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");}}
```

Beispiel 2:

// IE 5 bis 6

```
if (window.XMLHttpRequest)
{
    var ReqObjekt = new XMLHttpRequest();
    ReqObjekt.open("GET", "http://localhost/test.xml");
    ReqObjekt.send();
    alert(ReqObjekt.statusText);
}
```

Attribute:

onreadystatechange Schnittstelle zum Einbinden eines Eventhandler zu onreadystatechange
(Zuweisen z.B. eines Funktionszeigers)
Achtung: Namensidentität mit Event

Eventhandler einbinden, der den Status des Datenzugriffes überwacht:

Eigenschaft .readyState auswerten.

Es wird davon ausgegangen dass nicht nur 1 Datenübertragung stattfindet. Also soll mit jeder Statusänderung readyState ausgewertet werden.

```
XMLHttpRequest.onreadystatechange=zeiger_auf_handler;
```

lesen und schreiben

Beispiel:

```
function EventHandler_complete()
{if (ReqObjekt.readyState == 4)alert('Daten geladen.')}

var ReqObjekt = new XMLHttpRequest();
ReqObjekt.onreadystatechange = EventHandler_complete; // ohne '()' !!
ReqObjekt.open("GET", "http://localhost/test.xml", true);
ReqObjekt.send();
```

readyState liefert den aktuelle Status des Objektes

Status des Objektes

```
[ nState = ] XMLHttpRequest.readyState
```

nState	Integer
--------	---------

0	Objekt ist zwar erzeugt aber nicht initialisiert (Methode open noch nicht aktiviert worden)
---	---

1	Methode open ist aktiviert worden, Methode send nicht aktiviert worden
---	--

2	Methode send ist aktiviert worden, Status und Headers noch nicht verfügbar
---	--

3	Daten wurden empfangen, aber Status und Headers sind nicht vollständig verfügbar
---	--

4	Alle Daten sind verfügbar (Objekt komplett initialisiert)
---	---

(nicht 'complete' verwenden: das gibt es nur im HTML-DOM für XML-Objekt)

nur lesen

responseBody liefert Zeiger auf Feld aus vorzeichenlosen Bytes des Body

Daten des Antwort-Body als ein Feld vorzeichenloser Bytes holen

```
[ vBody = ] XMLHttpRequest.responseBody
```



vBody Zeiger auf Datenfeld

nur lesen

responseText liefert String des Body

Daten des Antwort-Body als ein String

```
[ sBody = ] XMLHttpRequest.responseText
```

sBody Zeiger auf String

nur lesen

responseXML liefert Body als Document im XML-DOM

Daten des Antwort-Body als ein XML-DOM-Objekt

```
[ oBody = ] XMLHttpRequest.responseXML
```

oBody Zeiger auf Object

nur lesen

Beispiel:

```
var ReqObjekt = new XMLHttpRequest();
ReqObjekt.open("GET", "http://localhost/test.xml",false);
ReqObjekt.send();
alert(ReqObjekt.responseXML.xml);
```

status liefert HTTP-Status des Datenzugriffes

numerischer HTTP Status des Request

```
[ nStatus = ] XMLHttpRequest.status
```

nur lesen

nStatus Integer

100	Continue
101	Switching protocols
200	OK
201	Created
202	Accepted
203	Non-Authoritative Information
204	No Content
205	Reset Content
206	Partial Content
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
307	Temporary Redirect
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden



404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Request Entity Too Large
414	Request-URI Too Long
415	Unsupported Media Type
416	Requested Range Not Suitable
417	Expectation Failed
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported

Beispiel:

```
var ReqObjekt = new XMLHttpRequest();
ReqObjekt.open("GET", "http://localhost/test.xml",false);
ReqObjekt.send();
if (ReqObjekt.status == 401){alert('Access denied.')}
else{alert(ReqObjekt.responseText);}
```

statusText liefert Text zum HTTP-Status des Datenzugriffes

Zeichenketten-HTTP Status des Request

[sStatus =] XMLHttpRequest.statusText

sStatus siehe .status

nur lesen

Beispiel:

```
var ReqObjekt = new XMLHttpRequest();
ReqObjekt.open("GET", "http://localhost/test.xml",false);
ReqObjekt.send();
if (ReqObjekt.status == 'OK'){alert(ReqObjekt.responseText);}
else{alert(ReqObjekt.statusText);}
```

Methoden:

abort Abbruch des aktuellen HTTP-Datenzugriffes
liefert nichts

getAllResponseHeaders komplette Liste der Antwort-Header liefern

sHeaders = XMLHttpRequest.getAllResponseHeaders()

String als Liste

Listenaufbau:

Folge von Paaren aus Name und Wert

Paare getrennt durch carriage return/line feed (CR/LF)



getResponseHeader Wert eines Antwort-Header liefern

```
p = XMLHttpRequest.getResponseHeader(bstrHeader)
```

bstrHeader String, Header-Name

open einen Datenzugriff öffnen im Sinne von Init

Request öffnen, ermöglicht danach send()

```
XMLHttpRequest.open(sMethod, sUrl [, bAsync] [, sUser] [, sPassword])
```

sMethod String, HTTP Methode

'GET'

'POST'

'HEAD'

'PUT'

'DELETE'

'MOVE'

'PROPFIND'

'PROPPATCH'

'MKCOL'

'COPY'

'LOCK'

'UNLOCK'

'OPTIONS'

Gross-kleinschreibung egal

sUrl String, Url (absolut oder relativ) der XML-Daten oder des server-seitigen XML-Web-Services.

alle Urls müssen identische Domain, Port und Protokoll nutzen wie die Webseite, die XML-Daten per XMLHttpRequest-Objekt nutzen will

bAsync optional.

true asynchroner Request

Eventhandler für onreadystatechange verwenden für Statusabfrage.

false nicht asynchroner Request

sUser optional

String Name des Users für Authentifizierung (wenn diese verlangt wird)

wenn Leerkette so Login-Fenster angezeigt

sPassword optional

String Passwort des Users für Authentifizierung (wenn diese verlangt wird)

nur ausgewertet wenn sUser nicht Leerkette ist (da ansonsten durch Login-Fenster bedient)

liefert nichts

Internet Explorer legt Daten aus GET immer in den Browsercache ab

aus POST niemals in den Browsercache ab

send HTTP-Datenzugriff als Senden ausführen:

Senden von HTTP-Headern an Server

Empfangen einer Antwort

benötigt irgendwann open() zuvor

