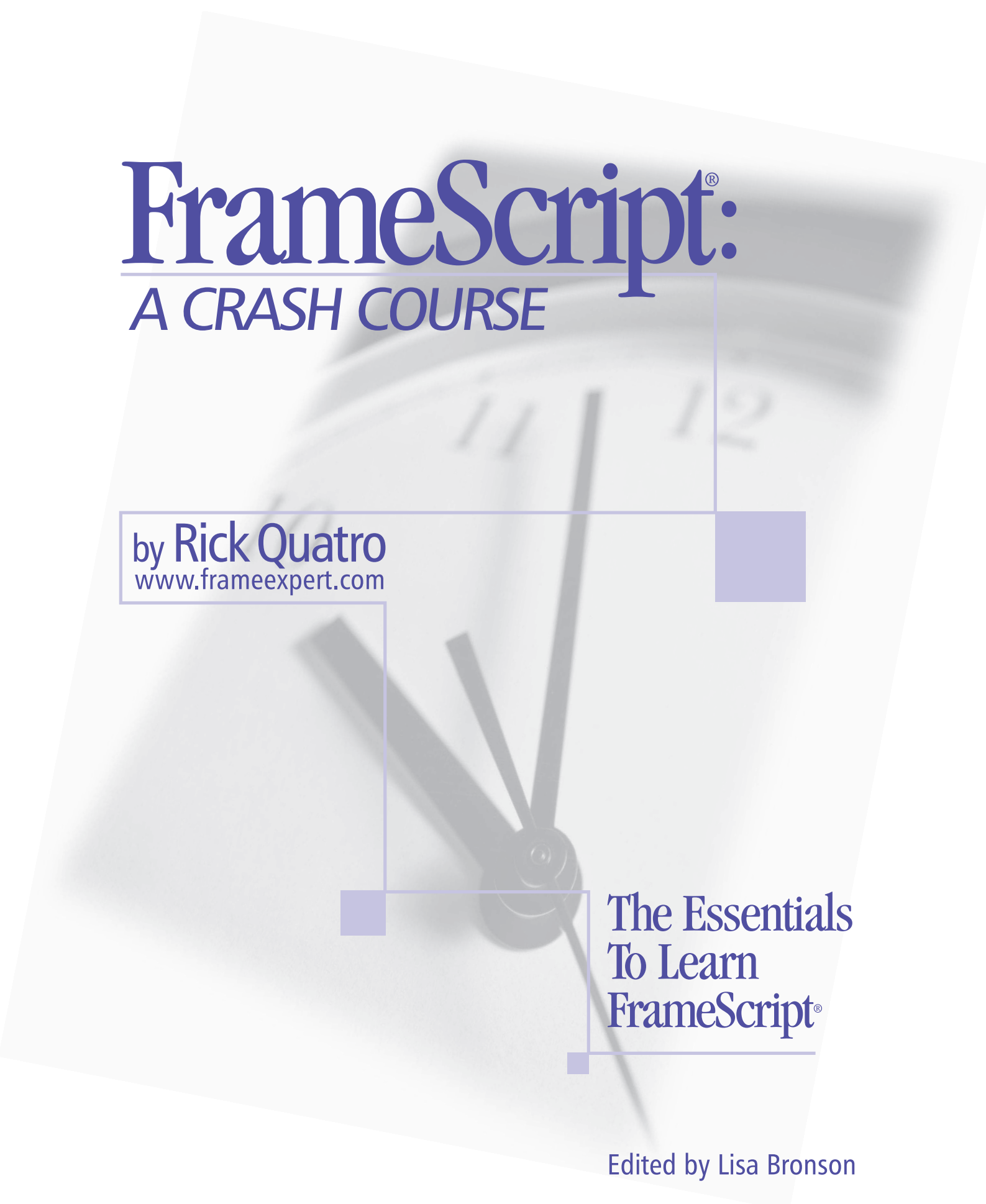
FrameScript®:
A CRASH COURSE



by Rick Quatro
www.frameexpert.com

The Essentials To Learn FrameScript®

Edited by Lisa Bronson

FrameScript: A Crash Course
The Essentials To Learn FrameScript

by Rick Quatro http://www.frameexpert.com

Edited by Lisa Bronson

Translated by / Übersetzt von Stefan Keller

FrameScript®: A Crash Course

First Edition published September 2002 in the United States of America.

