

```

var DA_Produkt2 = DA_Bibliothek.Mul(
    ZufallsWerteAlsFolgeErzeugen(KaroPositionsWechsel_Traegheit * Math.random()),
    DA_KaroAbstandFaktor
);

var DA_2DPunkt_Zufall = DA_Bibliothek.Translate2Anim(DA_Produkt1,DA_Produkt2);

// - - - Karo zufällig positionieren aber relativ zum Zentrum der Animation
var DA_2DKomposition = DA_Bibliothek.Compose2( DA_2DPunkt_Zufall,
    DA_2DPunkt_AnimationZentrum
);

// - - - und Karo nach 2D konvertieren
DA_Bild_Neu = DA_Bild_Neu.Transform(DA_2DKomposition);

// - - - Karo dem DA-Bild-Gesamt hinzufügen
DA_Bild_Gesamt = DA_Bibliothek.Overlay(DA_Bild_Gesamt, DA_Bild_Neu);
}

// ##### DA-Init #####
DACControl.Image = DA_Bild_Gesamt;

// ##### Start der Animation #####
DACControl.Start();
}
-->
</SCRIPT>
</HEAD>
<BODY onload="DirectAnimationStart();">
  <OBJECT ID="DACControl"
    CLASSID="CLSID:B6FFC24C-7E13-11D0-9B47-00C04FC2F51D"
    STYLE="position:absolute; left:30%; top:100;width:300;height:300"
  >
  </OBJECT>
</BODY>
</HTML>

```

5.2.3. JScript Laufzeit-Bibliothek (ActiveXObject) des Internet Explorer

Das Objekt ermöglicht den Zugriff auf die JScript-Laufzeit-Bibliothek als Erweiterung von JScript.

Dieses Objekt ist in der JScript-Maschine aufgrund von Active-X integriert und hat eigentlich nichts mit der DOM-Hierarchie des HTML-Dokumentes zu tun.

Die JScript-Erweiterung berührt auch das DOM, denn alle Bibliotheken-Elemente sind in eine Webseite implementierbar.

Es gibt zwei Hauptkomponenten der Bibliothek:

Dictionary	als Verwaltung von per Schlüssel indizierten Stringdaten per JScript als Addon zum Internet Explorer analog zur Datenbankverwaltung per Textdatei
FileSystemObject	als Verwaltung des Dateisystems im Betriebssystem per JScript als Addon zum Internet Explorer analog zu einer abgespeckten Dateiverwaltung per DOS

Warnung zum FileSystemObject:

Die Firewall müsste Zugriffe beanstanden (Abfrage auf Erlaubnis der Zugriffe auf die Festplatte).

Die Nutzung dieses Objektes berührt unmittelbar die Privatsphäre des Users. Die Nutzung des Objektes kann rechtlich relevant werden, so dass für den User Transparenz bezüglich der Objektverwendung vorliegen muss.

Ein User, der das Active-X-Control mit oder ohne sein Wissen zulässt, geht definitiv das Risiko einer missbräuchlichen Nutzung des Objektes ein, wenn kein Virenschanner die Scripte vor deren Aktivierung prüft und nach User-Entscheidung oder generell blockiert und somit das Privateigentum des Users schützt.

Eine aktive Firewall mit integriertem Virenschanner kann Scriptanweisungen mit Bezug auf das FileSystemObject als virenverseuchtes Script erkennen und die Scriptaktivierung verhindern. Falls der Virenschanner auch E-Mail scannen kann, in denen o.g. Scriptanweisungen vorliegen, könnte eine betroffene Email automatisch bereinigt und damit inhaltlich verändert werden. Der Test o.g. Scriptanweisungen ist also nur bei abgeschalteter Firewall und Virenschanner möglich. Scriptanweisungen mit Bezug auf das FileSystemObject sind absolute Ausnahme im Webdesign, so dass eine Webseite, die unkommentiert das FileSystemObject benutzt, als Virenangriff angesehen werden sollte.

Microsoft ändert fortlaufend die Active-X-Eigenschaften von Windows und somit auch des Internet Explorers



Diese fortlaufenden Änderungen muss der Programmierer in Erfahrung bringen.

Der Programmierer kann sich definitiv nicht auf Verfügbarkeit von Active-X-Controls verlassen und muss damit rechnen, dass seine Webseiten schlagartig nicht mehr komplett laufen weil u.a. Programmcode noch nicht angepasst ist. Ebenfalls muss der Programmierer Varianten von Windows und Patchzustände beachten, die prinzipiell Kostenprobleme verursachen können.

Mit anderen Worten: Wer Microsoft-Komponenten nutzt, muss wissen, was ihm blüht ... siehe nachfolgende Beispiel für Risiken.

Prinzipielle Lizenzprobleme für den Programmierer

Microsoft verlangt Lizenzierung von Windows. Bezüglich Windows-Versionen gibt es die Updatestufen z.B. per Servicepacks

Ein Windows mit Servicepack fällt unter die Lizenz des geupdateten Windows.

Ein Windows mit Vorversion zum Servicepack bedarf einer anderen Lizenz.

Will man z.B. den Internet Explorer 7 und 6 parallel testen, benötigt man 2 Windowslizenzen, da beide Versionen nicht parallel installierbar. Dazu kommt, dass es den IE 6 in 2 Versionen gibt: Win SP1 und SP2 (IE 7 nur ab Win SP2).

Für 3 Browserversionen benötigt man 3 Windowslizenzen, will man parallel testen.

Ein Blick auf Browser-Konkurrenzprodukte klärt die Sachlage unschlagbar: Opera ist z.B. parallel installierbar.

Hinweis: Man suche doch mal im Internet nach einem kostenlosen HTTP-Server vom Microsoft, um IE-Seite testen zu können, die JScript nutzen (inklusive Debugger). Denn sollte kein kostenloses Angebot findbar sein, kommen die Kosten von Entwicklungssoftware zum IE hinzu. Ein Blick auf Konkurrenz-HTTP-Server klärt die Sachlage: Apache-HTTP-Server ist kostenlos, allerdings nicht einfach einzurichten (Hinweis: Der HTTP-Server sollte virtuelle Hosts einrichten können und korrekt mit der Firewall des Users zusammenarbeiten können).

Abänderungen wegen Sicherheitspatches der jeweiligen Windows-Versionen

Abschaltungen von Active-X-Controls erfolgen auch im Rahmen der Sicherheitspatches zu Windows-Versionen.

Es ist auch möglich, dass wegen Sicherheitslücken abgeschaltet wird und somit Komponenten einer Webseite je nach Windowsversion nicht mehr laufen.

Im Rahmen der Sicherheitspatches ist es Microsoft sogar gelungen, Webseiten, die den MS-Encoder zur Komprimierung von HTML- und JScript-Code nutzen, schlagartig unnutzbar zu machen: Ein Bug in einem Patch zu Windows XP - Q918899 Das Patch verursacht IE-Browser-Absturz bei per MS ScriptEncoder gepacktem JScript unter SP1 und 2 wenn HTTP 1.1 mit Kompression genutzt wird z.B. bei onclick-Handler auf IMG klick ins Fenster per aktivem Popup

Der Absturz ist "read" -Fehler von immer ein und derselben Speicherstelle.

User, die dieses Patch installiert haben, können ab sofort keine IE-Seiten mit codiertem Script mehr ansehen.

Microsoft stellt Abhilfe nach geraumer Zeit zur Verfügung, jedoch spezifisch nach Windows XP-Version:

Patch Q918899 für

Windows XP SP1Download für jedermann bereitgestellt

SP2 nur auf kostenpflichtige telefonische Anfrage des Users per Downloadlink bereitgestellt, da

Microsoft explizit die User registriert haben will, bei denen das

Patchproblem auftritt (User muss sich Telefonnummer besorgen)

Solange also das Patch zum fehlerhaften Patch vom User nicht installiert wird,

z.B. weil der User keine Ahnung hat, dass und wo er sich die Telefonnummer

von Microsoft besorgen muss bzw. zu besorgen hat, wird der User

IE-Seiten mit komprimierten Code dauerhaft nicht nutzen können.

(Microsoft-Support ist z.T. nur in Englisch).

Abänderungen wegen Browser-Inkompatibilität

Popupblocker-Fehler

Die Microsoft Browser-Version IE 7 ist nicht abwärtskompatibel bezüglich Popup per window.createPopup()

Popup per window-Objekt ist ein Markenzeichen des IE, das im IE 7 nicht mehr fehlerfrei nutzbar ist.

Der Fehler liegt in der Popup-Blockerverwaltung des IE und wurde mit dem IE 7 implementiert.

Der Fehler tritt nicht auf, wenn ein Fenster per window.open() erzeugt wurde.

Bedingung:

Scriptfehleranzeige ist erlaubt im IE 7

Popupblocker ist im IE abgeschaltet

ein aktives Fenster (Register) mit Dokument, dass fortlaufend (rekursiv) genau 1 window.popup per .show()erzeugt.

ein weiteres Fenster (Register) z.B. leere Seite (about:blank)

beide (Register) liegen in einer gemeinsamen IE-Instanz

Ablauf: Wird Focus auf Register der leeren Seite gehalten und wird parallel das Popup per .show() erzeugt,

bricht der Browser das Dokument mit .show() ab (Scriptfehler).

Der Popupblocker für die leere Seite verursacht den Programmfehler im Dokument mit .show(). Es wird folgende

Meldung angezeigt (in der Informationsleiste):

'Ein Popup wurde geblockt. Klicken Sie hier, um das Popup bzw. weitere Optionen anzuzeigen.'

Die Bedeutung der Meldung laut Microsoft-Hilfe im IE 7:

Der Popupblocker hat ein Populfenster geblockt. Sie können den Popupblocker deaktivieren

oder Popups temporär zulassen, indem Sie auf die Informationsleiste klicken.

Die Realität zur obigen Meldung ist völlig anders:

Linke oder rechte Maus auf die Meldung liefert z.B. Einstellungen darunter



Popupblocker einschalten
 weitere Informationen
 jedoch keine Möglichkeit wie laut Bedeutung
 Damit gilt: Der abgeschaltete Popupblocker ist in Wirklichkeit aktiv.
 Pikant: Ein Popup erscheint normalerweise auch über fremde Fenster, die nicht das Popup erzeugt haben (z.B. Fenster einer Windowsanwendung z.B. einer anderen IE-Instanz)
 Der Popupblocker des IE bemerkt aber NUR Webseite, die das Popup erzeugt.
 Durch das Abwürgen von Popup wird das Popup natürlich auf und für anderen Seiten nicht relevant; im Falle einer anderen IE-Instanz also auch für diese nicht relevant, obwohl diese Instanz per Popupblocker verwaltet wird.
 Der Popupblocker beschneidet die Popup-Reichweite an der Wurzel, ist aber nicht objektorientiert zu den anderen Webseiten (die nicht das Popup erzeugt haben).
 Der Popupblocker ist nicht als Filter aufgesetzt sondern reingestrickt worden.
 Der Popupblockerfehler verändert die Eventverwaltung:
 Es werden u.a. ignoriert
 onfocus
 onblur
 onfocusin
 onfocusout
 und viele andere, so dass trotz Events z.B. des Body der Popupblockerfehler entsteht.

 // nachfolgender Code setzt focus nicht neu: Fenstereintrag in Taskleiste blinkt eventuell
 window.focus();
 window.document.focus();
 if(document.body!=null)
 {if(document.body.style!='hidden') // wenn hidden so focus() nicht möglich (Scriptfehler erzeugt)
 {document.body.focus();}
 }
 // wenn paralleles Fenster offen (on oder offline), so Scriptfehler erzeugt
 popupzeiger.show(...);

Nachfolgend eine Testdokument

```

<BODY>
<SCRIPT LANGUAGE="JScript">
// #####
//
//           Quelltext NICHT ändern wegen Fehlermeldung (siehe unten)
//
// #####

// Warnung: window.popup arbeitet im IE 7 fehlerhaft wegen Popupblocker-Fehler:
//   Bedingung:  Scripfehleranzeige ist erlaubt im IE 7
//               Popupblocker ist abgeschaltet
//               Ein Fenster mit Dokument aktiv, dass fortlaufend genau 1 window.popup neu erzeugt (per Rekursion)
//               Ein paralleles Fenster z.B. Leere Seite (about:blank) oder mit echtem Internetzugriff
//               das dauerhaft Focus haben muss
//               Beide Fenster sind Register in einer gemeinsamen IE-Instanz
// Solange Focus: Irgendwann erscheint eine Meldung des IE 7 per gelber Zeile
//               im Fenster, das das window.popup erzeugt (Meldung kommt nicht, wenn
//               Focus auf Fenster, das das window.popup erzeugt)
//
// 'Ein Popup wurde geblockt. Klicken Sie hier, um das Popup bzw. weitere Optionen anzuzeigen.'
//
// Bedeutung laut Microsoft-Hilfe:
//   Der Popupblocker hat ein Populfenster geblockt. Sie können den Popupblocker deaktivieren
//   oder Popups temporär zulassen, indem Sie auf die Informationsleiste klicken.
//
// Realität: linke oder rechte Maus liefert   z.B.   Einstellungen darunter Popupblocker einschalten
//                                               weitere
Informationen
//
//           jedoch keine Möglichkeit wie laut Bedeutung
//
//           Damit gilt: Der abgeschaltete Popupblocker ist aktiv.
//
//           Blockiertes window.popup:
//
//           Blockierung stoppt Abarbeitung des Scriptes, das das popup rekursiv erzeugt (stopp der
Rekursion)
//           und bewirkt zugleich Meldung eines Scriptfehlers an Quelltext-Stelle zu .show()
//
// #####
//
//           Es wird Zeile 55 in Spalte 2 als Fehlerstelle ausgeben: Unbekannter Fehler
  
```



```
//
// #####

var XPopupZeiger=window.createPopup();
XPopupZeiger.document.body.innerHTML='##### Test #####';
var XLeft=700;
var XTop=400;
var XTimeoutID1=0;
var XTimeoutID2=0;
var XAnzeigeErlaubt=false;

function Y_PopupShow()
{XLeft-=10;if(XLeft<50){XLeft=700;}
XTop-=10;if(XTop<50){XTop=400;}
XPopupZeiger.show(XLeft,XTop,300,300,document.body);
// Popup wird immer innerhalb Screen-Dimension angezeigt, wenn XLeft und /oder XTop
// ausserhalb der Screen-Dimension
if(XAnzeigeErlaubt){ XTimeoutID1=window.setTimeout('Y_PopupHide()','3000);}
}

function Y_PopupHide()
{XPopupZeiger.hide();
if(XAnzeigeErlaubt){XTimeoutID2=window.setTimeout('Y_PopupShow()','1000);}
}

function YAnzeigeStarten()
{XAnzeigeErlaubt=true;
Y_PopupShow();
}

function YAnzeigeStoppen()
{XAnzeigeErlaubt=false;}
</SCRIPT>

<INPUT type="button" value="Start" onclick="YAnzeigeStarten();">
<BR>
<BR>
<INPUT type="button" value="Stop" onclick="YAnzeigeStoppen();">
</BODY>
```

Hinweis: Der Popupfehler ist so elementar, dass die vielen Beta-Testphasen des IE mehr als fragwürdig erscheinen, wie die Angabe von Microsoft, dass Code neu programmiert wurde, um den IE sicherer zu machen.

focus-Methode beim IE 7

windows.focus() document.focus() und body.focus() funktionieren NICHT
zwischen Register in einem IE-Fenster
zwischen Fensters z.B. in Taskleiste

Hinweis:

- .focus() setzt Element aktiv, gibt dem Element den Focus und feuert dann onfocus
- .setActive() ist Teilmenge von .focus(): nur das aktiv setzen
- funktioniert nicht mit allen Elementen, mit denen .focus() funktioniert

animierte Gif (mit Timer)

Animierte Gifs (mit Timer), die unter IE 6 korrekt laufen, müssen unter IE 7 im Timer nicht mehr laufen:
z.B. garnicht mehr sichtbar, oder Timer nicht verwendet.
Dann müssen animierte Gif-Bilder nach IE-Version bereitgestellt werden.