```
XMLHttpRequest.send( [varBody])
```

varBody optional, Body der Sendedaten z.B. String, Feld vorzeichenloser Bytes oder XML-DOM-Objekt

liefert nichts



setRequestHeader benutzerdefinierten Header einem HTTP-Datenzugriff per send() hinzufügen

XMLHttpRequest.setRequestHeader(sName, sValue)

sName Required. String that specifies the header name. (Teil 1 des Paares)

sValue Required. String that specifies the header value. (Teil 2 des Paares)

liefert nichts

Beispiel:

```
var ReqObjekt = new XMLHttpRequest();
....
ReqObjekt.open("POST", sUrl, false);
....
ReqObjekt.setRequestHeader("Content-Type", "text/xml");
....
ReqObjekt.send(sRequestBody);
```

Events:

onreadystatechange

Styles:

keine

2.5. Microsoft Ajax für ASP.Net Framework

arbeitet zusammen mit folgenden Microsoft-Tools:

MS Visual Web Developer Expression (free)

hat keinen HTTP-Server

hat einfachen Debugger (Breakpoint setzbar)

Achtung: Auch wenn Installation auf einem anderen Laufwerk als C:\
erfolgen soll, so werden mindestens 1GByte Daten
auf C:\ installiert (der winzige Rest dann auf das
Installationslaufwerk ungleich C:\)

MS Visual Web Developer (kostenpflichtig)

bietet

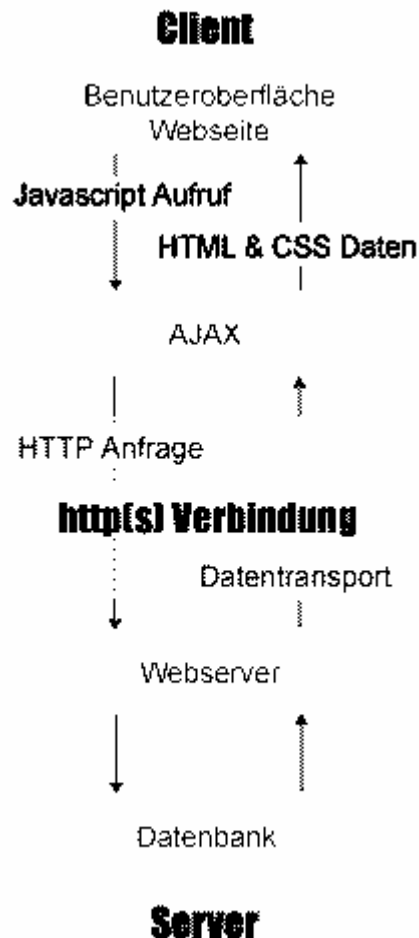
vollen Support für JavaScript

Komponenten zur Datenverwaltung und Datenrenderung

intelligente Code-Komplettierung

server- und clientseitiger Zugriff (z.B. falls verfügbar per JScript.Net)





<http://www.asp.net/>

Microsoft Ajax-Webseite im ASP-Net-Framework

'ASP.NET AJAX is a free framework'
...

'Featured ASP.NET AJAX Web Hosting
ASP.NET 2.0 Web Hosting

ASP.NET AJAX Compatible Hosting with MS SQL 2005, Real-Time SQL Backups, SQL Express to SQL 2005 Transfer Tool, SQL Restore Tool, Web Services Supported, Free ASP.NET Components and more from DiscountASP.NET, voted 2006 & 2005 Best ASP.NET Hosting Service by asp.netPRO Magazine. Get your ASP.NET AJAX powered web site/application online with the leader in ASP.NET Web Hosting.'

3. XML-DOM (Übersicht)

Vom XML-DOM wird zwecks Dateneinbindung in ein HTML-Code oder per JScript eigentlich nur `IXMLDOMElement`, also das XML-Dokument-Objekt (z.B. aus geladener XML-Daten), benötigt.

Zum Zweck der XML-Datenweiterverarbeitung im HTML-DOM ist aber die Schnittstelle zum XML-DOM nötig.

XML-DOM benötigt, wenn nicht über HTML-DOM angesprochen, das Active-X-Control je nach installierter MS XML-Version.

HTML-DOM

hat eigene Komponenten für XML-Datenverarbeitung
kann über Schnittstellen das XML-DOM benutzen.

z.B. per HTML-DOM-Eigenschaft `.XMLDocument`



Schnittstelle zum XML-Code: Root des XML-Daten-Knoten (XML-Objektes)
wird unterstützt von
Objekt document
Objekt XML
z.B. per XML-DOM-Eigenschaft .documentElement

Je nach XML-Version ist die XML-Datenverarbeitung möglich.

MS XML 6 schließt die Nutzung von XML-Dateninseln aus (siehe Objekt XML)

Vorgänger von MS XML 6 unterstützen zwar XML-Dateninseln, aber diese XML-Versionen laufen aus.

3.1. Knotenarten (*IXMLDOMNodeType*)

Knoten sind Kinder oder können Kinder vom / im XML-Dokument-Baum als Objekt sein.

sind selbst Objekte (Unterobjekte vom XML-Dokument-Baum-Objekt)

können eventuell selbst Kinder haben

bilden eine Baum-Hierarchie (Root ist das XML-Dokument-Baum-Objekt)

Knotenarten sind Datenarten

werden u.a. als vordefinierte Variablen mit konstantem Wert hinterlegt.

NODE_ELEMENT (1) mit .nodeTypeString "element"

Kindarten:

Element,
Text,
Comment,
ProcessingInstruction,
CDATASection
EntityReference.

kann Kind sein von

Document,
DocumentFragment,
EntityReference,
Element.

NODE_ATTRIBUTE (2) mit .nodeTypeString=="attribute"

Attribut eines Elementes

Kindarten:

Text
EntityReference.

kann nicht Kind sein

NODE_TEXT (3) mit .nodeTypeString "text"

Text zwischen Tag-Begrenzern

kann keine Kinder haben

kann Kind sein von

Attribute,
DocumentFragment,
Element,
EntityReference.

NODE_CDATA_SECTION (4) mit .nodeTypeString "cdatasection"

CDATA-Section der XML-Quelle

kann keine Kinder haben

kann Kind sein von

DocumentFragment,
EntityReference,
Element

NODE_ENTITY_REFERENCE (5) mit nodeTypeString "entityreference"

Referenz auf eine beliebige Daseinsform im XML-Dokument z.B. Zeichenform

Kindarten:

Element,



ProcessingInstruction,
 Comment,
 Text,
 CDATASection,
 EntityReference.
 kann Kind sein von
 Attribute,
 DocumentFragment,
 Element,
 EntityReference

NODE_ENTITY (6) mit .nodeTypeString "entity"
 Daseinsform im XML-Dokument: DocumentType
 Kindarten:
 Element,
 ProcessingInstruction,
 Comment,
 Text,
 CDATASection,
 EntityReference.
 kann Kind sein von
 DocumentType

NODE_PROCESSING_INSTRUCTION (7) mit .nodeTypeString "processinginstruction"
 Prozess-Instruktion im XML-Dokument
 kann keine Kinder haben
 kann Kind sein von
 Document,
 DocumentFragment,
 Element,
 EntityReference

NODE_COMMENT (8) mit .nodeTypeString "comment"
 Kommentar im XML-Dokument
 kann keine Kinder haben
 kann Kind sein von
 Document,
 DocumentFragment,
 Element,
 EntityReference

NODE_DOCUMENT (9) mit .nodeTypeString "document"
 Document-Objekt als Root des XML-Dokument-Baumes (beachte DocumentFragment)
 erzeugbar per
 XML-DOM

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.X.Y");
```