Published by Carmen Publishing, 557 County Line Road, Hamlin, New York 14464, rick@frameexpert.com, http://www.frameexpert.com

Inhalt

1 Erste Schritte

1 Erste Schritte

In diesem Kapitel lernen Sie die Grundlagen von FrameScript kennen und erhalten eine Einführung in die Vorgehensweisen zum Schreiben produktiver Skripts.

Was ist ein Skript?

In der einfachsten Form besteht ein Skript aus einem oder mehreren FrameScript-Befehlen. Hier ein Beispiel für ein einzeiliges Skript, das Rasterlinien im aktiven Dokument anzeigt:

```
Set ActiveDoc.ViewGrid = 1;
```

In der Regel enthalten Ihre Skripts wesentlich mehr Zeilen, aber schon Skripts mit etwa einem Dutzend Zeilen können durchaus sehr effektiv sein.

Skripts bestehen aus FrameScript-Befehlen, FrameMaker-Objekten und Eigenschaftswerten der Objekte. In unserem einzeiligen Beispiel ist das Wort **Set** ein FrameScript-Befehl, mit dem ein Eigenschaftswert festgelegt wird. *Befehle* sind in der Regel Wörter, die eine Aktion an einem Objekt durchführen. **ActiveDoc** ist ein FrameMaker-Objekt, das für das auf dem Bildschirm angezeigte Dokument mit dem Eingabefokus steht. **ViewGrid** ist eine der zahlreichen Eigenschaften, über die ein Dokument verfügt. Sie entspricht dem Menübefehl **Darstellung > Rasterlinien** in der FrameMaker-Benutzeroberfläche. **1** ist schließlich der Wert, der dieser Eigenschaft zugewiesen wird, und bedeutet so viel wie **True** oder »ein«.

Befehle und Parameter

FrameScript-Befehle sind Anweisungen bestehend aus einem oder zwei Wörtern, die beschreiben, was der Befehl tut. Der Befehl **MsgBox** zeigt dem Benutzer zum Beispiel ein Meldungsfeld an, der Befehl **New AFrame** erstellt einen verankerten Rahmen. Alle FrameScript-Befehle müssen mit einem Semikolon enden (;). Die einzigen Ausnahmen sind die Befehle **If/Else/EndIf**, **Loop/EndLoop**, **Sub/EndSub** und **Event/EndEvent**.

Befehle besitzen erforderliche oder optionale Parameter. Ein *Parameter* ist eine spezielle Information, die ein Befehl zur korrekten Ausführung verwendet. Der Befehl **MsgBox** benötigt zum Beispiel einen Zeichenfolgeparameter, der die anzuzeigende Meldung enthält.

Codebeispiel 1--1

```
MsgBox 'Zeichenfolgeparameter mit der Meldung. ';
```

In diesem Fall ist die Zeichenfolge ein erforderlicher Parameter, da das Meldungsfeld ohne diesen Parameter nicht angezeigt wird. Einige Parameter sind optional. Der Befehl **MsgBox** hat den optionalen Parameter **Mode**, der den Stil des Meldungsfelds beeinflusst.

Codebeispiel 1--2

```
MsgBox 'Zeichenfolgeparameter mit der Meldung. ' Mode(Warn);
```

Der Wert **Warn** für den Parameter **Mode** legt fest, dass ein Warnsymbol im Meldungsfeld angezeigt wird.