Abänderungen wegen Rechtstreitigkeiten von Microsoft mit Fremdanbietern

Ein sehr bekanntes Beispiel ist die nachträglich eingeführte Einschränkung von Active-X-Controls wegen Patentwahrung durch Microsoft, wobei für den JScript-Programmierer massive Änderungen eintreten.

Wegen Patentwahrung hat Microsoft ein zunächst freiwilliges Patch herausgegeben, dass bei ActiveX-Control per APPLET, EMBED oder OBJECT, die auf dem Bildschirm rendern (mit oder ohne Userschnittstelle), dafür sorgt, dass bei mouseover über das Control eine Sprechblase erscheint, die darauf hinweist, dass das Objekt als ActiveX-Control klickbar ist. Diese Sprechblase erscheint auch, wenn das Control keine Userschnittstelle hat, also diese gar nicht klickbar ist.

Es wurde das Eventmodell gleichzeitig geändert:

Es werden alle Events solange unterdrückt, bis der User die Sprechblase geklickt hat.



Das Klicken muss auf das Objekt im Sprechblasenrahmen erfolgen, der so groß ist, wie die Dimension, in der gerendert wurde.

Es muss also ERST per Mausklick das Control aktiviert werden, ehe das Control klickbar und damit die Eventsteuerung aktiviert ist.

Ein Control, das programmtechnisch zwar was rendert, aber ansonsten ohne sichtbare programmtechnisch startet, muss ebenfalls geklickt werden, obwohl es bereits läuft und es nichts zu klicken gäbe (wenn keine Eventsteuerung eingebaut wurde).

Wegen blockierter Eventsteuerung ist also die Sprechblase z.B. nicht automatisch klickbar.

Die Eventauslösung per nicht-objekteigenen Eventhandler, der für das Objekt per fireEvent() ein Event auslöst, ist solange blockiert, bis der User die Sprechblase geklickt hat.

style.visibility='hidden' wird ignoriert

Die Sprechblase erscheint auch dann, wenn das Control mit style.visibility='hidden' belegt ist, also sich unsichtbar rendert:

Der Sprechblasenrahmen hat genau die Dimension wie die des unsichtbaren Controls. Der Sprechblasenrahmen erscheint also Zusammenhangslos, und der User weiß nicht, warum er klicken soll, wenn er nichts sieht. Vor allem weiß er nicht, WAS er klickt ... ideale Basis für Schadsoftware per Script.

Diese Sprechblase erscheint nur DANN NICHT, wenn die Userschnittstelle mit Breite == Höhe == 0 gerendert wird. Sollte die Userschnittstelle in einem Container liegen, z.B. DIV, dann wird der Container, wenn er in der Dimension kleiner ist, also die Userschnittstelle, angepasst. Daher muss der Container ebenfalls mit Breite == Höhe == 0 gerendert werden. Wegen Dimensionierung auf 0 sollte style.visibility="hidden" sein. Im Falle eines Containers reicht es, den style des Containers zu ändern, da visibility normalerweise vererbt wird an Kinder, also auch an das Control.

Abänderung wegen Abschaltungen

DirectX ist wegen Abschaltung von Active-X--Controls nicht mehr abwärtskompatibel:

Z.B. wurde bei Win XP SP2 Direct Animation aus DirectX schlagartig durch Abschaltung von Bibliotheken dezimiert, die es bei Win XP SP1 aber noch gibt.

Hier ein Beispiel aus dem Jahr 2004: Abschaltungen von Active-X-Controls

ActiveX-Controls und Unterstützung/Verbot 20041215

erlaubt sind noch

Tabular Data-Steuerelement {333C7BC4-460F-11D0-BC04-0080C7055A83} Das TDC (Tabular Data-Steuerelement) ermöglicht die Weiterverarbeitung von Daten, die nur im Textformat vorliegen, beispielsweise durch Darstellung in einer Tabelle oder Sortierung. Weitere Informationen:

http://msdn.microsoft.com/workshop/database/tdc/tabular_data_control_node_entry.asp(http://msdn.microsoft.com/workshop/database/tdc/tabular_data_control_node_entry.asp)

Microsoft Agent Control - Version 2.0 {D45FD31B-5C6E-11D1-9EC1-00C04FD7081F} Microsoft Agent repräsentiert die neue Generation des ursprünglichen Office-Assistenten. Anstatt den Assistenten jedoch innerhalb eines Rahmens darzustellen wird hier lediglich der Charakter bzw. Agent selbst dargestellt und kann auch in Webseiten verwendet werden. Weitere Informationen:

<http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm>(<http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm>)

Microsoft MSChat-Steuerelement-Objekt 2.0 - 2.5 {D6526FE0-E651-11CF-99CB-00C04FD64497}

Dieses Steuerelement wird von Webautoren verwendet, um text- und graphisch basierte Chatgemeinden für Echtzeitkonversationen im Web zu erstellen.

Microsoft ActiveX Upload-Steuerelement, Version 1.5 {886e7bf0-c867-11cf-b1ae-00aa00a3f2c3} Dieses Steuerelement kann auf vielerlei Art genutzt werden, um auf einfache Weise Webinhalte via Drag and Drop zu veröffentlichen. Weitere Informationen: • 230298 (<http://support.microsoft.com/kb/230298/DE/>) - Posting Acceptor Release Notes

• http://msdn.microsoft.com/workshop/management/tools/reference/file_upload_control.asp
(http://msdn.microsoft.com/workshop/management/tools/reference/file_upload_control.asp)



verboten sind

Datenbindung RDS {BD96C556-65A3-11D0-983A-00C04FC29E36} {BD96C556-65A3-11D0-983A-00C04FC29E33} Die RDS (Remote Data Service) Steuerelemente ermöglichen dem Browser, client-basierte SQL Abfragen an einen Webserver zu stellen. Inzwischen wurde RDS jedoch durch neuere Standards wie SOAP abgelöst, von einer weiteren Verwendung von RDS wird daher abgeraten. Weitere Informationen:• 184375 (<http://support.microsoft.com/kb/184375/DE/>) - Sicherheitsaspekte bei RDS 1.5, IIS 3.0 oder 4.0 und ODBC

<http://msdn.microsoft.com/library/en-us/iissdk/iis/remotedatabindingwithremotedataservice.asp>
(<http://msdn.microsoft.com/library/en-us/iissdk/iis/remotedatabindingwithremotedataservice.asp>)

http://msdn.microsoft.com/library/en-us/dnmdac/html/data_mdacroadmap.asp
(http://msdn.microsoft.com/library/en-us/dnmdac/html/data_mdacroadmap.asp)

XMLDSO, XMLDocument, DOMDocument, und XMLIslandPeer {550dda30-0541-11d2-9ca9-0060b0ec3d39} {CFC399AF-D876-11d0-9C10-00C04FC99C8E} {e54941b2-7756-11d1-bc2a-00c04fb925f3} {7108ECB4-AFDC-11D1-ADC1-00805FC752D8} XMLDSO, XMLDocument, DOMDocument, und XMLIslandPeer ermöglichen die Verarbeitung von XML Daten, etwa die Bindung von HTML Elementen an einen XML Datensatz, oder das Einlesen, Manipulieren, und Zurückschreiben von XML Daten.

Die Steuerelemente DOMDocument und XMLIslandPeer bzw. die dazugehörigen ClassIDs sind nicht mehr aktuell, so dass von einer generellen Freigabe dieser Steuerelementgruppe abgeraten wird. Weitere Informationen:• http://msdn.microsoft.com/library/en-us/xmlsdk/htm/xml_concepts2_7ook.asp(http://msdn.microsoft.com/library/en-us/xmlsdk/htm/xml_concepts2_7ook.asp)

Internet Explorer

Active Setup / IE Active Setup-Steuerelement {F72A7B0E-0DD8-11D1-BD6E-00AA00B92AF1} Dieses Steuerelement enthält die in Microsoft Security Bulletin MS99-037 beschriebene Sicherheitsanfälligkeit. Um eine weitere Ausführung zu verhindern wurde im Rahmen dieses Security Bulletins ein Kill-Bit gesetzt, so dass selbst bei einer Freigabe dieses Controls eine Ausführung blockiert wird. Weitere Informationen:• <http://www.microsoft.com/technet/security/bulletin/ms99-037.msp>(<http://www.microsoft.com/technet/security/bulletin/ms99-037.msp>)

<http://www.microsoft.com/technet/security/bulletin/fq99-037.msp>
(<http://www.microsoft.com/technet/security/bulletin/fq99-037.msp>)

240797 (<http://support.microsoft.com/kb/240797/DE/>) - So verhindern Sie die Ausführung von ActiveX-Steuerelementen in Internet Explorer

Media Player / Active Movie Runtime {A4001DE0-7075-11d0-89AB-00A0C9054129} Die Funktionalität dieses Steuerelements wird nun durch das Windows Media Player ActiveX Steuerelement abgedeckt. Das Active Movie Runtime Steuerelement wird daher nicht mehr unterstützt, von einer Freigabe wird abgeraten.

Media Player / ActiveMovie-Steuerelement {05589FA1-C356-11CE-BF01-00AA0055595A} Die Funktionalität dieses Steuerelements wird nun durch das Windows Media Player ActiveX Steuerelement abgedeckt. Das Active Movie Steuerelement wird daher nicht mehr unterstützt, von einer Freigabe wird abgeraten.

Media Player / Microsoft NetShow Player {2179C5D3-EBFF-11CF-B6FD-00AA00B4E220} Die Funktionalität dieses Steuerelements wird nun durch das Windows Media Player ActiveX Steuerelement abgedeckt. Das NetShow Player Steuerelement wird daher nicht mehr unterstützt, von einer Freigabe wird abgeraten.

Media Player / Windows Media Player {22D6F312-B0F6-11D0-94AB-0080C74C7E95} Dies ist das Steuerelement für Windows Media Player version 6.4 und war Installationsbestandteil bis einschließlich Windows Media Player Version 8. Ab Windows Media Player 9 wurde diese ClassID durch die neue ClassID {6BF52A52-394A-11D3-B153-00C04F79FAA6} abgelöst, deren Verwendung stattdessen empfohlen wird. Ab



Windows Media Player Version 9 wird ferner die alte ClassID anhand eines Wrappers automatisch auf die neue ClassID umgeleitet. Die ClassID für Windows Media Player Version 9 ist jedoch nicht in der Liste der vom Administrator genehmigten Steuerelemente enthalten, und muss bei Bedarf manuell hinzugefügt werden.

Animierte Schaltflächen {0482B100-739C-11CF-A3A9-00A0C9034920} Dieses Steuerelement erlaubte in frühen Versionen des Internet Explorer die Verwendung animierter Schaltflächen auf Webseiten. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von der Freigabe des Steuerelements wird daher abgeraten.

IE Label-Steuerelement

{99B42120-6EC7-11CF-A6C7-00AA00A47DD2} Dieses Steuerelement ist nicht mehr aktuell und seit Internet Explorer Version 5 auch kein Bestandteil der Installation mehr. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: • 190045 (<http://support.microsoft.com/kb/190045/DE/>) - INFO: ActiveX Controls That Are Removed from Internet Explorer 5

IE Menu-Steuerelement {74701400-9DD9-11CF-A662-00AA00C066D2} Dieses Steuerelement ermöglicht die Handhabung von Menüstrukturen in Webseiten, wird jedoch nicht mehr unterstützt und dürfte nur noch selten Verwendung finden. Von einer Freigabe des Steuerelements wird daher abgeraten.

IE Preloader-Steuerelement {16E349E0-702C-11CF-A3A9-00A0C9034920} Dieses Steuerelement ermöglichte das Vorladen von Webseiten, ist jedoch inzwischen nicht mehr aktuell, wird nicht mehr unterstützt und dürfte nicht mehr im Einsatz sein. Aufgrund einer potentiellen Sicherheitsanfälligkeit in diesem Steuerelement wird von einer Freigabe abgeraten. Weitere Informationen: • 231452 (<http://support.microsoft.com/kb/231452/DE/>) - Update Available for "Legacy ActiveX Control" Issue

IE Timer-Steuerelement {59CCB4A0-727D-11CF-AC36-00AA00A47DD2} Dieses Steuerelement ist nicht mehr aktuell und seit Internet Explorer Version 5 kein Bestandteil der Installation mehr. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: • 190045 (<http://support.microsoft.com/kb/190045/DE/>) - INFO: ActiveX Controls That Are Removed from Internet Explorer 5

MCSiMenü {275E2FE0-7486-11D0-89D6-00A0C90C9B67} Dieses Steuerelement dient der Anpassung von Pop-upmenüs, ist jedoch nicht mehr aktuell und wurde nach Windows 98 nicht mehr ausgeliefert. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten.

Pop-upmenüobjekt {7823A620-9DD9-11CF-A662-00AA00C066D2} Dieses Steuerelement ist nicht mehr aktuell und seit Internet Explorer Version 5 kein Bestandteil der Installation mehr. Das Steuerelement wird nicht mehr unterstützt und dürfte nur noch vereinzelt im Einsatz sein. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: • 190045 (<http://support.microsoft.com/kb/190045/DE/>) - INFO: ActiveX Controls That Are Removed from Internet Explorer 5

Microsoft Agent Control - Version 1.5 {F5BE8BD2-7DE6-11D0-91FE-00C04FD701A5} Microsoft Agent repräsentiert die neue Generation des ursprünglichen Office-Assistenten. Anstatt den Assistenten jedoch innerhalb eines Rahmens darzustellen wird hier lediglich der Charakter bzw. Agent selbst dargestellt und kann auch in Webseiten verwendet werden. Diese Version des Steuerelements ist jedoch nicht mehr aktuell und wird nicht mehr unterstützt. Von einer Freigabe des Steuerelements wird daher abgeraten. Weitere Informationen: •



<http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm>
 (<http://msdn.microsoft.com/library/partbook/egvb6/introducingmicrosoftagent.htm>)

5.2.3.1. **Dictionary Objekt**

Objekt zur Datenspeicherung in feldähnlicher Form (Dictionary)

Aufbau der Date:

Schlüsselfeld String
 Index im Dictionary
 eindeutig im gesamten Dictionary:

Es muss durch den Programmierer die Eindeutigkeit per Methode .Exists() geprüft werden, da Fehlertoleranz im Dictionary-Objekt **nicht** vorgesehen ist. Bei Zugriff auf einen nicht vorhandenen Schlüssel bzw. bei Einfügung eines bereits im Dictionary vorhandenen Schlüssels erfolgt umgehend die Erzeugung eines Laufzeitfehlers.