 mit X.Y je nach installierter Version von MS XML
 z.B. 3.0
 4.0
 6.0
 wobei Versionen parallel installierbar sind
 .documentElement
 HTML-DOM
 XML-Objekt
 nicht mehr ab MS XML 6.0 da dieses keine Dateninseln unterstützt
 Dateninseln unterstützen Version unter 6.0, aber diese laufen aus
 SCRIPT
 .XMLDocument von
 Objekt document
 Objekt XML



Kindarten:

Element maximal 1 Kind dieser Art
 ProcessingInstruction,
 Comment,
 DocumentType

kann kein Kind sein

NODE_DOCUMENT_TYPE (10) mit .nodeTypeString property "documenttype"

Document-Type-Deklaration laut <!DOCTYPE>-Tag

Kindarten:

Notation
 Entity

kann Kind sein von

Document

NODE_DOCUMENT_FRAGMENT (11) mit nodeTypeString "documentfragment" (beachte Document)

Fragment des XML-Dokumentes als Knoten als Container für Unterbaum als Unterdokument

Kindarten

Element,
 ProcessingInstruction,
 Comment,
 Text,
 CDATASection,
 EntityReference

kann kein Kind sein

NODE_NOTATION (12) mit .nodeTypeString "notation"

Notation in der Document-Type-Deklaration laut <!DOCTYPE>-Tag

kann keine Kinder haben

kann Kind sein von

DocumentType

3.2. *Eigenschaften (Attribute) - Übersicht*

async asynchronen Datentransfer ein/aus, lesen und schreiben

```
boolValue = oXMLDOMDocument.async;
objXMLDOMDocument.async = boolValue;
```

boolValue true so asynchron Download
 false so synchroner Download

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
XML_DOM_Objekt.async = false;
XML_DOM_Objekt.load("xml_datei.xml");
```

attributes Zeiger auf Liste der Attribute eines Knoten, nur lesen

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var AttributCollection= XML_DOM_Objekt.documentElement.firstChild.attributes;
alert(AttributCollection.length);
```

baseName Basisname eines Namensraumes, nur lesen

childNodes Zeiger auf Liste aller Kinderknoten, nur lesen

Bsp.:



```
// XML-Dokument-Objekt erzeugen mit MSXML-Version 3.0
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
// Daten in das XML-Objekt laden
XML_DOM_Objekt.async = false;
XML_DOM_Objekt.load("xml_datei.xml");
// prüfen ob Daten ohne Fehler nach MSXML-Version 3.0
if (XML_DOM_Objekt.parseError.errorCode != 0)
{ var XML_Datei_Fehler = XML_DOM_Objekt.parseError;
  alert("Datenfehler in xml_datei.xml erkannt: " + XML_Datei_Fehler.reason);
}
else
{ // XML-Daten fehlerfrei
  // Root holen
  var XML_Daten_Root = XML_DOM_Objekt.documentElement;
  // Root abklappen und Textinhalte aller Kinder in der Root anzeigen
  for (var i=0; i<XML_Daten_Root.childNodes.length; i++)
  { alert(XML_Daten_Root.childNodes.item(i).childNodes.item(0).text); }
}
```

dataType Datentyp eines Knoten, lesen und schreiben

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
XML_DOM_Objekt.async = false;
XML_DOM_Objekt.loadXML("<root/>");
var root = XML_DOM_Objekt.documentElement;
root.dataType = "int";
root.nodeTypeValue = 5;
alert(XML_DOM_Objekt.xml);
```

definition Definition eines Knoten als Document Type Definition (DTD) oder als Schema, nur lesen

doctype Dokumenttyp laut DTD, nur lesen

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var MyDocType = XML_DOM_Objekt.doctype;
if (MyDocType != null) { alert(MyDocType.name); }
```

documentElement Zeiger auf Root des XML-Dokumentes, lesen und schreiben

Bsp.:

```
// XML-Dokument-Objekt erzeugen mit MSXML-Version 3.0
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
// Daten in das XML-Objekt laden
XML_DOM_Objekt.async = false;
XML_DOM_Objekt.load("xml_datei.xml");
// prüfen ob Daten ohne Fehler nach MSXML-Version 3.0
if (XML_DOM_Objekt.parseError.errorCode != 0)
{ var XML_Datei_Fehler = XML_DOM_Objekt.parseError;
  alert("Datenfehler in xml_datei.xml erkannt: " + XML_Datei_Fehler.reason);
}
else
{ // XML-Daten fehlerfrei
  // Root holen
  var XML_Daten_Root = XML_DOM_Objekt.documentElement;
  // Root abklappen und Textinhalte aller Kinder in der Root anzeigen
```



```

    for (var i=0; i<XML_Daten_Root.childNodes.length; i++)
    { alert(XML_Daten_Root.childNodes.item(i).childNodes.item(0).text); }
}