Wenn der Parameter nicht angegeben wird, verwenden die meisten Befehle voreingestellte Werte. Der Befehl **MsgBox** verwendet beispielsweise den Standardwert **Mode(Note)**, wenn **Mode** nicht angegeben wird. Das heißt, die folgenden beiden Befehle sind identisch, und beide Meldungsfelder zeigen das Hinweissymbol:

Codebeispiel 1--3

```
MsgBox 'Zeichenfolgeparameter mit der Meldung. ';
MsgBox 'Zeichenfolgeparameter mit der Meldung. ' Mode(Note);
```

Viele Befehle verfügen über mehrere Parameter. Manche Parameter geben Informationen *an* den Befehl weiter, andere Parameter geben Informationen *aus* dem Befehl zurück. Das folgende Beispiel zeigt den Befehl **MsgBox** mit den Parametern **Mode(YesNo)** und **Button**. Der Parameter **Button** gibt je nachdem, auf welche Schaltfläche der Benutzer klickt, die entsprechenden Informationen des Befehls **MsgBox** zurück.

Codebeispiel 1--4

```
MsgBox 'Klicken Sie auf eine Schaltflaeche. '
  Mode(YesNo) Button(vButton);
If (vButton = YesButton)
  MsgBox 'Sie haben auf Yes geklickt. ';
Else
  MsgBox 'Sie haben auf No geklickt. ';
EndIf
```

Wenn Sie einen Parameter verwenden, um Informationen aus einem Befehl zurückzugeben, weisen Sie dem Parameter eine Variable zu, in diesem Fall **vButton**.

Die Reihenfolge der Parameter eines Befehls spielt keine Rolle. Die folgenden beiden Befehle sind zum Beispiel identisch:

Codebeispiel 1--5

```
MsgBox 'Klicken Sie auf eine Schaltflaeche. '
  Mode(YesNo) Button(vButton);

MsgBox Button(vButton) Mode(YesNo)
  'Klicken Sie auf eine Schaltflaeche. ';
```

Wenn Sie eine bestimmte Aufgabe mit FrameScript ausführen möchten, können Sie in der *FrameScript Quick Reference* nach den entsprechenden Befehlen suchen. Eine ausführliche Zusammenstellung der FrameScript-Befehle und -Parameter finden Sie in der *FrameScript Scriptwriter's Reference.*

Leerräume und Kommentare

FrameScript ist nicht sehr restriktiv bezüglich der Formatierung von Skripts, sofern jeder Befehl mit einem Semikolon endet und die Syntax korrekt ist. Allerdings sollten Sie an die Lesbarkeit Ihrer Skripts denken, damit sie leichter zu bearbeiten und zu pflegen sind. Dies können Sie zum Beispiel durch Leerräume und Kommentare erreichen.

Im folgenden Beispiel sehen Sie ein Skript, das alle Absätze im Dokument durchläuft und den Text aller Heading1-Absätze in die Konsole schreibt.

Codebeispiel 1--6

```
If ActiveDoc = 0
MsgBox 'Kein aktives Dokument. ';
LeaveSub;
Else
Set vCurrentDoc = ActiveDoc;
EndIf
Loop ForEach(Pgf) In(vCurrentDoc) LoopVar(vPgf)
If vPgf.Name = 'Heading1'
Write Console vPgf.Text;
EndIf
EndLoop
```

Obwohl dieses Skript aus weniger als 12 Zeilen besteht, ist es kaum nachvollziehbar, da es völlig ohne Leerräume geschrieben ist. Effektive Leerräume in einem Skript lassen sich auf zweierlei Arten verwirklichen. Erstens können Sie Codeblöcke durch Leerzeilen voneinander trennen. Zweitens können Sie Zeilen innerhalb von Steuerstrukturen wie **If/Else/EndIf** und **Loop/EndLoop** einrücken.

Kommentare sind nützlich, da sie Ihr Skript dokumentieren, damit auch andere Benutzer Ihren Code nachvollziehen können, sollte dieser später bearbeitet werden müssen. Kommentare und Leerräume sind auch gut geeignet, um einzelne Blöcke Ihres Codes übertragbar zu machen, damit Sie sie in anderen Skripts wiederverwenden können.

Hier nun dasselbe Skript wie oben, allerdings mittels Leerräumen und Kommentaren verbessert.