Datenfeld String

Syntax:

```
[ var Zeiger = ] new ActiveXObject("Scripting.Dictionary")
```

Beispiel:

```
var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");
```

```
function SchlüsselVorhanden(DatenSchlüssel)
{return DatenSpeicher.Exists(DatenSchlüssel);}
```

```
// Daten-Elemente erzeugen
var DatenSchlüssel = "a";
var Date            = "test"
```

```
// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchlüsselVorhanden (DatenSchlüssel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchlüssel, Date);

    // und Meldung ob Date vorhanden ist
    alert(SchlüsselVorhanden(DatenSchlüssel));
}
```

Eigenschaften:

.Count Anzahl der Elemente im Dictionary
 Integer, ab 1

Methoden:

.Add() Date zum Dictionary Objekt hinzufügen

Beispiel:

```
var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");
```

```
function SchlüsselVorhanden(DatenSchlüssel)
{return DatenSpeicher.Exists(DatenSchlüssel);}
```

```
// Daten-Elemente erzeugen
var DatenSchlüssel = "a";
var Date            = "test"
```

```
// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchlüsselVorhanden (DatenSchlüssel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchlüssel, Date);

    // und Meldung ob Date vorhanden ist
    alert(SchlüsselVorhanden(DatenSchlüssel));
}
```

.Exists() prüfen auf Vorhandensein eines Datenschlüssels im Dictionary

Beispiel:

```
var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");
```

```
function SchlüsselVorhanden(DatenSchlüssel)
{return DatenSpeicher.Exists(DatenSchlüssel);}
```

```
// Daten-Elemente erzeugen
var DatenSchlüssel = "a";
var Date            = "test"
```

```
// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchlüsselVorhanden (DatenSchlüssel)))
```



```

{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchluessel, Date);

    // und Meldung ob Date vorhanden ist
    alert(SchluesselVorhanden(DatenSchluessel));
}

```

.Item() Inhalt des Datenfeldes anhand Schlüssel liefern

Beispiel:

```

var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchluesselVorhanden(DatenSchluessel)
{return DatenSpeicher.Exists(DatenSchluessel);}

// Daten-Elemente erzeugen
var DatenSchluessel = "a";
var Date = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchluesselVorhanden (DatenSchluessel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchluessel, Date);

    // und Date anzeigen
    alert(DatenSpeicher.Item(DatenSchluessel));
}

```

.Items() Inhalte aller Datenfelder des Dictionary-Objektes als Referenz liefern, die als Zeiger für den
Konstruktor new VBArray() (VisualBasic-Anweisung) dient
Datenfelder-Reihenfolge laut Folge im Dictionary

Beispiel:

```

var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchluesselVorhanden(DatenSchluessel)
{return DatenSpeicher.Exists(DatenSchluessel);}

// Daten-Elemente erzeugen
var DatenSchluessel = "a";
var Date = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchluesselVorhanden (DatenSchluessel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchluessel, Date);

    // und Daten-Referenz bilden
    var DatenReferenz = DatenSpeicher.Items();

    // und Feld bilden
    // VisualBasic-Feld erzeugen
    var DatenOhneSchluessel_Feld = new VBArray(DatenReferenz);
    // und nach JScript-Feld konvertieren
    DatenOhneSchluessel_Feld = DatenOhneSchluessel_Feld.toArray();

    // und Daten anzeigen
    var DatenOhneSchluessel_FeldLaenge = DatenOhneSchluessel_Feld.length

    if (DatenOhneSchluessel_FeldLaenge > 0)
    {
        for (var i = 0 ; i < DatenOhneSchluessel_FeldLaenge; i++)
        {alert("Date " + i + " = " + DatenOhneSchluessel_Feld [i]);}
    }
    else
    {alert("Dictionary ist leer!");}
}

```

.Key() Datenschlüssel im Dictionary ändern

Beispiel:

```

var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchluesselVorhanden(DatenSchluessel)

```



```

{return DatenSpeicher.Exists(DatenSchluessel);}

// Daten-Elemente erzeugen
var DatenSchluessel = "a";
var Date = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchluesselVorhanden (DatenSchluessel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchluessel, Date);

    // dann Schlüssel ändern
    DatenSpeicher.Key(DatenSchluessel) = "b";

    // und Meldung ob Date vorhanden ist
    alert(SchluesselVorhanden("b"));
}

```

.Keys() Inhalt aller Schlüsselfelder des Dictionary-Objektes als Referenz liefern, die als Zeiger für den Konstruktor `new VBArray ()` (VisualBasic-Anweisung) dient
Schlüsselfelder-Reihenfolge laut Folge im Dictionary

Beispiel:

```

var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchluesselVorhanden(DatenSchluessel)
{return DatenSpeicher.Exists(DatenSchluessel);}

// Daten-Elemente erzeugen
var DatenSchluessel = "a";
var Date = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchluesselVorhanden (DatenSchluessel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchluessel, Date);

    // und Daten-Referenz bilden
    var SchluesselReferenz = DatenSpeicher.Keys();

    // und Feld bilden
    // VisualBasic-Feld erzeugen
    var SchluesselOhneDaten_Feld = new VBArray(SchluesselReferenz);
    // und nach JScript-Feld konvertieren
    SchluesselOhneDaten_Feld = SchluesselOhneDaten_Feld.toArray();

    // und Daten anzeigen
    var SchluesselOhneDaten_FeldLaenge = SchluesselOhneDaten_Feld.length

    if (SchluesselOhneDaten_FeldLaenge > 0)
    {
        for (var i = 0 ; i < SchluesselOhneDaten_FeldLaenge; i++)
        {alert("Schluessel " + i + " = " + SchluesselOhneDaten_Feld [i]);}
    }
    else
    {alert("Dictionary ist leer!");}
}

```

.Remove() genau eine Date aus dem Dictionary Objekt entfernen (Schlüssel- und Datenfeld entfernen)

Beispiel:

```

var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchluesselVorhanden(DatenSchluessel)
{return DatenSpeicher.Exists(DatenSchluessel);}

// Daten-Elemente erzeugen
var DatenSchluessel = "a";
var Date = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchluesselVorhanden(DatenSchluessel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen

```



```

    DatenSpeicher.add (DatenSchluessel, Date);

    // und Date anzeigen
    alert(DatenSpeicher.Item(DatenSchluessel));

    // und löschen
    DatenSpeicher.remove(DatenSchluessel);

    // und Meldung
    alert(SchluesselVorhanden(DatenSchluessel));
}

```

.RemoveAll() alle Daten aus dem Dictionary entfernen (alle Schlüssel- und Datenfelder), so dass das Dictionary Objekt leer aber weiterhin instanziiert ist

Beispiel:

```

var DatenSpeicher = new ActiveXObject("Scripting.Dictionary");

function SchluesselVorhanden(DatenSchluessel)
{return DatenSpeicher.Exists(DatenSchluessel);}

// Daten-Elemente erzeugen
var DatenSchluessel = "a";
var Date = "test"

// prüfen ob Schlüssel noch nicht vorhanden ist
if ( ! (SchluesselVorhanden(DatenSchluessel)))
{
    // Schlüssel nicht vorhanden, also Date hinzufügen
    DatenSpeicher.add (DatenSchluessel, Date);

    // und Date anzeigen
    alert(DatenSpeicher.Item(DatenSchluessel));

    // und löschen
    DatenSpeicher.removeAll();

    // und Meldung
    alert(SchluesselVorhanden(DatenSchluessel));
}

```

5.2.3.2. **FileSystemObject Objekt**

Objekt des abgespeckten Dateisystems im Betriebssystem.

Der Zugriff auf das Dateisystem über dieses Objekt hängt teilweise vom Betriebssystem bzw. von Einstellungen zur Ordern und Dateien ab.

Das FileSystemObject Objekt besitzt weitere Objekte sowie interne Collectionen. Diese müssen per Script durch Ableitung von FileSystemObject instanziiert werden und sind erst danach verfügbar. Abgeleitete Objekte besitzen eigene Eigenschaften und Methoden und bekommen **nichts** vom FileSystemObject vererbt. Damit ist sichergestellt, dass der Zugriff auf das Dateisystem nur im jeweiligen Context des **instanziierten** Objektes stattfindet. Mit anderen Worten: Es existiert kein pauschaler kompletter Zugriff auf das Dateisystem außerhalb der Eigenschaften und Methoden von FileSystemObject, wenn kein weiteres Objekt abgeleitet wurde.. Ein Programmierer, der ableitet, **will** also den Zugriff auf das Dateisystem **bewusst** erweitern.

Der Zugriff auf interne Collectionen ist z.T. nur über das JScript-Objekt Enumerator möglich.

Warnungen:

Die Nutzung dieses Objektes berührt unmittelbar die Privatsphäre des Users. Die Nutzung des Objektes kann rechtlich relevant werden, so dass für den User Transparenz bezüglich der Objektverwendung vorliegen muss.

Es kann sein, dass ein aktiver Virens Scanner, der Scripte in Webseiten untersucht, Virenmeldungen erzeugt.

Ein User, der das Active-X-Control mit oder ohne sein Wissen zulässt, geht definitiv das Risiko einer missbräuchlichen Nutzung des Objektes ein, wenn kein Virens Scanner die Scripte vor deren Aktivierung prüft und nach User-Entscheidung oder generell blockiert und somit das Privateigentum des Users schützt.

Sämtliche Beispiele sind mit Vorsicht zu genießen und möglichst nicht zu testen, es sei denn, der Programmierer weiss genau, was er bewirken will ! In einigen Beispielen wird das Laufwerk C: derart manipuliert, dass bei fehlerhafter Programmierung ein totaler Datenverlust eintreten kann. Das Testsystem sollte also eine Kopie des ursprünglichen Systems sein, das alternativ vorher mit einem Backup-Programm komplett gesichert werden sollte ! Desweiteren müssen eine aktive Firewall und deren Virens Scanner deaktiviert sein.

Ziel der Beispiele ist es nur, die Gefährlichkeit der Verwendung des Objektes FileSystemObject zu zeigen.

Syntax:

```
[ var Zeiger = ] new ActiveXObject("Scripting.FileSystemObject");
```



Beispiel Laufwerke auf dem PC des Users ermitteln (ein harmloses Beispiel):

```

var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Laufwerke = new Enumerator(DateiSystem.Drives);
var Kette = "";
var Laufwerk;
var LaufwerkBuchstabe;
var LaufwerkName; // Netzname oder Volume-Bezeichner

for ( ; !DateiSystem_Laufwerke.atEnd();DateiSystem_Laufwerke.moveNext()
{
    // Laufwerk ermitteln
    Laufwerk = DateiSystem_Laufwerke.item();

    // Laufwerksbuchstabe ermitteln
    LaufwerkBuchstabe = Laufwerk.DriveLetter;
    Kette = Kette + LaufwerkBuchstabe + " - ";

    // Art des Laufwerkes ermitteln

    if (Laufwerk.DriveType == 3)
    {
        // ist Netzlaufwerk

        // öffentlicher Netz-Name des Laufwerkes
        LaufwerkName = Laufwerk.ShareName;
    }
    else
    {
        // nicht Netzwerk-Laufwerk

        // prüfen ob Laufwerk bereit ist
        if (Laufwerk.IsReady)
        {
            // ist bereit, dann Volume-Bezeichner ermittelbar
            LaufwerkName = Laufwerk.VolumeName;
        }
        else
        { LaufwerkName = "[Drive not ready]";}
    }

    Kette += LaufwerkName + "\n";
}
alert (Kette);

```

Eigenschaften:

keine

Methoden:

Hinweise: Alle Angaben zu einer Datei mit Pfad dürfen nicht mit "\" bzw. "\\\" enden, da sonst ein Ordnername erkannt wird.
 Methoden zu Ordnern müssen bei Pfadangaben kein "\" bzw. "\\\" am Ende kodiert haben
 Verwendung von Wildcard "*" ist nur z.T. möglich. Bei Ordnern darf "*" nur am Ende einer Pfadanangabe stehen.
 Pfade sind fast immer relativ und absolut kodierbar (Relativ z.B. per "\\.").

.BuildPath() PATH-Variable im Dateisystem erweitern durch Anhängen
 keine Erweiterung des Dateisystems per Ordner (dafür gibt es andere Methoden)

.CopyFile() Datei(en) kopieren
 Wenn Copy abbricht, so bleiben bisher erfolgte Kopieraktionen leider erhalten

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.CopyFile ("c:\\eigene dateien\\*.doc", "c:\\recycled\\")

```

.CopyFolder() Ordner kopieren
 Wenn Copy abbricht, so bleiben bisher erfolgte Kopieraktionen leider erhalten

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.CopyFolder("c:\\*", "d:\\recycled\\")

```

.CreateFolder() Ordner erzeugen, der nicht bereits existieren darf (sonst wird Fehler erzeugt)

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.CreateFolder("c:\\test")

```

.CreateTextFile() Textdatei anlegen und zum Schreiben als Textstream öffnen



Schreiben und Schliessen der Datei per Objekt FileSystemObject.TextStream

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.CreateTextFile("c:\\test.txt", true);
DateiOffen.WriteLine("Test");
DateiOffen.Close();
```

.DeleteFile() vorhandene Datei(en) löschen
Wenn Delete abbricht, so bleiben bisher erfolgte Löschkaktionen leider erhalten

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.DeleteFile("c:\\*.txt", true);
```

.DeleteFolder() vorhandene Ordner löschen, wobei es egal ist, ob Daten und/oder Unterordner im jeweiligen Ordner liegen oder nicht
Wenn Delete abbricht, so bleiben bisher erfolgte Löschkaktionen leider erhalten

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.DeleteFolder("c:\\*", true);
```

.DriveExists() prüfen auf Laufwerk im Dateisystem
Laufwerk muss nicht bereits ein (z.B. Diskettenlaufwerk muss kein Medium enthalten)

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.DriveExists("A"));
```

.FileExists() prüfen auf Datei

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.FileExists("c:\\test.txt"));
```

.FolderExists() prüfen auf Ordner

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.FolderExists("c:\\test\\"));
```

.GetAbsolutePathName() absoluten Pfad ab Root mit Laufwerk ermitteln
Angabe von Wildcard "*" möglich wird **leider nicht** aufgelöst
Angabe von "\\." möglich falls Anzahl von "\\." eine Logik hinter die Root erzeugen würde, so wird kein Fehler erzeugt, sondern korrekt ab Root aufgelöst
Bsp.: c:\test\ existiert
Auflösung von c:\\.\\..\\test bringt keinen Fehler

Beispiel 1:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetAbsolutePathName("..\test\\"));
```

Beispiel 2:

Es gilt: aktuelle Position im Dateisystem ist c:\test\texte
Ordner texte hat folgenden Unterordner jahr2002
Ordner jahr2002 hat folgenden Unterordner mai2002

Kette1	Kette2	Bedeutung
"c:"	"c:\test\texte"	aktuelle Position
"c:.."	"c:\test"	Ordner oberhalb der aktuelle Position
"c:\\"	"c:\"	Root und nicht aktuelle Position
"c:.*\mai2002"	"c:\test\texte\.*\mai2002"	Pfad von mai2002
"jahr2002"	"c:\test\texte\jahr2002"	keine Auflösung von "*" Pfad von jahr2002
"c:\\.\\..\\test"	"c:\test"	kein Fehler

.GetBaseName() absoluten Pfad einer Datei im Pfad liefern
Datei muss nicht existieren

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetBaseName("..\test\text.txt"));
```

.GetDrive() Objekt FileSystemObject.Drive anhand des Laufwerksbuchstaben eines existierenden Laufwerkes erzeugen (auch bei Netzlaufwerk) sonst wird Fehler erzeugt

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk = DateiSystem.GetDrive("C");
var Laufwerk_VolumeName = Laufwerk.VolumeName;
```



```
var Laufwerk_TotalerPlatz = Laufwerk.TotalSize;
alert(Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_TotalerPlatz);
```

`.GetDriveName()` Laufwerksbuchstabe aus Pfad ermitteln in der Form "x:" mit x für Laufwerk
 Pfad muss nicht existieren
 muss Laufwerksbuchstaben enthalten
 wird nicht auf Syntax geprüft

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetDriveName("c:\\.\\.*\\test\\***"));
```

`.GetExtensionName()` Suffix einer Datei liefern ohne Trenner "."
 Datei muss nicht existieren

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetExtensionName("../test\\text.txt"));
```

`.GetFile()` Objekt FileSystemObject.File anhand Dateinamen einer existierenden Datei erzeugen
 (sonst wird Fehler erzeugt)

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFile("../test\\text.txt");
alert(Datei);
```

`.GetFileName()` Name einer Datei mit Suffix und mit Trenner "."
 Datei muss nicht existieren

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetFileName("../test\\text.txt"));
```

`.GetFolder()` Objekt FileSystemObject.Folder anhand Ordnernamen eines existierenden Ordners erzeugen
 sonst wird Fehler erzeugt

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner = DateiSystem.GetFolder("../test\\");
alert(Ordner);
```

`.GetParentFolderName()` Elternordner einer Datei liefern
 Datei muss nicht existieren

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
alert(DateiSystem.GetParentFolderName("../test\\text.txt"));
```