```

firstChild Zeiger auf 1. Kind nur lesen

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.firstChild;
alert(currNode.xml);

```

implementation Zeiger auf IXMLDOMImplementation-Object des XML-Dokumentes, nur lesen

lastChild Zeiger auf letztes Kind, nur lesen

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
var newNode = XML_DOM_Objekt.createElement(1, "VIDEOS", "");
var currNode = root.insertBefore(newNode, root.lastChild);
alert(root.xml);

```

namespaceURI liefert Uniform Resource Identifier (URI) des Namensraumes, nur lesen

nextSibling nächster Eintrag des Knoten in der elterlichen Knotenkinderliste, nur lesen

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.childNodes.item(0);
var nextNode = currNode.nextSibling;
alert(nextNode.xml);

```

nodeName Name des Knoten folgender Art

- attribute,
- document type,
- element,
- entity,
- notation nodes
- alle anderen Knotentypen unter gemeinsamen Pauschalnamen

nur lesen

Knotennamen sind

NODE_ATTRIBUTE	enthält Attributname
NODE_CDATA_SECTION	enthält den String "#cdata-section"
NODE_COMMENT	enthält den String "#comment"
NODE_DOCUMENT	enthält den String "#document"
NODE_DOCUMENT_TYPE	enthält den Namen des Dokument-Typen z.B. xxx in <!DOCTYPE xxx ...>
NODE_DOCUMENT_FRAGMENT	enthält den String "#document-fragment"
NODE_ELEMENT	enthält den XML-Tag mit allen verfügbaren Namensräumen-Präfixen
NODE_ENTITY	enthält den Namen des entity
NODE_ENTITY_REFERENCE	enthält den Namen der Referent auf entity Name enthält keine führende & und ; enthält verfügbaren Namensraum
NODE_NOTATION	enthält Name der notation



NODE_PROCESSING_INSTRUCTION enthält das Ziel-Token dem folgt '<?'
 NODE_TEXT enthält den String "#text"

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.childNodes.item(0);
alert(currNode.nodeName);
```

nodeType Knotentyp (Knotenart), nur lesen

Knotentypen sind

NODE_ATTRIBUTE (2)
 NODE_CDATA_SECTION (4)
 NODE_COMMENT (8)
 NODE_DOCUMENT (9)
 NODE_DOCUMENT_TYPE (10)
 NODE_DOCUMENT_FRAGMENT (11)
 NODE_ELEMENT (1)
 NODE_ENTITY (6)
 NODE_ENTITY_REFERENCE (5)
 NODE_NOTATION (12)
 NODE_TEXT (3)
 NODE_PROCESSING_INSTRUCTION (7)

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.childNodes.item(0);
alert(currNode.nodeType);
```

nodeTypedValue Wert des Knotens laut Datentyp, lesen und schreiben

Datentypen sind

VT_BSTR	string
VT_BSTR	number
VT_I4	Int
VT_CY	Fixed.14.4
VT_BOOL	Boolean
VT_DATE	dateTime
VT_DATE	dateTime.tz
VT_DATE	Date
VT_DATE	Time
VT_DATE	Time.tz
VT_I1	i1 byte
VT_I2	i2
VT_I4	i4, int
VT_UI1	ui1
VT_UI2	ui2
VT_UI4	ui4
VT_FLOAT	r4
VT_DOUBLE	r8, float
VT_BSTR	uuid
VT_ARRAY	bin.hex
VT_ARRAY	bin.base64

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.FreeThreadingDOMDocument.3.0");
```



```
XML_DOM_Objekt.documentElement = XML_DOM_Objekt.createElement("Test");
XML_DOM_Objekt.documentElement.dataType = "bin.hex";
XML_DOM_Objekt.documentElement.nodeTypeValue = "ffab123d";
alert(XML_DOM_Objekt.xml);
```

nodeTypeString Knotentyp als String, nur lesen

Knotentypen sind

String	Knotenart
NODE_ATTRIBUTE	attribute
NODE_CDATA_SECTION	cdatasection
NODE_COMMENT	comment
NODE_DOCUMENT	document
NODE_DOCUMENT_FRAGMENT	documentfragment
NODE_DOCUMENT_TYPE	documenttype
NODE_ELEMENT	element
NODE_ENTITY	entity
NODE_ENTITY_REFERENCE	entityreference
NODE_NOTATION	notation
NODE_PROCESSING_INSTRUCTION	processinginstruction
NODE_TEXT	text

Bsp:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.childNodes.item(0);
alert(currNode.nodeTypeString);
```

nodeValue Textwert des Knotens, lesen und schreiben

Knotentypen sind

```
NODE_ATTRIBUTE (2)
NODE_CDATA_SECTION (4)
NODE_COMMENT (8)
NODE_DOCUMENT (9)
NODE_DOCUMENT_TYPE (10)
NODE_DOCUMENT_FRAGMENT (11)
NODE_ELEMENT (1)
NODE_ENTITY (6)
NODE_ENTITY_REFERENCE (5)
NODE_NOTATION (12)
NODE_TEXT (3)
NODE_PROCESSING_INSTRUCTION (7)
```

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.childNodes.item(0);
if (currNode.nodeTypeString == "comment") { alert(currNode.nodeValue); }
```

ondataavailable zu belegen mit Zeiger auf Eventhandler, nur schreiben

onreadystatechange zu belegen mit Zeiger auf Eventhandler, nur schreiben

ontransformnode zu belegen mit Zeiger auf Eventhandler, nur schreiben

ownerDocument Zeiger auf Root des Dokumentes, das Knoten enthält, nur lesen



Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.childNodes.item(0).childNodes.item(1);
var owner = currNode.ownerDocument;
alert(owner.documentElement.tagName);
```

parentNode Zeiger auf Elternknoten, nur lesen

Knotentypen sind