`.GetSpecialFolder()` Objekt FileSystemObject.Folder von Ordnern des Betriebssytemes erzeugen

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner = DateiSystem.GetSpecialFolder(0);
var DateiOffen = Ordner.CreateTextFile("test.txt");
DateiOffen.writeline("Test");
DateiOffen.close();
```

`.GetTempName()` Bezeichner anhand Zufallszahl erzeugen
 Bezeichner verwendbar als Datei- oder Ordner-Bezeichner

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner = DateiSystem.GetSpecialFolder(2);
var DateiName = DateiSystem.GetTempName();
var DateiOffen = Ordner.CreateTextFile(DateiName);
DateiOffen.writeline("Test");
DateiOffen.close();
```

`.MoveFile()` Datei(en) verschieben
 Wenn Move abbricht, so bleiben bisher erfolgte Verschiebungen leider erhalten

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.MoveFile("c:\\eigene dateien\\*.doc", "c:\\recycled\\")
```

`.MoveFolder()` Ordner verschieben
 Wenn Move abbricht, so bleiben bisher erfolgte Verschiebungen leider erhalten

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
DateiSystem.MoveFolder("c:\\eigene dateien\\", "c:\\recycled\\")
```

`.OpenTextFile()` Datei als Text öffnen zum Lesen, Schreiben oder Append (Schreiben durch Anhängen)



Schreiben und Schliessen der Datei per Objekt FileSystemObject.TextStream

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0)
DateiOffen.WriteLine("Test");
DateiOffen.Close();
```

5.2.3.2.1. FileSystemObject.Drive Objekt

Objekt für Zugriff auf Laufwerke (lokale oder im Netz)

Erzeugung:

```
[ var Zeiger = ] new ActiveXObject("Scripting.FileSystemObject")
```

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_FreierPlatz = Laufwerk.FreeSpace;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_FreierPlatz);
```

Eigenschaften:

.AvailableSpace Freien Speicher in Bytes auf Laufwerk ermitteln

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_FreierPlatz = Laufwerk.AvailableSpace;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_FreierPlatz);
```

.DriveLetter Laufwerksbuchstaben liefern (ohne : und \\ etc.)

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Buchstabe = Laufwerk.DriveLetter;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Buchstabe);
```

.DriveType Laufwerkstyp liefern

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Typ = Laufwerk.DriveType;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Typ);
```

.FileSystem Typ des Dateisystems auf dem Laufwerk liefern

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Dateisystem = Laufwerk.FileSystem;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Dateisystem);
```

.FreeSpace Freien Speicher in Bytes auf Laufwerk ermitteln

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_FreierPlatz = Laufwerk.FreeSpace;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_FreierPlatz);
```

.IsReady Bereitschaft des Laufwerkes für Zugriffe z.B. ob Medium im Laufwerk liegt

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Bereitschaft = Laufwerk.IsReady;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Bereitschaft);
```

.Path Pfad eines Laufwerkes ermitteln in voller Länge (nicht 8.3 Pfad)



Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Pfad = Laufwerk.Path;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Pfad);
```

.RootFolder Root des Laufwerkes liefern, also mit ":"

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Root = Laufwerk.RootFolder;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_Root);
```

.SerialNumber Seriennummer des Laufwerkes liefern

Achtung: Die Seriennummer lässt eine eindeutige Identifizierung der Festplattenhardware zu vor allem dann, wenn zusätzlich Userdaten zum PC bekannt sind.

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_Seriennummer = Laufwerk.SerialNumber;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " +
      Laufwerk_Seriennummer);
```

.ShareName öffentlicher Netzwerkname des Laufwerkes liefern

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_ShareName = Laufwerk.ShareName;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_ShareName);
```

.TotalSize Gesamten Speicher in Bytes auf Laufwerk ermitteln

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_TotalePlatz = Laufwerk.TotalSize;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_TotalePlatz);
```

.VolumeName Volumenbezeichner des Laufwerkes

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Laufwerk_Bezeichnung = DateiSystem.GetDriveName("C:");
var Laufwerk = DateiSystem.GetDrive(Laufwerk_Bezeichnung);
var Laufwerk_VolumeName = Laufwerk.VolumeName;
var Laufwerk_TotalePlatz = Laufwerk.TotalSize;
alert(Laufwerk_Bezeichnung + " " + Laufwerk + " " + Laufwerk_VolumeName + " " + Laufwerk_TotalePlatz);
```

Methoden:

keine

5.2.3.2.2. FileSystemObject.Drives Collection

interne Collection aller Laufwerke, egal ob sie bereit sind oder nicht (z.B. Diskettenlaufwerk muss kein Medium besitzen) alle Elemente in der Collection sind nur lesbar

Erzeugung:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Laufwerke = DateiSystem.Drives; // interne Collection
var DateiSystem_DrivesCollection = new Enumerator(DateiSystem_Laufwerke);
// durch Programmierer handhabbare Collection
```

Zugriff:

z.T. nur in Verbindung mit JScript-Objekt Enumerator

Beispiel Laufwerke auf dem PC des Users ermitteln (ein harmloses Beispiel):

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Laufwerke = new Enumerator(DateiSystem.Drives);
var Kette = "";
var Laufwerk;
```



```

var LaufwerkBuchstabe;
var LaufwerkName; // Netzname oder Volume-Bezeichner

for ( ; ! DateiSystem_Laufwerke.atEnd();DateiSystem_Laufwerke.moveNext()
{
    // Laufwerk ermitteln
    Laufwerk = DateiSystem_Laufwerke.item();

    // Laufwerksbuchstabe ermitteln
    LaufwerkBuchstabe = Laufwerk.DriveLetter;
    Kette = Kette + LaufwerkBuchstabe + " - ";

    // Art des Laufwerkes ermitteln

    if (Laufwerk.DriveType == 3)
    {
        // ist Netzlaufwerk

        // öffentlicher Netz-Name des Laufwerkes
        LaufwerkName = Laufwerk.ShareName;
    }
    else
    {
        // nicht Netzwerk-Laufwerk

        // prüfen ob Laufwerk bereit ist
        if (Laufwerk.IsReady)
        {
            // ist bereit, dann Volume-Bezeichner ermittelbar
            LaufwerkName = Laufwerk.VolumeName;
        }
        else
        { LaufwerkName = "[Drive not ready]";}
    }

    Kette += LaufwerkName + "\n";
}
alert (Kette);

```

Eigenschaften:

.Count Anzahl der Elemente in der Collection FileSystemObject.Drives
Integer, ab 1

Methoden:

.Item() Eintrag in der Collection FileSystemObject.Drives liefern
nur in Verbindung mit JScript-Objekt Enumerator

Beispiel Laufwerke auf dem PC des Users ermitteln:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Laufwerke = new Enumerator(DateiSystem.Drives);
var Kette = "";
var Laufwerk;
var LaufwerkBuchstabe;
var LaufwerkName; // Netzname oder Volume-Bezeichner

for ( ; ! DateiSystem_Laufwerke.atEnd();DateiSystem_Laufwerke.moveNext()
{
    // Laufwerk ermitteln
    Laufwerk = DateiSystem_Laufwerke.item();

    // Laufwerksbuchstabe ermitteln
    LaufwerkBuchstabe = Laufwerk.DriveLetter;
    Kette = Kette + LaufwerkBuchstabe + " - ";

    // Art des Laufwerkes ermitteln

    if (Laufwerk.DriveType == 3)
    {
        // ist Netzlaufwerk

        // öffentlicher Netz-Name des Laufwerkes
        LaufwerkName = Laufwerk.ShareName;
    }
    else
    {
        // nicht Netzwerk-Laufwerk

```



```

        // prüfen ob Laufwerk bereit ist
        if (Laufwerk.IsReady)
        {
            // ist bereit, dann Volume-Bezeichner ermittelbar
            LaufwerkName = Laufwerk.VolumeName;
        }
        else
        { LaufwerkName = "[Drive not ready]"; }
    }

    Kette += LaufwerkName + "\n";
}
alert (Kette);

```

5.2.3.2.3. FileSystemObject.File Objekt

Objekt für Zugriff auf eine **vorhandene** Datei (sonst Fehler erzeugt)

Erzeugung:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFile(Kette);

```

Kette String
Dateiname mit Pfad z.B. "c:\\test.txt"

Beispiel:

```

var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFile(DateiNameMitPfad);
alert(Datei.DateCreated);

```

Eigenschaften:

.Attributes Attribute der Datei

Beispiel:

```

var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFile(DateiNameMitPfad);
var Datei_Attribute = Datei.attributes;

```

```

// auf gesetztes Archivattribut prüfen
if (Datei_Attribute && 32)
{
    // löschen
    Datei_Attribute -= 32;
}
else
{
    // setzen
    Datei_Attribute += 32;
}

```

.DateCreated Datum und Zeit der Dateierstellung liefern

Beispiel:

```

var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFile(DateiNameMitPfad);
alert(Datei.DateCreated);

```

.DateLastAccessed Datum und Zeit des letzten Dateizugriffes liefern

Beispiel:

```

var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFile(DateiNameMitPfad);
alert(Datei.DateLastAccessed);

```

.DateLastModified Datum und Zeit der letzten Dateiänderung liefern

Beispiel:

```

var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFile(DateiNameMitPfad);
alert(Datei.DateLastModified);

```

.Drive Buchstaben des Laufwerkes liefern, auf dem die Datei liegt

Beispiel:

```

var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFile(DateiNameMitPfad);

```



alert(Datei.Drive);

.Name Dateibezeichner in voller Länge (nicht 8.3 Bezeichner)

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFiles(DateiNameMitPfad);
alert(Datei.Name);
```

.ParentFolder Bezeichner des Ordners liefern, in dem die Datei liegt

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFiles(DateiNameMitPfad);
alert(Datei.ParentFolder);
```

.Path Pfad der Datei liefern in voller Länge (nicht 8.3 Pfad)

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFiles(DateiNameMitPfad);
alert(Datei.Path);
```

.ShortName Dateibezeichner als 8.3 Bezeichner liefern

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFiles(DateiNameMitPfad);
alert(Datei.ShortName);
```

.ShortPath Pfad der Datei als 8.3 Pfad liefern

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFiles(DateiNameMitPfad);
alert(Datei.ShortPath);
```

.Size Dateigrösse in Bytes liefern

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFiles(DateiNameMitPfad);
alert(Datei.Size);
```

.Type Beschreibung zum Suffix des Dateibezeichners liefern laut Registry

z.B. Suffix ist .TXT Beschreibung ist "Text Document"

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFiles(DateiNameMitPfad);
alert(Datei.Type);
```

Methoden:

.Copy() vorhandene Datei kopieren (sonst Fehler erzeugt)

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFiles(DateiNameMitPfad);
Datei.Copy("c:\\windows\\desktop\\test2.txt");
```

.Delete() vorhandene Datei löschen (sonst Fehler erzeugt)

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFiles(DateiNameMitPfad);
Datei.Delete();
```

.Move() vorhandene Datei verschieben (sonst Fehler erzeugt)

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Datei = DateiSystem.GetFiles(DateiNameMitPfad);
Datei.Move("c:\\windows\\desktop\\");
```

.OpenAsTextStream() vorhandene Datei als Text öffnen zum Lesen und Schreiben oder Append, wenn Dateiattribute es



zulassen (sonst Fehler erzeugt)
Schreiben und Schliessen der Datei per Objekt FileSystemObject.TextStream

Beispiel:

```
var DateiNameMitPfad = "c:\\test.txt";
var DateiSystem      = new ActiveXObject("Scripting.FileSystemObject");
var Datei            = DateiSystem.GetFile(DateiNameMitPfad);
var DateiOffen      = Datei.OpenAsTextStream(2,0);
DateiOffen.Write( "neuer Text" );
DateiOffen.Close();
DateiOffen          = Datei.OpenAsTextStream(1,2);
var Kette           = DateiOffen.ReadLine( );
DateiOffen.Close();
alert(Kette);
```

5.2.3.2.4. FileSystemObject.Folder Objekt

Objekt für Zugriff auf eine **vorhandenen** Ordner (sonst Fehler erzeugt)

Erzeugung:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner      = DateiSystem.GetFolder(Kette);
```

Kette String
Ordner mit Pfad z.B. "c:\\test\\"

```
var SpezialOrdner = DateiSystem.GetSpecialFolder(Wert)
```

Wert Integer
0 Windows-Ordner
1 System-Ordner
2 Temp-Ordner

Eigenschaften:

.Attributes Attribute des Ordner

Beispiel:

```
var OrdnerName = "c:\\windows\\desktop\\";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner      = DateiSystem.GetFolder(OrdnerName);
var Ordner_Attribute = Ordner.attributes;
```

```
// auf gesetztes Archivattribut prüfen
if (Ordner_Attribute && 32)
{
    // löschen
    Ordner_Attribute -= 32;
}
else
{
    // setzen
    Ordner_Attribute += 32;
}
```

.DateCreated Datum und Zeit der Ordnererstellung liefern

Beispiel:

```
var OrdnerName = "c:\\windows\\desktop\\";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner      = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.DateCreated);
```

.DateLastAccessed Datum und Zeit des letzten Ordnerzugriffes liefern

Beispiel:

```
var OrdnerName = "c:\\windows\\desktop\\";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner      = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.DateLastAccessed);
```

.DateLastModified Datum und Zeit der letzten Ordneränderung liefern

Beispiel:

```
var OrdnerName = "c:\\windows\\desktop\\";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner      = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.DateLastModified);
```

.Drive Buchstaben des Laufwerkes liefern, auf dem der Ordner liegt

Beispiel:

```
var OrdnerName = "c:\\windows\\desktop\\";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner      = DateiSystem.GetFolder(OrdnerName);
```



```
alert(Ordner.Drive);
```

.Files Zeiger auf die interne Collection FileSystemObject.Folder.Files
Collection kann nur z.T. über das JScript-Objekt Enumerator angesprochen werden

.IsRootFolder prüfen auf Root

Beispiel:

```
var OrdnerName              = "c:\\windows\\desktop\\";
var DateiSystem            = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                  = DateiSystem.GetFolder(OrdnerName);
var Zahler                 = 0;
if (!Ordner.IsRootFolder)
{
    do
    {
        Ordner = Ordner.ParentFolder;
        Zahler++;
    }
    while (!Ordner.IsRootFolder);
}

alert("Ordner liegt " + Zahler + " Ebenen unter der Root");
```

.Name Ordnerbezeichner in voller Länge (nicht 8.3 Bezeichner)

Beispiel:

```
var OrdnerName              = "c:\\windows\\desktop\\";
var DateiSystem            = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                  = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.Name);
```

.ParentFolder Bezeichner des Eltern-Ordners liefern, in dem der Ordner liegt

Beispiel 1:

```
var OrdnerName              = "c:\\windows\\desktop\\";
var DateiSystem            = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                  = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.ParentFolder);
```

Beispiel 2:

```
var OrdnerName              = "c:\\windows\\desktop\\";
var DateiSystem            = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                  = DateiSystem.GetFolder(OrdnerName);
var Zahler                 = 0;
if (!Ordner.IsRootFolder)
{
    do
    {
        Ordner = Ordner.ParentFolder;
        Zahler++;
    }
    while (!Ordner.IsRootFolder);
}

alert("Ordner liegt " + Zahler + " Ebenen unter der Root");
```

.Path Pfad des Ordners liefern in voller Länge (nicht 8.3 Pfad)

Beispiel:

```
var OrdnerName              = "c:\\windows\\desktop\\";
var DateiSystem            = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                  = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.Path);
```

.ShortName Ordnerbezeichner als 8.3 Bezeichner liefern

Beispiel:

```
var OrdnerName              = "c:\\windows\\desktop\\";
var DateiSystem            = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                  = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.ShortName);
```

.ShortPath Pfad des Ordners als 8.3 Pfad liefern

Beispiel:

```
var OrdnerName              = "c:\\windows\\desktop\\";
var DateiSystem            = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                  = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.ShortPath);
```



.Size Ordnergrösse in Bytes liefern

```

Beispiel:
var OrdnerName           = "c:\\windows\\desktop\\";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.Size);

```

.SubFolders Zeiger auf die interne Collection FileSystemObject.Folder.Folders
Collection kann nur z.T. über das JScript-Objekt Enumerator angesprochen werden

.Type Beschreibung zum Suffix des Ordnerbezeichners liefern laut Registry

```

Beispiel:
var OrdnerName           = "c:\\windows\\desktop\\";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                = DateiSystem.GetFolder(OrdnerName);
alert(Ordner.Type);

```

Methoden:

.Copy() vorhandenen Ordner kopieren (sonst Fehler erzeugt)

```

Beispiel:
var OrdnerName           = "c:\\windows\\desktop\\";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                = DateiSystem.GetFolder(OrdnerName);
Ordner.Copy("d:\\recycled\\");

```

.Delete() vorhandenen Ordner löschen (sonst Fehler erzeugt)

```

Beispiel:
var OrdnerName           = "c:\\windows\\desktop\\";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                = DateiSystem.GetFolder(OrdnerName);
Ordner.Delete();

```

.Move() vorhandenen Ordner verschieben (sonst Fehler erzeugt)

```

Beispiel:
var OrdnerName           = "c:\\windows\\desktop\\";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var Ordner                = DateiSystem.GetFolder(OrdnerName);
Ordner.Move("d:\\recycled\\");

```

5.2.3.2.4.1. FileSystemObject.Folder.Files Collection

interne Collection aller Dateien innerhalb eines Ordners (ohne Dateien in Unterordnern)
alle Elemente in der Collection sind nur lesbar

Erzeugung:

```

var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Ordner  = DateiSystem.GetFolder(Kette);
var DateiSystem_Dateien = DateiSystem_Ordner.Files; // interne Collection
var DateiSystem_FilesCollection = new Enumerator(DateiSystem_Dateien);
// vom Programmierer verwaltbare Collection
Kette String
Pfad und Name des Ordners

```

Zugriff:

z.T. nur in Verbindung mit JScript-Objekt Enumerator

Beispiel:

```

var Ordner                = "c:\\windows\\desktop\\";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Ordner  = DateiSystem.GetFolder(Ordner);
var DateiSystem_Dateien = DateiSystem_Ordner.Files;
var DateiSystem_FilesCollection = new Enumerator(DateiSystem_Dateien);
var Kette                = "";
for (; !DateiSystem_FilesCollection.atEnd(); DateiSystem_FilesCollection.moveNext())
{Kette = Kette + DateiSystem_FilesCollection.item() + "\n";}
alert(Kette);

```

Eigenschaften:

.Count Anzahl der Elemente in der Collection FileSystemObject.Folder.Files
Integer, ab 1

Methoden:

.Item() Eintrag in der Collection FileSystemObject.Files liefern
nur in Verbindung mit JScript-Objekt Enumerator

```

Beispiel:
var Ordner                = "c:\\windows\\desktop\\";
var DateiSystem          = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Ordner  = DateiSystem.GetFolder(Ordner);
var DateiSystem_Dateien = DateiSystem_Ordner.Files;
var DateiSystem_FilesCollection = new Enumerator(DateiSystem_Dateien);

```



```

var Kette = "";
for (; ! DateiSystem_FilesCollection.atEnd(); DateiSystem_FilesCollection.moveNext())
{Kette = Kette + DateiSystem_FilesCollection.item() + "\n";}
alert(Kette);

```

5.2.3.2.4.2. **FileSystemObject.Folder.Folders Collection**

interne Collection aller Ordner innerhalb eines **Ordners**, wobei nur die Ordner eine Hierarchiestufe tiefer gemeint sind !
alle Elemente in der Collection sind nur lesbar

Erzeugung:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiSystem_Ordner = DateiSystem.GetFolder(Kette);
var DateiSystem_FoldersCollection = new Enumerator(DateiSystem_Ordner.SubFolders);
// vom Programmierer verwaltbare Collection

```

Kette String
Pfad und Name des Ordners

Zugriff:

z.T. nur in Verbindung mit JScript-Objekt Enumerator

Beispiel:

```

var OrdnerName = "c:\\windows\\desktop\\";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner = DateiSystem.GetFolder(OrdnerName);
var DateiSystem_FoldersCollection = new Enumerator(Ordner.SubFolders);
var Kette = "";
for (; ! DateiSystem_FoldersCollection.atEnd(); DateiSystem_FoldersCollection.moveNext())
{Kette = Kette + DateiSystem_FoldersCollection.item() + "\n";}
alert(Kette);

```

Eigenschaften:

.Count Anzahl der Elemente in der Collection FileSystemObject.Folder.Folders
Integer, ab 1

Methoden:

.Item() Eintrag in der Collection Collection FileSystemObject.Folder.Folders liefern
nur in Verbindung mit JScript-Objekt Enumerator

Beispiel:

```

var OrdnerName = "c:\\windows\\desktop\\";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner = DateiSystem.GetFolder(OrdnerName);
var DateiSystem_FoldersCollection = new Enumerator(Ordner.SubFolders);
var Kette = "";
for (; ! DateiSystem_FoldersCollection.atEnd(); DateiSystem_FoldersCollection.moveNext())
{Kette = Kette + DateiSystem_FoldersCollection.item() + "\n";}
alert(Kette);

```

.Add() Ordner der Collection Collection FileSystemObject.Folder.Folders hinzufügen
Ordner darf in der Collection nicht bereits existieren (sonst Fehler erzeugt)

Beispiel:

```

var OrdnerName = "c:\\windows\\desktop\\";
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var Ordner = DateiSystem.GetFolder(OrdnerName);
var DateiSystem_FoldersCollection = Ordner.SubFolders;
var NeuerOrdner = DateiSystem_FoldersCollection.Add("Neuer Ordner");

```

5.2.3.2.5. **FileSystemObject.TextStream Objekt**

Objekt einer sequentiellen und offenen Text-Datei auf Basis eines Textstreams (Datenfluss aus Textzeichen), der

intern zeilenweise (durch Programmierer nicht veränderbar)
extern zeilen- und/oder zeilenweise (durch Programmierer einstellbar)
verwaltet wird.

Datei ist lesbar **oder** schreibbar (beides gleichzeitig geht leider **nicht**):

satzweise: zeilenweise
es muss Zeichen newline
per .WriteLine() bzw. .WriteBlankLines() geschrieben worden sein
gelesen worden sein

zeilenweise: wenn newline-Zeichen per .WriteBlankLines() geschrieben wurde, so wird Satzendumarke
erzeugt

zeichen- und satzweise: ist möglich, denn ein Zeilenende wird nur erkannt, wenn newline-Zeichen gelesen wurde
Achtung: zeilenweises Lesen per .ReadLine() erwartet entweder newline-Zeichen oder
Dateieinde (Textstream-Ende)

als komplette Datei nur Lesen per .ReadAll()
(alle Daten auf einen Schlag lesen, egal ob die Datei zeichen- oder zeilenweise
erstellt wurde)

interner Satzzeiger verfügbar:

bei zeichweisem Lesen bzw. Schreiben ist es ein **Zeichenzeiger**
bei zeilenweisem Lesen bzw. Schreiben ist es ein **Zeilenzeiger**
wird nicht verwendet, wenn Datei auf einen Schlag komplett gelesen wird per .ReadAll()



numerischer Zeichenzeiger verfügbar (nur lesen): nur bei zeichenweiser Dateiverarbeitung (in der Datei darf kein newline-Zeichen auftauchen)

Die Daten der Textdatei werden im Hauptspeicher gepuffert (Grösse des verfügbaren RAM bei Größe der Textdatei beachten).

Hinweis: Neu anlegen **und öffnen** einer Text-Datei per Methode .CreateTextFile() des Objektes FileSystemObject

Erzeugung:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile(OpenTextFile(Kette [, Wert 1[, Wert2 [,Wert3]]])
```

Kette	String	
		Pfad (relativ oder absolut) mit Dateiname ohne Wildcards
Wert1	Integer	
	1	zum Lesen
	2	zum Schreiben
	8	für Append
Wert2	Boolean	
	True	Datei wird erzeugt, wenn nicht vorhanden
	False	Datei wird nicht erzeugt, muss also vorhanden sein (sonst wird Fehler erzeugt)
		Standard
Wert2	Integer	
	0	als ASCII öffnen
	1	als Unicode öffnen
	2	Codeart beim Öffnen laut Systemeinstellung
		Standard
Zeiger		auf offene Datei

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.WriteLine("Test");
DateiOffen.Close();
```

Eigenschaften:

.AtEndOfLine Satzzeiger bezüglich Zeilenende (End Of Line (EOL)) per newline-Zeichen wird mit jedem Lesen oder Schreiben verändert
Hinweis: newline-Zeichen per .WriteLine oder .WriteBlankLines() schreibbar immer bei satzweisem Schreiben/Lesen möglich beim zeichenweisem Schreiben/Lesen

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
var Kette = "";
while (!DateiOffen.AtEndOfLine)
{Kette = Kette + DateiOffen.ReadLine() + "\n";}
DateiOffen.Close();
alert(Kette);
```

.AtEndOfStream Satzzeiger bezüglich Dateiende, also Textstream-Ende (End Of Stream (EOS)) wird mit jedem Lesen oder Schreiben verändert

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
var Kette = "";
while (!DateiOffen.AtEndOfStream)
{Kette = Kette + DateiOffen.ReadLine() + "\n";}
DateiOffen.Close();
alert(Kette);
```

.Column aktuelle Spalte des Zeichens im Textstream, also Nummer des Zeichens im Stream nur bei **zeichenweiser** Dateiverarbeitung verwendbar, das jedes gelesene newline-Zeichen den Wert von .Column auf 1 setzt

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.WriteLine("Test");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
```



```
DateiOffen.ReadLine();
alert(DateiOffen.Column);
DateiOffen.Close();
```

.Line Anzahl der Zeilen im Textstream
sinnvoll nur verwendbar, wenn mindestens 1 newline-Zeichen in der Datei auftaucht

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.WriteLine("Test");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
DateiOffen.ReadAll();
alert(DateiOffen.Line);
DateiOffen.Close();
```

Methoden:

.Close() offene Datei schliessen
nach Schliessen der Datei sind keine Methoden mehr verwendbar, ohne die Datei vorher neu zu öffnen

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.WriteLine("Test");
DateiOffen.Close();
```

.Read() in der offenen Datei die nächsten Zeichen lesen (ab Position laut Satzzeiger)
verändert den Satzzeiger

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
alert(DateiOffen.Read(6));
DateiOffen.Close();
```

.ReadAll() offene Datei komplett auslesen (auf einen Schlag)
nur direkt nach Dateiöffnen möglich

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
alert(DateiOffen.ReadAll());
DateiOffen.Close();
```

.ReadLine() in der offene Datei die nächste Zeile lesen (ab Position laut Satzzeiger)
verändert den Satzzeiger

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
alert(DateiOffen.ReadLine());
DateiOffen.Close();
```

.Skip() in der offenen Datei die nächsten Zeichen überlesen (egal ob newline dabei ist oder nicht)
verändert den Satzzeiger

Beispiel:

```
var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
DateiOffen.Skip(1);
alert(DateiOffen.Read(1));
DateiOffen.Close();
```

.SkipLine() in der offenen Datei die nächsten Zeilen überlesen
verändert den Satzzeiger
sinnvoll nur verwendbar, wenn mindestens 1 newline-Zeichen in der Datei auftaucht

Beispiel:



```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("Test");
DateiOffen.Write("123456789");
DateiOffen.Close();
DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 1,false,0);
DateiOffen.SkipLine(1);
alert(DateiOffen.ReadLine());
DateiOffen.Close();

```

.Write() in die offenen Datei zeichenweise schreiben (ab Position laut Satzzeiger)
verändert den Satzzeiger

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.Close();

```

.WriteBlankLines() in die offenen Datei newline-Zeichen schreiben (ab Position laut Satzzeiger)
verändert den Satzzeiger

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.Write("123456789");
DateiOffen.WriteBlankLines(1);
DateiOffen.Close();

```

.WriteLine() in die offenen Datei genau 1 Zeile schreiben (ab Position laut Satzzeiger)
wobei am Ende automatisch genau ein newline-Zeichen geschrieben wird
verändert den Satzzeiger

Beispiel:

```

var DateiSystem = new ActiveXObject("Scripting.FileSystemObject");
var DateiOffen = DateiSystem.OpenTextFile("c:\\test.txt", 2,true,0);
DateiOffen.WriteLine("123456789");
DateiOffen.Close();

```

5.2.3.2.6. FileSystemObject und Windows Script Host (WSH)

Das File-System-Objekt kann teilweise nur in Verbindung mit dem Windows Script Host verwendet werden.

Dadurch werden z.B. windows-spezifische Ordner wie „Recent“, also spezielle Ordner (special folder), zugänglich.

JScript und WSH arbeiten bestens zusammen.

5.2.3.2.6.1. Beispiel: Zugriff auf Recent-Ordner

Beispiel für die Bereinigung des Recent-Ordners zum aktuellen Benutzer: Entfernen von ungültigen Verknüpfungen im Ordner.

Es soll der Ordner der letzten Dateizugriffe verarbeitet werden,
dessen Name "Recent" ist
der physisch im aktuellen Benutzerprofil liegt z.B. c:\Dokumente und Einstellungen\user_name\Recent
der von Windows verwaltet wird und daher ein Spezial-Ordner (special folder) ist
der Verknüpfungen (Ink-Dateien) auf diejenigen Dateien/Ordner enthält, auf die der Benutzer zuletzt zugegriffen hat

```

// ##### Variablen

// Name des Recent-Ordners auf Festplatte
var SpecialFolderName = 'Recent';

// Suffix der Verknüpfungen aus dem Ordner Recent
// Recent-Ordner enthält nur Verknüpfungen (Shortcuts), als Ink-Dateien
var VerknuepfungDateiSuffix = 'lnk'; // nur Kleinbuchstaben !

// WinScriptHost-Objekte
var WinScriptHost_LaufzeitUmgebung;
var WinScriptHost_DateiVerknuepfung;
var WinScriptHost_DateiVerknuepfung_ZielPfad;

// JScript-Objekte zum Dateisystem
var JScript_DateiSystem;

var SpecialFolder;
var SpecialFolder_Pfad;
var SpecialFolder_Inhalt;
var SpecialFolder_DateienSammlung;
var SpecialFolder_DateiMitPfad;
var SpecialFolder_DateiMitPfad_Suffix;