```
NODE_ATTRIBUTE (2)
NODE_CDATA_SECTION (4)
NODE_COMMENT (8)
NODE_DOCUMENT (9)
NODE_DOCUMENT_TYPE (10)
NODE_DOCUMENT_FRAGMENT (11)
NODE_ELEMENT (1)
NODE_ENTITY (6)
NODE_ENTITY_REFERENCE (5)
NODE_NOTATION (12)
NODE_TEXT (3)
NODE_PROCESSING_INSTRUCTION (7)
```

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.childNodes.item(1).childNodes.item(0);
var newNode = currNode.parentNode;
alert(newNode.xml);
```

parsed Status des Parsen eines Knoten, nur lesen

Bsp:

```
// XML-Dokument-Objekt erzeugen mit MSXML-Version 3.0
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
// Daten in das XML-Objekt laden
XML_DOM_Objekt.async = false;
XML_DOM_Objekt.load("xml_datei.xml");
// prüfen ob Daten ohne Fehler nach MSXML-Version 3.0
if (XML_DOM_Objekt.parseError.errorCode != 0)
{ var XML_Datei_Fehler = XML_DOM_Objekt.parseError;
  alert("Datenfehler in xml_datei.xml erkannt: " + XML_Datei_Fehler.reason);
}
else
{ // XML-Daten fehlerfrei
  var root = XML_DOM_Objekt.documentElement;
  alert(root.parsed);
}
```

parseError Fehlerstatus des Parsens eines Knoten, nur lesen

Bsp.:

```
// XML-Dokument-Objekt erzeugen mit MSXML-Version 3.0
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
// Daten in das XML-Objekt laden
XML_DOM_Objekt.async = false;
```



```

XML_DOM_Objekt.load("xml_datei.xml");
// prüfen ob Daten ohne Fehler nach MSXML-Version 3.0
if (XML_DOM_Objekt.parseError.errorCode != 0)
{var XML_Datei_Fehler = XML_DOM_Objekt.parseError;
 alert("Datenfehler in xml_datei.xml erkannt: " + XML_Datei_Fehler.reason);
}
else
{// XML-Daten fehlerfrei
 // Root holen
 var XML_Daten_Root = XML_DOM_Objekt.documentElement;
 // Root abklappen und Textinhalte aller Kinder in der Root anzeigen
 for (var i=0; i<XML_Daten_Root.childNodes.length; i++)
 {alert(XML_Daten_Root.childNodes.item(i).childNodes.item(0).text);}
}

```

prefix Präfix des Namensraumes, nur lesen

preserveWhiteSpace Behandlung von White space, lesen und schreiben

previousSibling vorheriger Eintrag des Knoten in der elterlichen Kontenkindliste, nur lesen

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.childNodes.item(1);
var prevNode = currNode.previousSibling;
alert(prevNode.xml);

```

readyState Status des XML-Dokumentes, nur lesen

Bedeutung der numerischen Werte von readyState

- | | |
|---|---|
| 1 | Laden erfolgt gerade, Daten noch nicht geparkt |
| 2 | Laden erfolgt gerade, Daten geparkt |
| 3 | Laden erfolgt gerade, Daten geparkt, Objekt nicht initialisiert |
| 4 | Laden erfolgt, Objekt initialisiert |
- Es muss .parseError abgefragt werden auf Parser-Error

Bsp.:

```

<script>
var XML_DOM_Objekt;
function Load()
{
 XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
 XML_DOM_Objekt.onreadystatechange = CheckState;
 XML_DOM_Objekt.load(ID_Input_Url.value);
 if (XML_DOM_Objekt.parseError.errorCode != 0)
 {var XML_Datei_Fehler = XML_DOM_Objekt.parseError;
 alert("Dateifehler erkannt: " + XML_Datei_Fehler.reason);
}
}

```

```

function CheckState()
{
 var state = XML_DOM_Objekt.readyState;
 ID_SnzeigeDIV.innerHTML += "readyState = " + state + "<BR>"
 if (state == 4) // numerisch !!
 {
 var err = XML_DOM_Objekt.parseError;
 if (err.errorCode != 0)

```



```

        {ID_SnzeigeDIV.innerHTML += err.reason + "<BR>"}
        {else ID_SnzeigeDIV.innerHTML +=", also erfolgreich" + "<BR>"}
    }
}
<script>
Schritt 1 Bitte Url des XML-Dokumentes eingeben: <input type=text size=60 id='ID_Input_Url'>
Schritt 2 Klicken f&uuml;l; Laden des XML-Dokumentes <input type=button value='Laden'
                onclick="javascript:Load()">
<div id='ID_SnzeigeDIV'></div>

```

resolveExternals Entfernung von externen Definitionen zum Parsenzeitpunkt, lesen und schreiben

specified Knoten definiert als Wert in DTD oder als Schema, lesen

text Text eines Knotens, lesen und schreiben

nur bei Knoten NODE_TEXT

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.childNodes.item(0);
alert(currNode.text);

```

url Orl des geladenen XML-Dokumentes, lesen

wird nur nach erfolgreichem Laden gefüllt (nicht nach save)
ist null falls XML-Daten nicht als Datei geladen

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
alert(XML_DOM_Objekt.url);

```

validateOnParse Status der Auswertung des Dokumentes, lesen und schreiben

```

boolValue = oXMLDOMDocument.validateOnParse;
objXMLDOMDocument.validateOnParse = boolValue;

```

boolValue true so Validate während Parsen, Standard
false so nur prüfen ob XML-Daten wohlgeformt sind

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
XML_DOM_Objekt.validateOnParse = true;
XML_DOM_Objekt.load("xml_datei.xml");