```



```

// Zähler
var AnzahlVerknuepfungen_Gefunden = 0;
var AnzahlVerknuepfungen_Geloscht = 0;
var AnzahlVerknuepfungen_ZugriffGesperrt = 0;

// ##### Funktion

function VerknuepfungPruefenUndBereinigen(DateiMitPfad)
// Eine Verknüpfung kann auf eine Datei oder einen Ordner verweisen (Ziel der Verknüpfung).

// Es wird die Verknüpfung laut DateiMitPfad überprüft, ob sie noch gültig ist.
// Gültig: Verknüpfung verweist auf eine real-existierende Datei bzw. Ordner.

// Es kann sein, dass auf Verknüpfungen im Ordner Recent wegen wegen NTFS-Zugriffsbeschränkungen
// (zum aktuellen Benutzer) nicht zugegriffen werden kann.

// Fehlschlagende Zugriffe sollten abgefangen werden: Dazu muss der Zugriff
// in die JScript-eigene Funktion try-catch eingebettet werden, um
// Ausnahmefehler während der Scriptlaufzeit auswerten zu können
// (Error-Exceptions).
{
// +++++ Anzahl der gefundenen Dateien (Verknüpfungen) erhöhen
AnzahlVerknuepfungen_Gefunden++;

// +++++ Annahme: Auf die Verknüpfungsdatei kann zugegriffen werden
var ZugriffGesperrt=false;

// +++++ Aus der Verknüpfung per WinScript einen Zeiger bilden, anhand dem im Script geprüft werden kann.
try // versuche Verknüpfung per WinScript zu erzeugen
// wenn dabei ein Laufzeit-Fehler auftritt, wird catch-Zweig aktiviert
{
// Verknüpfung als Zeiger erzeugen
WinScriptHost_DateiVerknuepfung = WinScriptHost_LaufzeitUmgebung.CreateShortcut(DateiMitPfad);

// +++++ aus erzeugter Verknüpfung den Pfad, auf den die Verknüpfung weist, holen
WinScriptHost_DateiVerknuepfung_ZielPfad = WinScriptHost_DateiVerknuepfung.TargetPath

// +++++ überprüfen, ob Zielpfad auf ein nicht-existentes Element verweist (Datei oder Ordner)

// ---- auf nicht-existente Datei prüfen
if (!JScript_DateiSystem.FileExists(WinScriptHost_DateiVerknuepfung_ZielPfad))
{
// Datei nicht existent, aber es könnte ja noch auf einen Ordner verwiesen werden

// ---- auf nicht-existent Ordner prüfen
if (!JScript_DateiSystem.FolderExists(WinScriptHost_DateiVerknuepfung_ZielPfad))
{
// weder Datei noch Ordner real vorhanden, also Verknüpfung
// ist ungültig
// wird entfernt aus dem Ordner Recent

try // versuche ungültige Verknüpfung zu löschen
// wenn dabei ein Laufzeit-Fehler auftritt, wird catch-Zweig aktiviert
{
JScript_DateiSystem.DeleteFile(DateiMitPfad);

// Anzahl der Verknüpfungen, die ungültig sind und gelöscht wurden, erhöhen
AnzahlVerknuepfungen_Geloscht++;
}
catch(LoeschenFehlgeschlagen) // Reaktion bei eingefangenen (ge-catchten) Fehler
{ZugriffGesperrt=true;}
}
}
}
catch(VerknuepfungFehlgeschlagen) // Reaktion bei eingefangenen (ge-catchten) Fehler
{ZugriffGesperrt=true;}

if (ZugriffGesperrt)
{
// Anzahl der Verknüpfungen, auf die nicht zugegriffen werden kann, erhöhen
AnzahlVerknuepfungen_ZugriffGesperrt++;
}
}
}

```



```
// ##### Prüfen und bereinigen

// ***** Erzeugung der WinScriptHost-LaufzeitUmgebung für Zugriff auf JScript_DateiSystem
var WinScriptHost_LaufzeitUmgebung = WScript.CreateObject("WScript.Shell");

// ***** Erzeugung des DateiSystem als ActiveX per JScript
JScript_DateiSystem = new ActiveXObject("Scripting.FileSystemObject");

// ***** Recent-Ordner als Objekt per WinScript erzeugt
SpecialFolder = WinScriptHost_LaufzeitUmgebung.SpecialFolders(SpecialFolderName);

// ***** Pfad des Recent-Ordners holen
SpecialFolder_Pfad = JScript_DateiSystem.GetFolder(SpecialFolder)

// ***** Inhalt des Recent-Ordners als Objekt
SpecialFolder_Inhalt = SpecialFolder_Pfad.Files;

// ***** Inhalt des Recent Ordners als Aufzählungsobjekt (Enumerator) mit internem Index
//      interner Index steht auf 1. Element der Aufzählung
SpecialFolder_DateienSammlung = new Enumerator(SpecialFolder_Inhalt)

// ***** Recent-Ordner abklappern
while (!SpecialFolder_DateienSammlung.atEnd()) // solange kein Ende erreicht
{
// +++++ aktuelles Element laut internen Index, also Verknüpfungs-Datei, einstellen
//      und internen Index danach um 1 erhöhen
SpecialFolder_DateienSammlung.moveNext();

// +++++ zum Element den Pfad holen
var SpecialFolder_DateiMitPfad = SpecialFolder_DateienSammlung.item();

//      und den Dateisuffix in Kleibuchstaben holen
SpecialFolder_DateiMitPfad_Suffix = JScript_DateiSystem.GetExtensionName(SpecialFolder_DateiMitPfad);
SpecialFolder_DateiMitPfad_Suffix = SpecialFolder_DateiMitPfad_Suffix.toLowerCase();

// +++++ Element auf lnk-Datei prüfen
if (SpecialFolder_DateiMitPfad_Suffix == VerknuepfungDateiSuffix)
{VerknuepfungPruefenUndBereinigen(SpecialFolder_DateiMitPfad);}
}

alert('Im Ordner ' + SpecialFolderName + ' wurden ' + AnzahlVerknuepfungen_Gefunden + ' Verknüpfungen gefunden !');
alert('Im Ordner ' + SpecialFolderName + ' wurden ' + AnzahlVerknuepfungen_Geloscht + ' ungültige Verknüpfungen gelöscht !');
alert('Im Ordner ' + SpecialFolderName + ' sind ' + AnzahlVerknuepfungen_ZugriffGesperrt + ' Verknüpfungen im Zugriff gesperrt !');
```

5.2.3.2.6.2. **Beispiel: Zugriff auf Registry**

Nachfolgender Quellcode muss in einer eigenständigen JS-Datei untergebracht sein.

Grund: Das Script muss in der Registry registriert werden, damit es per Kontextmenü aktivierbar ist.

Mit der Script-Registrierung wird in das Kontextmenü der Fenster des Windows Explorers einen Eintrag eingefügt, mit dem ein neuer Ordner erzeugt UND angezeigt werden kann.

Standardname des erzeugten Ordners liegt in der Variablen StandardOrdnerName.

Der Kontextmenü-Eintrag-Text liegt in der Variablen KontextMenuEintragText.

Die Erzeugung des neuen Ordners erfolgt durch Aufruf dieses Scriptes, das dazu BEREITS in der Registry registriert sein muss.

Mit der Script-De-Registrierung wird zugleich der Kontextmenü-Eintrag entfernt.

Scriptaufruf: script_js_date [ordner_name]

```
ordner_name: optional
Name des Ordners in dem ein Unterordner erzeugt werden soll
wenn nicht kodiert, so gilt:
    Ist das Script noch nicht in der Registry registriert, so wird es registriert.
    Ist ein Teil der Registry-Schlüssel zum Script nicht vorhanden, so wird
    das Script neu registriert.
    Ist das Script bereits in der Registry registriert, so wird es de-registriert.
```

// ##### Variablen

```
// +++++ Variablen, die frei wählbar sind
var StandardOrdnerName = "Neuer Ordner";
```



```

var KontextMenuEintragText = "Neuen Unterordner erzeugen";
var Script_RegistryKennung = "TestScript";

// +++++ interne Schlüssel des Script zum Registrieren und De-Registrieren (nicht verändern)
//   Lage der Schlüssel in der Registry:
//     Directory und Drive für Dateisystem-Verwaltung durch Windows
//     HKCR = Schlüssel des aktuellen Users
var RegistrySchluessel1 = "HKCR\\Directory\\shell\\" + Script_RegistryKennung + "\\";
var RegistrySchluessel2 = "HKCR\\Drive\\shell\\" + Script_RegistryKennung + "\\";

// +++++ WinScriptHost-Objekte
var WinScriptHost_LaufzeitUmgebung;
var WinScriptHost_ScriptDateiname;
var WinScriptHost_ScriptAufruf_ParameterListe;
var WinScriptHost_ScriptAufruf_ParameterAnzahl;

// +++++ sonstiges
var Parameter;

// ##### Funktionen

function Registry_SchluesselSuchen()
// sucht in der Registry die Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
// liefert nur dann true, wenn beide Schlüssel in der Registry vorhanden sind
{
// Annahme: Schlüssel ist in der Registry vorhanden
var Gefunden1 = true; // RegistrySchluessel1 vorhanden
var Gefunden2 = true; // RegistrySchluessel2 vorhanden

// Schlüssel in Registry suchen
// nicht gefunden wird als Laufzeitfehler abgefangen !

// RegistrySchluessel1 suchen
try {WinScriptHost_LaufzeitUmgebung.RegRead(RegistrySchluessel1);}
catch(Nichtgefunden){Gefunden1 = false;}

// RegistrySchluessel2 suchen
try {WinScriptHost_LaufzeitUmgebung.RegRead(RegistrySchluessel2);}
catch(Nichtgefunden){Gefunden2 = false;}

// true, wenn beide Schlüssel vorhanden sind
return (Gefunden1 && Gefunden2);
}

function Registry_ScriptRegistrator(Flag)
// Flag true, so registrieren, also Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
//   in der Registry erzeugen
//   false, so de-registrieren, also Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
//   in der Registry löschen
{
// +++++ per JScript das Dateisystem referenzieren
var JScript_DateiSystem = new ActiveXObject("Scripting.FileSystemObject");

// +++++ weitere Schlüssel für Registrator definieren
var RegistrySchluessel3 = RegistrySchluessel1 + 'command\\';
var RegistrySchluessel4 = RegistrySchluessel2 + 'command\\';
var RegistryWert3 = 'wscript.exe \'' + WinScriptHost_ScriptDateiname + '\" \"%L\\\"';
var RegistryWert4 = 'wscript.exe \'' + WinScriptHost_ScriptDateiname + '\" \"%L\\\"';

// +++++ Meldungstexte erzeugen
var Kette1 = WScript.ScriptFullName + " wurde";
var Kette2 = "! \\Im Kontextmenü aller Ordner und Laufwerke steht nun der Menüpunkt \"
+ KontextMenuEintragText + \"";
var Kette3 = "";

// +++++ Registration je nach Art
if (Flag)
{
// registrieren
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel1,KontextMenuEintragText);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel3,RegistryWert3);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel2,KontextMenuEintragText);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel4,RegistryWert4);

```



```

}
else
{
// de-registrieren
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel3);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel1);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel4);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel2);

Kette1 = Kette1 + "de-";
Kette3 = " nicht mehr";
}

// +++++ WHS-Analogon zur JScript-Funktion alert()
WScript.Echo(Kette1 + " installiert" + Kette2 + Kette3 + " bereit !");
}

// #####

// +++++ WHS-Laufzeitumgebung erzeugen
var WinScriptHost_LaufzeitUmgebung = WScript.CreateObject("WScript.Shell");

// +++++ Dateiname der JS-Datei holen, in der dieses Script steht
var WinScriptHost_ScriptDateiname = WScript.ScriptFullName; // Dateiname der JS-Datei

// +++++ Parameter zum Script-Aufruf holen
var WinScriptHost_ScriptAufruf_ParameterListe = WScript.Arguments;
var WinScriptHost_ScriptAufruf_ParameterAnzahl = WinScriptHost_ScriptAufruf_ParameterListe.length;

// +++++ Registry-Schlüssel in der Registry suchen, also prüfen, ob das Script bereits
// registriert ist oder nicht
// gesucht werden Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel1
if (Registry_SchluesselSuchen())
{
// beide Schlüssel vorhanden (RegistrySchluessel1 und RegistrySchluessel2 gefunden)
// Script ist bereits registriert

// +++++ auf Anzahl der Parameter bei Script-Aufruf prüfen
if (WinScriptHost_ScriptAufruf_ParameterAnzahl == 0)
{
// kein Parameter, also Script aus Registry austragen und Kontextmenü-Eintrag löschen
Registry_ScriptRegistratur(false);
}
else
{
// +++++ mindestens 1 Parameter bei Scriptaufruf
// aber nur den 1. Parameter holen, also Pfad des Ordners, in dem ein neuer Unterordner erzeugt werden soll
Parameter = WinScriptHost_ScriptAufruf_ParameterListe(0);

// prüfen ob Ordner real vorhanden ist
if (JScript_DateiSystem.FolderExists(Parameter))
{
// Ordner real vorhanden
// also Pfad des Ordners erweitern um den Namen des neuen Unterordners laut StandardOrdnerName
Parameter = Parameter + StandardOrdnerName;

// prüfen ob Pfad MIT neuem Unterordner NICHT real vorhanden ist
if (!JScript_DateiSystem.FolderExists(Parameter))
{
// neuer Unterordner ist NICHT real vorhanden, also erzeugen
JScript_DateiSystem.CreateFolder(Parameter);

// und den neuen Unterordner als Fenster anzeigen
WinScriptHost_LaufzeitUmgebung.Run("explorer.exe /select," + Parameter);

// 1,3 Sekunden warten
WScript.Sleep(1300);

// Ordnername editierbar machen durch Simulation des Tastendruckes F2
WinScriptHost_LaufzeitUmgebung.SendKeys("{F2}");
}
}
}
}
}
}

```



```

else
{
//+++++ mindesten 1 Schlüssel fehlt, also Script in Registry eintragen
Registry_ScriptRegistrierung(true);
}

```

5.2.3.2.6.3. **Beispiel: Ordner erzeugen**

Erzeugung eines Ordners per Kontextmenü des Windows Explorers.

Nachfolgender Quellcode muss in einer eigenständigen JS-Datei untergebracht sein.

Grund: Das Script muss in der Registry registriert werden, damit es per Kontextmenü aktivierbar ist.

Mit der Script-Registrierung wird in das Kontextmenü der Fenster des Windows Explorers ein Eintrag eingefügt, mit dem ein neuer Ordner erzeugt UND angezeigt werden kann.

Standardname des erzeugten Ordners liegt in der Variablen StandardOrdnerName.

Der Kontextmenü-Eintrag-Text liegt in der Variablen KontextMenuEintragText.

Die Erzeugung des neuen Ordners erfolgt durch Aufruf dieses Scriptes, das dazu BEREITS in der Registry registriert sein muss.

Mit der Script-De-Registrierung wird zugleich der Kontextmenü-Eintrag entfernt.

Scriptaufruf: `script_js_date [ordner_name]`

```

ordner_name: optional
    Name des Ordners in dem ein Unterordner erzeugt werden soll
    wenn nicht kodiert, so gilt:
        Ist das Script noch nicht in der Registry registriert, so wird es registriert.
        Ist ein Teil der Registry-Schlüssel zum Script nicht vorhanden, so wird
        das Script neu registriert.
        Ist das Script bereits in der Registry registriert, so wird es de-registriert.

```

//##### Variablen

```

//+++++ Variablen, die frei wählbar sind
var StandardOrdnerName = "Neuer Ordner";
var KontextMenuEintragText = "Neuen Unterordner erzeugen";
var Script_RegistryKennung = "TestScript";

```

//+++++ interne Schlüssel des Script zum Registrieren und De-Registrieren (nicht verändern)

```

// Lage der Schlüssel in der Registry:
// Directory und Drive für Dateisystem-Verwaltung durch Windows
// HKCR = Schlüssel des aktuellen Users
var RegistrySchlüssel1 = "HKCR\\Directory\\shell\\" + Script_RegistryKennung + "\\";
var RegistrySchlüssel2 = "HKCR\\Drive\\shell\\" + Script_RegistryKennung + "\\";

```

```

//+++++ WinScriptHost-Objekte
var WinScriptHost_LaufzeitUmgebung;
var WinScriptHost_ScriptDateiname;
var WinScriptHost_ScriptAufruf_ParameterListe;
var WinScriptHost_ScriptAufruf_ParameterAnzahl;

```

```

//+++++ sonstiges
var Parameter;

```

//##### Funktionen

```

function Registry_SchlüsselSuchen()
// sucht in der Registry die Schlüssel laut Variablen RegistrySchlüssel1 und RegistrySchlüssel2
// liefert nur dann true, wenn beide Schlüssel in der Registry vorhanden sind
{
// Annahme: Schlüssel ist in der Registry vorhanden
var Gefunden1 = true; // RegistrySchlüssel1 vorhanden
var Gefunden2 = true; // RegistrySchlüssel2 vorhanden

// Schlüssel in Registry suchen
// nicht gefunden wird als Laufzeitfehler abgefangen !

// RegistrySchlüssel1 suchen
try {WinScriptHost_LaufzeitUmgebung.RegRead(RegistrySchlüssel1);}
catch(Nichtgefunden){Gefunden1 = false;}

```



```

// RegistrySchluessel2 suchen
try {WinScriptHost_LaufzeitUmgebung.RegRead(RegistrySchluessel2);}
catch(Nichtgefunden){Gefunden2 = false;}

// true, wenn beide Schlüssel vorhanden sind
return (Gefunden1 && Gefunden2);
}

function Registry_ScriptRegistratur(Flag)
// Flag true, so registrieren, also Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
// in der Registry erzeugen
// false, so de-registrieren, also Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
// in der Registry löschen
{
// +++++ per JScript das Dateisystem referenzieren
var JScript_DateiSystem = new ActiveXObject("Scripting.FileSystemObject");

// +++++ weitere Schlüssel für Registratur definieren
var RegistrySchluessel3 = RegistrySchluessel1 + 'command\';
var RegistrySchluessel4 = RegistrySchluessel2 + 'command\';
var RegistryWert3 = 'wscript.exe \'' + WinScriptHost_ScriptDateiname + '\' "%L\';
var RegistryWert4 = 'wscript.exe \'' + WinScriptHost_ScriptDateiname + '\' "%L\';

// +++++ Meldungstexte erzeugen
var Kette1 = WScript.ScriptFullName + " wurde";
var Kette2 = "! \Im Kontextmenü aller Ordner und Laufwerke steht nun der Menüpunkt \\"
+ KontextMenuEintragText + "\\";
var Kette3 = "";

// +++++ Registration je nach Art
if (Flag)
{
// registrieren
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel1,KontextMenuEintragText);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel3,RegistryWert3);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel2,KontextMenuEintragText);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel4,RegistryWert4);
}
else
{
// de-registrieren
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel3);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel1);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel4);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel2);

Kette1 = Kette1 + "de-";
Kette3 = " nicht mehr";
}

// +++++ WHS-Analogon zur JScript-Funktion alert()
WScript.Echo(Kette1 + " installiert" + Kette2 + Kette3 + " bereit !");
}

#####

// +++++ WHS-Laufzeitumgebung erzeugen
var WinScriptHost_LaufzeitUmgebung = WScript.CreateObject("WScript.Shell");

// +++++ Dateiname der JS-Datei holen, in der dieses Script steht
var WinScriptHost_ScriptDateiname = WScript.ScriptFullName; // Dateiname der JS-Datei

// +++++ Parameter zum Script-Aufruf holen
var WinScriptHost_ScriptAufruf_ParameterListe = WScript.Arguments;
var WinScriptHost_ScriptAufruf_ParameterAnzahl = WinScriptHost_ScriptAufruf_ParameterListe.length;

// +++++ Registry-Schlüssel in der Registry suchen, also prüfen, ob das Script bereits
// registriert ist oder nicht
// gesucht werden Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel1
if (Registry_SchluesselSuchen())
{
// beide Schlüssel vorhanden (RegistrySchluessel1 und RegistrySchluessel2 gefunden)
// Script ist bereits registriert

```



```

//+++++ auf Anzahl der Parameter bei Script-Aufruf prüfen
if (WinScriptHost_ScriptAufruf_ParameterAnzahl == 0)
{
// kein Parameter, also Script aus Registry austragen und Kontextmenü-Eintrag löschen
Registry_ScriptRegistrierung(false);
}
else
{
//+++++ mindestens 1 Parameter bei Scriptaufruf
// aber nur den 1. Parameter holen, also Pfad des Ordners, in dem ein neuer Unterordner erzeugt werden soll
Parameter = WinScriptHost_ScriptAufruf_ParameterListe(0);

// prüfen ob Ordner real vorhanden ist
if (JScript_DateiSystem.FolderExists(Parameter))
{
// Ordner real vorhanden
// also Pfad des Ordners erweitern um den Namen des neuen Unterordners laut StandardOrdnerName
Parameter = Parameter + StandardOrdnerName;

// prüfen ob Pfad MIT neuem Unterordner NICHT real vorhanden ist
if (!JScript_DateiSystem.FolderExists(Parameter))
{
// neuer Unterordner ist NICHT real vorhanden, also erzeugen
JScript_DateiSystem.CreateFolder(Parameter);

// und den neuen Unterordner als Fenster anzeigen
WinScriptHost_LaufzeitUmgebung.Run("explorer.exe /select," + Parameter);

// 1,3 Sekunden warten
WScript.Sleep(1300);

// Ordnername editierbar machen durch Simulation des Tastendruckes F2
WinScriptHost_LaufzeitUmgebung.SendKeys("{F2}");
}
}
}
else
{
//+++++ mindesten 1 Schlüssel fehlt, also Script in Registry eintragen
Registry_ScriptRegistrierung(true);
}

```

5.2.3.2.6.4. **Beispiel: Lokales Programm starten**

Windows Explorer als lokales Programm starten

Nachfolgender Quellcode muss in einer eigenständigen JS-Datei untergebracht sein.

Grund: Das Script muss in der Registry registriert werden, damit es per Kontextmenü aktivierbar ist.

Mit der Script-Registrierung wird in das Kontextmenü der Fenster des Windows Explorers einen Eintrag eingefügt, mit dem ein neuer Ordner erzeugt UND angezeigt werden kann.

Standardname des erzeugten Ordners liegt in der Variablen StandardOrdnerName.

Der Kontextmenü-Eintrag-Text liegt in der Variablen KontextMenuEintragText.

Die Erzeugung des neuen Ordners erfolgt durch Aufruf dieses Scriptes, das dazu BEREITS in der Registry registriert sein muss.

Mit der Script-De-Registrierung wird zugleich der Kontextmenü-Eintrag entfernt.

Scriptaufruf: `script_js_date [ordner_name]`

ordner_name: optional

Name des Ordners in dem ein Unterordner erzeugt werden soll
wenn nicht kodiert, so gilt:

Ist das Script noch nicht in der Registry registriert, so wird es registriert.

Ist ein Teil der Registry-Schlüssel zum Script nicht vorhanden, so wird das Script neu registriert.

Ist das Script bereits in der Registry registriert, so wird es de-registriert.

//##### Variablen



```
// +++++ Variablen, die frei wählbar sind
var StandardOrdnerName = "Neuer Ordner";
var KontextMenuEintragText = "Neuen Unterordner erzeugen";
var Script_RegistryKennung = "TestScript";

// +++++ interne Schlüssel des Script zum Registrieren und De-Registrieren (nicht verändern)
// Lage der Schlüssel in der Registry:
// Directory und Drive für Dateisystem-Verwaltung durch Windows
// HKCR = Schlüssel des aktuellen Users
var RegistrySchluessel1 = "HKCR\\Directory\\shell\\" + Script_RegistryKennung + "\\";
var RegistrySchluessel2 = "HKCR\\Drive\\shell\\" + Script_RegistryKennung + "\\";

// +++++ WinScriptHost-Objekte
var WinScriptHost_LaufzeitUmgebung;
var WinScriptHost_ScriptDateiname;
var WinScriptHost_ScriptAufruf_ParameterListe;
var WinScriptHost_ScriptAufruf_ParameterAnzahl;

// +++++ sonstiges
var Parameter;

// ##### Funktionen

function Registry_SchluesselSuchen()
// sucht in der Registry die Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
// liefert nur dann true, wenn beide Schlüssel in der Registry vorhanden sind
{
// Annahme: Schlüssel ist in der Registry vorhanden
var Gefunden1 = true; // RegistrySchluessel1 vorhanden
var Gefunden2 = true; // RegistrySchluessel2 vorhanden

// Schlüssel in Registry suchen
// nicht gefunden wird als Laufzeitfehler abgefangen !

// RegistrySchluessel1 suchen
try {WinScriptHost_LaufzeitUmgebung.RegRead(RegistrySchluessel1);}
catch(Nichtgefunden){Gefunden1 = false;}

// RegistrySchluessel2 suchen
try {WinScriptHost_LaufzeitUmgebung.RegRead(RegistrySchluessel2);}
catch(Nichtgefunden){Gefunden2 = false;}

// true, wenn beide Schlüssel vorhanden sind
return (Gefunden1 && Gefunden2);
}

function Registry_ScriptRegistatur(Flag)
// Flag true, so registrieren, also Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
// in der Registry erzeugen
// false, so de-registrieren, also Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
// in der Registry löschen
{
// +++++ per JScript das Dateisystem referenzieren
var JScript_DateiSystem = new ActiveXObject("Scripting.FileSystemObject");

// +++++ weitere Schlüssel für Registrator definieren
var RegistrySchluessel3 = RegistrySchluessel1 + 'command\\';
var RegistrySchluessel4 = RegistrySchluessel2 + 'command\\';
var RegistryWert3 = 'wscript.exe \'' + WinScriptHost_ScriptDateiname + '\" \"%L\\\"';
var RegistryWert4 = 'wscript.exe \'' + WinScriptHost_ScriptDateiname + '\" \"%L\\\"';

// +++++ Meldungstexte erzeugen
var Kette1 = WScript.ScriptFullName + " wurde";
var Kette2 = "! \Im Kontextmenü aller Ordner und Laufwerke steht nun der Menüpunkt \"
+ KontextMenuEintragText + \"\";
var Kette3 = "";

// +++++ Registration je nach Art
if (Flag)
{
// registrieren
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel1,KontextMenuEintragText);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel3,RegistryWert3);

```



```

WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel2,KontextMenuEintragText);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel4,RegistryWert4);
}
else
{
// de-registrieren
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel3);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel1);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel4);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel2);

Kette1 = Kette1 + "de-";
Kette3 = " nicht mehr";
}

// +++++ WHS-Analogon zur JScript-Funktion alert()
WScript.Echo(Kette1 + " installiert" + Kette2 + Kette3 + " bereit !");
}

// #####

// +++++ WHS-Laufzeitumgebung erzeugen
var WinScriptHost_LaufzeitUmgebung = WScript.CreateObject("WScript.Shell");

// +++++ Dateiname der JS-Datei holen, in der dieses Script steht
var WinScriptHost_ScriptDateiname = WScript.ScriptFullName; // Dateiname der JS-Datei

// +++++ Parameter zum Script-Aufruf holen
var WinScriptHost_ScriptAufruf_ParameterListe = WScript.Arguments;
var WinScriptHost_ScriptAufruf_ParameterAnzahl = WinScriptHost_ScriptAufruf_ParameterListe.length;

// +++++ Registry-Schlüssel in der Registry suchen, also prüfen, ob das Script bereits
// registriert ist oder nicht
// gesucht werden Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel1
if (Registry_SchluesselSuchen())
{
// beide Schlüssel vorhanden (RegistrySchluessel1 und RegistrySchluessel2 gefunden)
// Script ist bereits registriert

// +++++ auf Anzahl der Parameter bei Script-Aufruf prüfen
if (WinScriptHost_ScriptAufruf_ParameterAnzahl == 0)
{
// kein Parameter, also Script aus Registry austragen und Kontextmenü-Eintrag löschen
Registry_ScriptRegistratur(false);
}
else
{
// +++++ mindestens 1 Parameter bei Scriptaufruf
// aber nur den 1. Parameter holen, also Pfad des Ordners, in dem ein neuer Unterordner erzeugt werden soll
Parameter = WinScriptHost_ScriptAufruf_ParameterListe(0);

// prüfen ob Ordner real vorhanden ist
if (JScript_DateiSystem.FolderExists(Parameter))
{
// Ordner real vorhanden
// also Pfad des Ordners erweitern um den Namen des neuen Unterordners laut StandardOrdnerName
Parameter = Parameter + StandardOrdnerName;

// prüfen ob Pfad MIT neuem Unterordner NICHT real vorhanden ist
if (!JScript_DateiSystem.FolderExists(Parameter))
{
// neuer Unterordner ist NICHT real vorhanden, also erzeugen
JScript_DateiSystem.CreateFolder(Parameter);

// und den neuen Unterordner als Fenster anzeigen
WinScriptHost_LaufzeitUmgebung.Run("explorer.exe /select," + Parameter);

// 1,3 Sekunden warten
WScript.Sleep(1300);

// Ordnername editierbar machen durch Simulation des Tastendruckes F2
WinScriptHost_LaufzeitUmgebung.SendKeys("{F2}");
}
}
}
}

```



```

}
}
else
{
// +++++ mindesten 1 Schlüssel fehlt, also Script in Registry eintragen
Registry_ScriptRegistrierung(true);
}

```

5.2.3.2.6.5. **Beispiel: Tastensimulation**

Tastensimulation für den Windows Explorer

Nachfolgender Quellcode muss in einer eigenständigen JS-Datei untergebracht sein.

Grund: Das Script muss in der Registry registriert werden, damit es per Kontextmenü aktivierbar ist.

Mit der Script-Registrierung wird in das Kontextmenü der Fenster des Windows Explorers ein Eintrag eingefügt, mit dem ein neuer Ordner erzeugt UND angezeigt werden kann.

Standardname des erzeugten Ordners liegt in der Variablen StandardOrdnerName.

Der Kontextmenü-Eintrag-Text liegt in der Variablen KontextMenuEintragText.

Die Erzeugung des neuen Ordners erfolgt durch Aufruf dieses Scriptes, das dazu BEREITS in der Registry registriert sein muss.

Mit der Script-De-Registrierung wird zugleich der Kontextmenü-Eintrag entfernt.

Scriptaufruf: `script_js_date [ordner_name]`

ordner_name: optional
 Name des Ordners in dem ein Unterordner erzeugt werden soll
 wenn nicht kodiert, so gilt:
 Ist das Script noch nicht in der Registry registriert, so wird es registriert.
 Ist ein Teil der Registry-Schlüssel zum Script nicht vorhanden, so wird das Script neu registriert.
 Ist das Script bereits in der Registry registriert, so wird es de-registriert.

// ##### Variablen

```

// +++++ Variablen, die frei wählbar sind
var StandardOrdnerName = "Neuer Ordner";
var KontextMenuEintragText = "Neuen Unterordner erzeugen";
var Script_RegistryKennung = "TestScript";

```

// +++++ interne Schlüssel des Script zum Registrieren und De-Registrieren (nicht verändern)

```

// Lage der Schlüssel in der Registry:
// Directory und Drive für Dateisystem-Verwaltung durch Windows
// HKCR = Schlüssel des aktuellen Users
var RegistrySchlüssel1 = "HKCR\\Directory\\shell\\" + Script_RegistryKennung + "\\";
var RegistrySchlüssel2 = "HKCR\\Drive\\shell\\" + Script_RegistryKennung + "\\";

```

```

// +++++ WinScriptHost-Objekte
var WinScriptHost_LaufzeitUmgebung;
var WinScriptHost_ScriptDateiname;
var WinScriptHost_ScriptAufruf_ParameterListe;
var WinScriptHost_ScriptAufruf_ParameterAnzahl;

```

```

// +++++ sonstiges
var Parameter;

```

// ##### Funktionen

```

function Registry_SchlüsselSuchen()
// sucht in der Registry die Schlüssel laut Variablen RegistrySchlüssel1 und RegistrySchlüssel2
// liefert nur dann true, wenn beide Schlüssel in der Registry vorhanden sind
{
// Annahme: Schlüssel ist in der Registry vorhanden
var Gefunden1 = true; // RegistrySchlüssel1 vorhanden
var Gefunden2 = true; // RegistrySchlüssel2 vorhanden

```

```

// Schlüssel in Registry suchen
// nicht gefunden wird als Laufzeitfehler abgefangen !