```

xml XML-Repräsentation eines Knoten, lesen

liefert String aus Unicode
z.B. <?xml version="1.0" encoding="UTF-8"?>

geliefert wird aber <?xml version="1.0"?>
da UTF-8 nicht Unicode ist

bei save wird der aktuelle Zeichensatz des Encodens verwendet
z.B. wenn 1252 aktiv so auch 1251 benutzt beim save

Bsp.:




```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.childNodes.item(0);
alert(currNode.xml);
```

3.3. Methoden - Übersicht

Funktionen sind teilweise analog zum HTML-DOM bzw. werden aus dem HTML-DOM übernommen

abort asynchronen Datendownload abbrechen

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
XML_DOM_Objekt.async = true;
XML_DOM_Objekt.onreadystatechange = doOnReadyStateChange;
XML_DOM_Objekt.load("xml_datei.xml");
if (XML_DOM_Objekt.parseError.errorCode != 0)
{ var XML_Datei_Fehler = XML_DOM_Objekt.parseError;
  alert("Dateifehler: " + XML_Datei_Fehler.reason);
}

function doOnReadyStateChange ()
{ if (XML_DOM_Objekt.readyState == 1)
  { XML_DOM_Objekt.abort(); }
}
```

appendChild bereits erzeugtes Kind hinter das letzte Kind anhängen
Achtung: Man sollte validate-Methode vorher benutzen !

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
var newNode = XML_DOM_Objekt.createElement(1, "newChild", "");
root.appendChild(newNode);
```

cloneNode Knoten klonen

```
var objXMLDOMNode = oXMLDOMNode.cloneNode(boolValue);
```

boolValue true so Knoten und alle Kinder klonen
false so Knoten ohne Kinder klonen

Achtung: DOMDocument-Knoten besser nicht klonen sondern sichern per save ()

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
var currNode = root.childNodes.item(1);
var MyNewNode = currNode.cloneNode(true);
root.appendChild(MyNewNode);
alert(XML_DOM_Objekt.xml);
```

createAttribute Attribut für einen Knoten erzeugen

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
```



```
var newAtt = XML_DOM_Objekt.createAttribute("ID");
```

createCDATASection CDATA-Section-Knoten erzeugen

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
var CDATASection = XML_DOM_Objekt.createCDATASection("Hello World!");
root.appendChild(CDATASection);
alert(root.xml);
```

createComment Comment-Knoten erzeugen

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
var comment = XML_DOM_Objekt.createComment("Hello World!");
root.appendChild(comment);
alert(root.xml);
```

createDocumentFragment leere IXMLDOMDocumentFragment-Object erzeugen

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var docFragment = XML_DOM_Objekt.createDocumentFragment();
docFragment.appendChild(XML_DOM_Objekt.createElement("node1"));
docFragment.appendChild(XML_DOM_Objekt.createElement("node2"));
docFragment.appendChild(XML_DOM_Objekt.createElement("node3"));
alert("fragment: " + docFragment.xml);
XML_DOM_Objekt.documentElement.appendChild(docFragment);
alert("document: " + XML_DOM_Objekt.xml);
```

createElement Element erzeugen, aber nicht anhängen

createElement fügt Objekt nicht in DOM ein, sondern nur
appendChild()
replaceChild()
insertBefore()

Knotentyp ist NODE_ELEMENT

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
var newElem = XML_DOM_Objekt.createElement("NEW");
root.childNodes.item(0).appendChild(newElem);
root.childNodes.item(0).lastChild.text = "123";
alert(root.childNodes.item(0).xml);
```

createEntityReference EntityReference-Knoten erzeugen

createNode Knoten erzeugen

Achtung: Man sollte validate-Methode vorher benutzen !

Knotentypen sind



NODE_ATTRIBUTE (2)
 NODE_CDATA_SECTION (4)
 NODE_COMMENT (8)
 NODE_DOCUMENT (9)
 NODE_DOCUMENT_TYPE (10)
 NODE_DOCUMENT_FRAGMENT (11)
 NODE_ELEMENT (1)
 NODE_ENTITY (6)
 NODE_ENTITY_REFERENCE (5)
 NODE_NOTATION (12)
 NODE_TEXT (3)
 NODE_PROCESSING_INSTRUCTION (7)

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
var newNode = XML_DOM_Objekt.createElement(1, "newChild", "");
root.appendChild(newNode);
  
```

createProcessingInstruction Processing instruction-Knoten erzeugen

createTextNode Text-Knoten erzeugen

Knotentyp ist NODE_TEXT

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
var MyText = XML_DOM_Objekt.createTextNode("Hello World");
var MyNewNode = root.insertBefore(MyText, root.childNodes.item(0));
alert(XML_DOM_Objekt.xml);
  
```

getElementsByTagName Zeiger auf Element holen

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var objNodeList = XML_DOM_Objekt.getElementsByTagName("author");
  
```

hasChildNodes ermitteln ob Kindknoten vorhanden sind

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var currNode = XML_DOM_Objekt.documentElement.firstChild;
if (currNode.hasChildNodes())
{ alert(currNode.childNodes.length); }
else { alert("no child nodes"); }
  
```

insertBefore Kind einfügen

Achtung: Man sollte validate-Methode vorher benutzen !