```

```

// RegistrySchlüssel1 suchen

```



```

try {WinScriptHost_LaufzeitUmgebung.RegRead(RegistrySchluessel1);}
catch(Nichtgefunden){Gefunden1 = false;}

// RegistrySchluessel2 suchen
try {WinScriptHost_LaufzeitUmgebung.RegRead(RegistrySchluessel2);}
catch(Nichtgefunden){Gefunden2 = false;}

// true, wenn beide Schlüssel vorhanden sind
return (Gefunden1 && Gefunden2);
}

function Registry_ScriptRegistrator(Flag)
// Flag true, so registrieren, also Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
//   in der Registry erzeugen
//   false, so de-registrieren, also Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
//   in der Registry löschen
{
// +++++ per JScript das Dateisystem referenzieren
var JScript_DateiSystem = new ActiveXObject("Scripting.FileSystemObject");

// +++++ weitere Schlüssel für Registrator definieren
var RegistrySchluessel3 = RegistrySchluessel1 + 'command\';
var RegistrySchluessel4 = RegistrySchluessel2 + 'command\';
var RegistryWert3 = 'wscript.exe \'' + WinScriptHost_ScriptDateiname + '\' "%L\';
var RegistryWert4 = 'wscript.exe \'' + WinScriptHost_ScriptDateiname + '\' "%L\';

// +++++ Meldungstexte erzeugen
var Kette1 = WScript.ScriptFullName + " wurde";
var Kette2 = "! Im Kontextmenü aller Ordner und Laufwerke steht nun der Menüpunkt \''
    + KontextMenuEintragText + "'";
var Kette3 = "";

// +++++ Registration je nach Art
if (Flag)
{
// registrieren
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel1,KontextMenuEintragText);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel3,RegistryWert3);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel2,KontextMenuEintragText);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel4,RegistryWert4);
}
else
{
// de-registrieren
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel3);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel1);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel4);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel2);

Kette1 = Kette1 + "de-";
Kette3 = " nicht mehr";
}

// +++++ WHS-Analogon zur JScript-Funktion alert()
WScript.Echo(Kette1 + " installiert" + Kette2 + Kette3 + " bereit !");
}

// #####

// +++++ WHS-Laufzeitumgebung erzeugen
var WinScriptHost_LaufzeitUmgebung = WScript.CreateObject("WScript.Shell");

// +++++ Dateiname der JS-Datei holen, in der dieses Script steht
var WinScriptHost_ScriptDateiname = WScript.ScriptFullName; // Dateiname der JS-Datei

// +++++ Parameter zum Script-Aufruf holen
var WinScriptHost_ScriptAufruf_ParameterListe = WScript.Arguments;
var WinScriptHost_ScriptAufruf_ParameterAnzahl = WinScriptHost_ScriptAufruf_ParameterListe.length;

// +++++ Registry-Schlüssel in der Registry suchen, also prüfen, ob das Script bereits
//   registriert ist oder nicht
//   gesucht werden Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel1
if (Registry_SchluesselSuchen())
{

```



```

// beide Schlüssel vorhanden (RegistrySchlüssel1 und RegistrySchlüssel2 gefunden)
// Script ist bereits registriert

// +++++ auf Anzahl der Parameter bei Script-Aufruf prüfen
if (WinScriptHost_ScriptAufruf_ParameterAnzahl == 0)
{
// kein Parameter, also Script aus Registry austragen und Kontextmenü-Eintrag löschen
Registry_ScriptRegistratur(false);
}
else
{
// +++++ mindestens 1 Parameter bei Scriptaufruf
// aber nur den 1. Parameter holen, also Pfad des Ordners, in dem ein neuer Unterordner erzeugt werden soll
Parameter = WinScriptHost_ScriptAufruf_ParameterListe(0);

// prüfen ob Ordner real vorhanden ist
if (JScript_DateiSystem.FolderExists(Parameter))
{
// Ordner real vorhanden
// also Pfad des Ordners erweitern um den Namen des neuen Unterordners laut StandardOrdnerName
Parameter = Parameter + StandardOrdnerName;

// prüfen ob Pfad MIT neuem Unterordner NICHT real vorhanden ist
if (!JScript_DateiSystem.FolderExists(Parameter))
{
// neuer Unterordner ist NICHT real vorhanden, also erzeugen
JScript_DateiSystem.CreateFolder(Parameter);

// und den neuen Unterordner als Fenster anzeigen
WinScriptHost_LaufzeitUmgebung.Run("explorer.exe /select," + Parameter);

// 1,3 Sekunden warten
WScript.Sleep(1300);

// Ordnername editierbar machen durch Simulation des Tastendruckes F2
WinScriptHost_LaufzeitUmgebung.SendKeys("{F2}");
}
}
}
else
{
// +++++ mindesten 1 Schlüssel fehlt, also Script in Registry eintragen
Registry_ScriptRegistratur(true);
}

```

5.2.3.2.6.6. **Beispiel: Zugriff auf Kontextmenü des Windows Explorers**

Kontextmenü-Eintrag für den Windows Explorer erzeugen

Nachfolgender Quellcode muss in einer eigenständigen JS-Datei untergebracht sein.

Grund: Das Script muss in der Registry registriert werden, damit es per Kontextmenü aktivierbar ist.

Mit der Script-Registrierung wird in das Kontextmenü der Fenster des Windows Explorers einen Eintrag eingefügt, mit dem ein neuer Ordner erzeugt UND angezeigt werden kann.

Standardname des erzeugten Ordners liegt in der Variablen StandardOrdnerName.

Der Kontextmenü-Eintrag-Text liegt in der Variablen KontextMenuEintragText.

Die Erzeugung des neuen Ordners erfolgt durch Aufruf dieses Scriptes, das dazu BEREITS in der Registry registriert sein muss.

Mit der Script-De-Registrierung wird zugleich der Kontextmenü-Eintrag entfernt.

Scriptaufruf: `script_js_date [ordner_name]`

`ordner_name`: optional

Name des Ordners in dem ein Unterordner erzeugt werden soll
wenn nicht kodiert, so gilt:

Ist das Script noch nicht in der Registry registriert, so wird es registriert.

Ist ein Teil der Registry-Schlüssel zum Script nicht vorhanden, so wird das Script neu registriert.

Ist das Script bereits in der Registry registriert, so wird es de-registriert.



```
// ##### Variablen

// +++++ Variablen, die frei wählbar sind
var StandardOrdnerName = "Neuer Ordner";
var KontextMenuEintragText = "Neuen Unterordner erzeugen";
var Script_RegistryKennung = "TestScript";

// +++++ interne Schlüssel des Script zum Registrieren und De-Registrieren (nicht verändern)
// Lage der Schlüssel in der Registry:
// Directory und Drvie für Dateisystem-Verwaltung durch Windows
// HKCR = Schlüssel des aktuellen Users
var RegistrySchluessel1 = "HKCR\Directory\shell\" + Script_RegistryKennung + "\";
var RegistrySchluessel2 = "HKCR\Drive\shell\" + Script_RegistryKennung + "\";

// +++++ WinScriptHost-Objekte
var WinScriptHost_LaufzeitUmgebung;
var WinScriptHost_ScriptDateiname;
var WinScriptHost_ScriptAufruf_ParameterListe;
var WinScriptHost_ScriptAufruf_ParameterAnzahl;

// +++++ sonstiges
var Parameter;

// ##### Funktionen

function Registry_SchluesselSuchen()
// sucht in der Registry die Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
// liefert nur dann true, wenn beide Schlüssel in der Registry vorhanden sind
{
// Annahme: Schlüssel ist in der Registry vorhanden
var Gefunden1 = true; // RegistrySchluessel1 vorhanden
var Gefunden2 = true; // RegistrySchluessel2 vorhanden

// Schlüssel in Registry suchen
// nicht gefunden wird als Laufzeitfehler abgefangen !

// RegistrySchluessel1 suchen
try {WinScriptHost_LaufzeitUmgebung.RegRead(RegistrySchluessel1);}
catch(Nichtgefunden){Gefunden1 = false;}

// RegistrySchluessel2 suchen
try {WinScriptHost_LaufzeitUmgebung.RegRead(RegistrySchluessel2);}
catch(Nichtgefunden){Gefunden2 = false;}

// true, wenn beide Schlüssel vorhanden sind
return (Gefunden1 && Gefunden2);
}

function Registry_ScriptRegistrator(Flag)
// Flag true, so registrieren, also Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
// in der Registry erzeugen
// false, so de-registrieren, also Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
// in der Registry löschen
{
// +++++ per JScript das Dateisystem referenzieren
var JScript_DateiSystem = new ActiveXObject("Scripting.FileSystemObject");

// +++++ weitere Schlüssel für Registrator definieren
var RegistrySchluessel3 = RegistrySchluessel1 + 'command\';
var RegistrySchluessel4 = RegistrySchluessel2 + 'command\';
var RegistryWert3 = 'wscript.exe \'' + WinScriptHost_ScriptDateiname + '\' "%L"';
var RegistryWert4 = 'wscript.exe \'' + WinScriptHost_ScriptDateiname + '\' "%L"';

// +++++ Meldungstexte erzeugen
var Kette1 = WScript.ScriptFullName + " wurde";
var Kette2 = '! \Im Kontextmenü aller Ordner und Laufwerke steht nun der Menüpunkt \''
+ KontextMenuEintragText + "\'";
var Kette3 = "";

// +++++ Registration je nach Art
if (Flag)
{
// registrieren
```



```

WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel1,KontextMenuEintragText);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel3,RegistryWert3);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel2,KontextMenuEintragText);
WinScriptHost_LaufzeitUmgebung.RegWrite(RegistrySchluessel4,RegistryWert4);
}
else
{
// de-registrieren
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel3);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel1);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel4);
WinScriptHost_LaufzeitUmgebung.RegDelete(RegistrySchluessel2);

Kette1 = Kette1 + "de-";
Kette3 = " nicht mehr";
}

// +++++ WHS-Analogon zur JScript-Funktion alert()
WScript.Echo(Kette1 + " installiert" + Kette2 + Kette3 + " bereit !");
}

// #####

// +++++ WHS-Laufzeitumgebung erzeugen
var WinScriptHost_LaufzeitUmgebung = WScript.CreateObject("WScript.Shell");

// +++++ Dateiname der JS-Datei holen, in der dieses Script steht
var WinScriptHost_ScriptDateiname = WScript.ScriptFullName; // Dateiname der JS-Datei

// +++++ Parameter zum Script-Aufruf holen
var WinScriptHost_ScriptAufruf_ParameterListe = WScript.Arguments;
var WinScriptHost_ScriptAufruf_ParameterAnzahl = WinScriptHost_ScriptAufruf_ParameterListe.length;

// +++++ Registry-Schlüssel in der Registry suchen, also prüfen, ob das Script bereits
// registriert ist oder nicht
// gesucht werden Schlüssel laut Variablen RegistrySchluessel1 und RegistrySchluessel2
if (Registry_SchluesselSuchen())
{
// beide Schlüssel vorhanden (RegistrySchluessel1 und RegistrySchluessel2 gefunden)
// Script ist bereits registriert

// +++++ auf Anzahl der Parameter bei Script-Aufruf prüfen
if (WinScriptHost_ScriptAufruf_ParameterAnzahl == 0)
{
// kein Parameter, also Script aus Registry austragen und Kontextmenü-Eintrag löschen
Registry_ScriptRegistrierung(false);
}
else
{
// +++++ mindestens 1 Parameter bei Scriptaufruf
// aber nur den 1. Parameter holen, also Pfad des Ordners, in dem ein neuer Unterordner erzeugt werden soll
Parameter = WinScriptHost_ScriptAufruf_ParameterListe(0);

// prüfen ob Ordner real vorhanden ist
if (JScript_DateiSystem.FolderExists(Parameter))
{
// Ordner real vorhanden
// also Pfad des Ordners erweitern um den Namen des neuen Unterordners laut StandardOrdnerName
Parameter = Parameter + StandardOrdnerName;

// prüfen ob Pfad MIT neuem Unterordner NICHT real vorhanden ist
if (!JScript_DateiSystem.FolderExists(Parameter))
{
// neuer Unterordner ist NICHT real vorhanden, also erzeugen
JScript_DateiSystem.CreateFolder(Parameter);

// und den neuen Unterordner als Fenster anzeigen
WinScriptHost_LaufzeitUmgebung.Run("explorer.exe /select," + Parameter);

// 1,3 Sekunden warten
WScript.Sleep(1300);

// Ordnername editierbar machen durch Simulation des Tastendruckes F2
WinScriptHost_LaufzeitUmgebung.SendKeys("{F2}");
}
}
}
}
}

```



```

}
}
}
}
else
{
//+++++ mindesten 1 Schlüssel fehlt, also Script in Registry eintragen
Registry_ScriptRegistratur(true);
}

```

5.2.3.2.6.7. Beispiel: Zugriff auf PATH-Variablen

Zugriff auf PATH-Variablen

```

// WinScriptHost-Objekte
var WinScriptHost_LaufzeitUmgebung = WScript.CreateObject("WScript.Shell");
var WinScriptHost_PATHVariable = WinScriptHost_LaufzeitUmgebung.ExpandEnvironmentStrings("%path%");

// JScript-Dateisystem
var JScript_DateiSystem = new ActiveXObject("Scripting.FileSystemObject");

var PfadTrenner = ";";
var TeilPfad;
var TeilPfade_Anzahl = 0; // ab 1
var TeilPfade_Ungültige = "";

var Zahler = 0;

// "\" ersetzen durch Leerkette
WinScriptHost_PATHVariable = WinScriptHost_PATHVariable.replace(/\\/g, "");

// Semikolon zählen (Pfad-Trenner) durch zeichenweises abklappern
for (Zahler = 0; Zahler < WinScriptHost_PATHVariable.length; Zahler++)
{
    if (WinScriptHost_PATHVariable.charAt(Zahler) == PfadTrenner) {TeilPfade_Anzahl++;}
}

// Feld der Teilpfade erzeugen
var TeilPfade_Feld = new Array(TeilPfade_Anzahl-1); // Index ab 0

// Path-Kette nach Feld: Feldelement liegt zwischen Pfadtrenner
TeilPfade_Feld = WinScriptHost_PATHVariable.split(PfadTrenner);

// Teilpfad auf realen Ordner prüfen
for (Zahler = 0; Zahler < TeilPfade_Feld.length; Zahler++)
{
    // nächster Teilpfad
    TeilPfad=TeilPfade_Feld[Zahler];

    // prüfen auf realen Ordner
    if (JScript_DateiSystem.FolderExists(TeilPfad) == false)
    {
        // kein realer Ordner vorhanden
        TeilPfade_Ungültige = TeilPfade_Ungültige + TeilPfad + "\n";
    }

    // Anzeige aller ungültigen Teilpfade im Path
    if (TeilPfade_Ungültige == "")
    {WScript.Echo("Teilpfade existieren real !");}
    else
    WScript.Echo("Ungültige Teilpfade sind \n\n" + TeilPfade_Ungültige);
}

```

5.2.4. Windows Media Player 7.1 und Internet Explorer

Der Windows Media Player dient als Addon u.a. zur Wiedergabe diverser Medien anhand von Medien-Dateien oder einer Playliste von Medien-Dateien.

Der Mediaplayer 7.1 wurde exemplarisch gewählt, da im SDK die Scriptprogrammierung offeriert wurde.

Es besteht die Möglichkeit, dass Nachfolgeversionen des Mediaplayer 7.1 ebenfalls diese Scriptprogrammierung unterstützen.

Vermutlich wurde aber ab MediaPlayer 10 Inkompatibilität hergestellt: Man muss es eben austesten.

Der Windows Media Player ist im HTML-Dokument per HTML und optional per JScript ansteuerbar, wobei der Programmierungsaufwand in JScript erheblich ist. Eine sehr einfache, aber wenig dynamische Programmierungsvariante ist beim Objekt bgsound zu finden (siehe dort).

Der Windows Media Player 7.1 wird für den Netscape im Gegensatz zum Windows Media Player 6.4 als Plugin **nicht** mehr unterstützt. Ab dem IE 6.x werden generell keine Plugins mehr unterstützt. Microsoft favorisiert ihr eigenes ActiveX-Konzept und überlässt z.T.

Fremdherstellern die Programmierung von ActiveX-Controls z.B. als Addon zum Internet Explorer. Vorteil ist die Normung per Active-X-