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
  
```



```
var newNode = XML_DOM_Objekt.createNode(1, "CHILD2", "");
var currNode = root.insertBefore(newNode, root.childNodes.item(1));
```

load

XML-Dokument laden

```
boolValue = oXMLDOMDocument.load(xmlSource);
```

xmlSource Url einer XML-Datei

```
boolValue      true so XML-Datei geladen
                 es muss aber .parseError noch verwendet werden
```

Es muss .async auf false gesetzt sein.

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
XML_DOM_Objekt.async = false;
XML_DOM_Objekt.load("xml_datei.xml");
if (XML_DOM_Objekt.parseError.errorCode != 0)
{ var XML_Datei_Fehler = XML_DOM_Objekt.parseError;
  alert("Dateifehler: " + XML_Datei_Fehler.reason);
}
else { alert(XML_DOM_Objekt.xml); }
```

loadXML

XML-dokument laden

```
boolValue = oXMLDOMDocument.loadXML(strXML);
```

```
strXML XML-String als Dokument oder Dokumentfragment
        nur UTF-16 oder UCS-2-Econdings
```

```
boolValue true so String geladen, es muss aber .parseError noch verwendet werden
```

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
XML_DOM_Objekt.async = false;
XML_DOM_Objekt.loadXML(
  "<customer><first_name>Joe</first_name><last_name>Smith</last_name></customer>");
if (XML_DOM_Objekt.parseError.errorCode != 0)
{ var XML_Datei_Fehler = XML_DOM_Objekt.parseError;
  alert("Dateifehler: " + XML_Datei_Fehler.reason);
}
else { alert(XML_DOM_Objekt.xml); }
```

nodeFromID

ID-Attribut liefern wenn vorhanden

removeChild

Kind entfernen aus XML-DOM

Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
var currNode = root.childNodes.item(1);
var oldChild = currNode.removeChild(currNode.childNodes.item(1));
alert(oldChild.text);
```

replaceChild

Kind ersetzen

Bsp.:



```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var root = XML_DOM_Objekt.documentElement;
var newElem = XML_DOM_Objekt.createElement("PAGES");
root.childNodes.item(1).replaceChild(newElem,root.childNodes.item(1).childNodes.item(0));
alert(root.childNodes.item(1).xml);

```

save

XML-Dokument speichern

oXMLDOMDocument.save(destination);

destination String Url z.B. 'c:\test\text.xml'
 ASP-Response-Objekt
 DOMDocument-Objekt
 benutzerdefiniertes Objekt

Es wird UTF-8 verwendet wenn nichts anderes kodiert

<?xml version="1.0" encoding="windows-1252"?>

1252 verwenden
 kein UTF-8

Achtung: Man sollte validate-Methode vorher benutzen !

Bsp.:

```

var XML_DOM_Objekt1 = new ActiveXObject("Msxml2.DOMDocument.3.0");
var XML_DOM_Objekt2 = new ActiveXObject("Msxml2.DOMDocument.3.0");
XML_DOM_Objekt1.load("sample.xml");
XML_DOM_Objekt1.save(XML_DOM_Objekt2.XML_DOM_Objektument);

```

selectNodes

Zeiger auf IXMLDOMNodeList

selectSingleNode

Erster ausgewerteter Knoten (pattern-matching operation)

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
XML_DOM_Objekt.setProperty("SelectionLanguage", "XPath");
var currNode = XML_DOM_Objekt.selectSingleNode("//book/author");
alert(currNode.text);

```

setAttributeNode

Attributknoten setzen oder updaten

Bsp.:

```

var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
var nodePublishDate = XML_DOM_Objekt.createAttribute("PublishDate");
nodePublishDate.value = String(Date());
var nodeBook = XML_DOM_Objekt.selectSingleNode("//book");
nodeBook.setAttributeNode(nodePublishDate);
alert(nodeBook.getAttribute("PublishDate"));

```

transformNode

Knoten und seine Kinder per XSLT styl sheet transformieren

transformNodeToObject

Knoten und seine Kinder per XSLT styl sheet transformieren zu Objekt

validate

Validierung anhand DTD, Schema oder Schema-Collection

ab MS XML 6.0 werden nur noch DTD und XSD-Schemas verwendet
 Validierung nur aufrufbar, wenn readyState == 4 bereits erkannt wurde



Bsp.:

```
var XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
XML_DOM_Objekt.async = false;
XML_DOM_Objekt.validateOnParse = true;
XML_DOM_Objekt.load("http://server/myData.xml");
XML_DOM_Objekt.documentElement.setAttribute("a", "123");
var err = XML_DOM_Objekt.validate();
if (err.errorCode == 0){alert("Document is valid");}
else{alert("Validation error:" + err.reason);}
```

3.4. Events - Übersicht

ondataavailable	Event erzeugt wenn neue Daten verfügbar sind
onreadystatechange	Event erzeugt wenn Status des XML-Dokumentes readyState sich ändert

Bedeutung der numerischen Werte von readyState

- | | |
|---|---|
| 1 | Laden erfolgt gerade, Daten noch nicht geparkt |
| 2 | Laden erfolgt gerade, Daten geparkt |
| 3 | Laden erfolgt gerade, Daten geparkt, Objekt nicht initialisiert |
| 4 | Laden erfolgt, Objekt initialisiert |
- Es muss .parseError abgefragt werden auf Parser-Error

Bsp.:

```
<script>
var XML_DOM_Objekt;
function Load()
{
    XML_DOM_Objekt = new ActiveXObject("Msxml2.DOMDocument.3.0");
    XML_DOM_Objekt.onreadystatechange = CheckState;
    XML_DOM_Objekt.load(ID_Input_Url.value);
    if (XML_DOM_Objekt.parseError.errorCode != 0)
    {var XML_Datei_Fehler = XML_DOM_Objekt.parseError;
    alert("Dateifehler erkannt: " + XML_Datei_Fehler.reason);
    }
}

function CheckState()
{
    var state = XML_DOM_Objekt.readyState;
    ID_SnzeigeDIV.innerHTML += "readyState = " + state + "<BR>"
    if (state == 4)    // numerisch !!
    {
        var err = XML_DOM_Objekt.parseError;
        if (err.errorCode != 0)
        {ID_SnzeigeDIV.innerHTML += err.reason + "<BR>"}
        {else ID_SnzeigeDIV.innerHTML += ", also erfolgreich" + "<BR>"}
    }
}
</script>
Schritt 1 Bitte Url des XML-Dokumentes eingeben: <input type=text size=60 id='ID_Input_Url'>
Schritt 2 Klicken f&uuml;l; Laden des XML-Dokumentes <input type=button value='Laden'
onclick="javascript:Load()">
<div id='ID_SnzeigeDIV'></div>
```

ontransformnode	Event erzeugt bevor ein Style-Sheet einem Knoten zugewiesen wurde
-----------------	---



4. Index

"attribute"	15	.nodeTypeString.....	15, 21
"cdatasection"	15	.nodeValue	21
"comment"	16	.normalize	7
"document"	16	.ondataavailable	21
"documentfragment"	17	.onreadystatechange	9, 21
"doctype"	17	.ontransformnode	21
"element"	15	.open.....	12
"entity"	16	.ownerDocument.....	21
"entityreference"	15	.parentElement	6
"notation"	17	.parentNode.....	22
"processinginstruction"	16	.parsed	22
"text"	15	.parseError.....	22
.abort.....	11, 25	.prefix.....	23
.addBehavior	7	.preserveWhiteSpace.....	23
.appendChild	25	.previousSibling	23
.async.....	17	.readyState.....	6, 9, 23
.attributes	17	.recordset.....	6
.baseName	17	.removeAttributeNode	7
.canHaveHTML.....	6	.removeBehavior	7
.childNodes	17	.removeChild.....	28
.cloneNode	25	.replaceChild	28
.componentFromPoint	7	.resolveExternals	24
.createAttribute	25	.responseBody	9
.createCDATASection.....	26	.responseText	10
.createComment.....	26	.responseXML.....	10
.createDocumentFragment	26	.save	29
.createElement	26	.scopeName.....	6
.createEntityReference	26	.selectNodes	29
.createNode.....	26	.send	12
.createProcessingInstruction.....	27	.setAttributeNode	7, 29
.createTextNode	27	.setRequestHeader.....	13
.dataType	18	.specified	24
.definition	18	.status	10
.doctype	18	.statusText	11
.documentElement.....	8, 18	.tagUrn	6
.fireEvent	7	.text	24
.firstChild	19	.transformNode	29
.getAllResponseHeaders.....	11	.transformNodeToObject	29
.getAttributeNode	7	.validate	29
.getElementsByTagName	27	.validateOnParse	24
.getResponseHeader	12	.xml	24
.hasChildNodes	27	.XMLDocument	7
.id.....	6	abort	11, 25
.implementation.....	19	addBehavior	7
.insertBefore	27	Ajax für ASP.Net Framework.....	13
.isContentEditable	6	appendChild	25
.isDisabled	6	'ASP.NET AJAX	14
.isMultiLine	6	ASP.Net Framework	13
.lastChild	19	async	17
.load	28	attributes	17
.loadXML	28	baseName.....	17
.namedRecordset	7	behavior	7
.namespaceURI	19	canHaveHTML	6
.nextSibling	19	CDATA-Section	26
.nodeFromID	28	childNodes	17
.nodeName	19	cloneNode	25
.nodeType	20	componentFromPoint.....	7
.nodeTypedValue	20	COPY.....	12



createAttribute	25	NODE_NOTATION	17
createCDATASection	26	NODE_PROCESSING_INSTRUCTION	16
createComment	26	NODE_TEXT	15
createDocumentFragment	26	nodeFromID	28
createElement	26	nodeName	19
createEntityReference	26	nodeType	20
createNode	26	nodeTypedValue	20
createProcessingInstruction	27	nodeTypeString	15, 21
createTextNode	27	nodeValue	21
Data Islands	5	normalize	7
dataType	18	Objekt XML	5
Dateninseln	5	ondataavailable	7, 21, 30
definition	18	ondatasetchanged	7
DELETE	12	ondatasetcomplete	7
doctype	18	onreadystatechange	7, 9, 13, 21, 30
documentElement	8, 18	onrowenter	7
fireEvent	7	onrowexit	7
firstChild	19	onrowsdelete	7
GET	12	onrowsinserted	7
getAllResponseHeaders	11	ontransformnode	21, 30
getAttributeNode	7	open	12
getElementsByTagName	27	OPTIONS	12
getResponseHeader	12	ownerDocument	21
hasChildNodes	27	parentElement	6
HEAD	12	parentNode	22
HTML-DOM	4	parsed	22
HTTP Methode	12	parseError	22
HTTP-Header	12	POST	12
id	6	prefix	23
implementation	19	preserveWhiteSpace	23
insertBefore	27	previousSibling	23
isContentEditable	6	PROPFIND	12
isDisabled	6	PROPPATCH	12
isMultiLine	6	PUT	12
IXMLDOMDocumentFragment	26	readyState	6, 9, 23
IXMLDOMElement	8	recordset	6
IXMLDOMNodeType	15	removeAttributeNode	7
Knotenarten	15	removeBehavior	7
LANGUAGE="XML"	5	removeChild	28
lastChild	19	replaceChild	28
load	28	resolveExternals	24
loadXML	28	responseBody	9
LOCK	12	responseText	10
Microsoft Ajax für ASP.Net Framework	13	responseXML	10
MKCOL	12	Root des XML-Baumes	7
MOVE	12	save	29
Mxml2.DOMDocument.3.0	8	scopeName	6
namedRecordset	7	selectNodes	29
namespaceURI	19	send	12
new ActiveXObject("Mxml2.DOMDocument.3.0")	8	setAttributeNode	7, 29
nextSibling	19	setRequestHeader	13
NODE_ATTRIBUTE	15	specified	24
NODE_CDATA_SECTION	15	src	6
NODE_COMMENT	16	status	10
NODE_DOCUMENT	16	statusText	11
NODE_DOCUMENT_FRAGMENT	17	tagUrn	6
NODE_DOCUMENT_TYPE	17	text	24
NODE_ELEMENT	15	text/xml	5
NODE_ENTITY	16	textAutospace	7
NODE_ENTITY_REFERENCE	15	textUnderlinePosition	7



transformNode.....	29	XML-Baum.....	7
transformNodeToObject.....	29	XML-Dateninsel	4
TYPE="text/xml"	5	XMLDocument	7
UNLOCK	12	XML-Dokument-Root	7
validate	29	XMLHttpRequest.....	9
validateOnParse.....	24	XMLHttpRequest-Objekt	4
Visual Web Developer	13	XML-Objekt	5
Visual Web Developer Expression.....	13	XPath	4
xml.....	24	XSD	4
XML Schema	4	XSLT	4