Codebeispiel 1--7

```
// Prüfen, ob aktives Dokument vorhanden.
If (ActiveDoc = 0)
  // Falls kein Dokument vorhanden, Meldung anzeigen und Skript beenden.
  MsgBox 'Kein aktives Dokument. ';
  LeaveSub;
Else
  // Variable für aktives Dokument setzen.
  Set vCurrentDoc = ActiveDoc;
EndIf

// Jeden Absatz im Dokument durchlaufen.
Loop ForEach(Pgf) In(vCurrentDoc) LoopVar(vPgf)
  // Prüfen, ob Absatzstil Heading1 ist.
  If (vPgf.Name = 'Heading1')
    // Text in die Konsole ausgeben.
    Write Console vPgf.Text;
  EndIf
EndLoop
```

Michaels Tipp Zusätzlich zu den Kommentaren und Einrückungen wurden die Parameter der **If**-Abfrage in Klammern gesetzt. Dies ist nicht erforderlich, verbessert aber ebenfalls die Lesbarkeit und hilft Fehlern vorzubeugen.

Sie können zwei Arten von Kommentaren verwenden: Kommentarzeilen und Kommentarblöcke. Im obigen Beispiel wurden Kommentarzeilen verwendet. Diese beginnen mit zwei Schrägstrichen (//). Alles zwischen den beiden Schrägstrichen und dem Zeilenende wird als Kommentar interpretiert. Kommentarblöcke beginnen mit einem Schrägstrich gefolgt von einem Stern (/*) und enden mit einem Stern gefolgt von einem Schrägstrich (*/). Kommentarblöcke können sich über mehrere Zeilen erstrecken. Alle Zeichen zwischen den beiden Sternen werden als Kommentar interpretiert. Eine typische Anwendung für Kommentarblöcke sind mehrzeilige Informationen am Anfang eines Skripts.

Codebeispiel 1--8

```
/* SaveAsMif.fsl Version 1.2
Copyright 2002, Carmen Publishing. Alle Rechte vorbehalten.

Dies ist ein Ereignisskript, das bei jedem Speichervorgang eine Kopie des aktuellen
Dokuments als MIF-Datei speichert. Das Skript muss installiert und nicht ausgeführt
werden. Sie können ein Initialisierungsskript verwenden, um dieses Skript bei jedem
Start von FrameMaker automatisch zu installieren.

Weitere Informationen erhalten Sie beim Autor unter rick@frameexpert.com.
*/
```

Sie sollten für einzeilige Kommentare in Ihren Skripts keine Kommentarblöcke verwenden. Betrachten Sie folgendes Beispiel mit Kommentarblöcken:

```
/* Jeden Absatz im Dokument durchlaufen. */
Loop ForEach(Pgf) In(vCurrentDoc) LoopVar(vPgf)
  /* Prüfen, ob Absatzstil Heading1 ist. */
  If (vPgf.Name = 'Heading1')
    /* Text in die Konsole ausgeben. */
    Write Console vPgf.Text;
  EndIf
EndLoop
```

Beim Entwickeln und Testen des Skripts müssen Sie möglicherweise einen ganzen Block von Code auskommentieren. Wenn Sie Kommentarblöcke für einzeilige Kommentare verwendet haben, ist dies jedoch nicht ohne Weiteres möglich. Zum Beispiel können Sie vor und nach dem oben stehenden Codeblock keine Kommentarblöcke einfügen.

```
/* Versuchen Sie, diesen Codeblock auszukommentieren. Geht nicht!
/* Jeden Absatz im Dokument durchlaufen. */
Loop ForEach(Pgf) In(vCurrentDoc) LoopVar(vPgf)
  /* Prüfen, ob Absatzstil Heading1 ist. */
  If (vPgf.Name = 'Heading1')
    /* Text in die Konsole ausgeben. */
    Write Console vPgf.Text;
  EndIf
EndLoop
*/
```

Wenn Sie aber im gesamten Code Kommentarzeilen verwenden, können Sie später mit Kommentarblöcken ganze Abschnitte des Codes auskommentieren.

Codebeispiel 1--9

```
/* Hier können Sie diesen Codeblock erfolgreich auskommentieren.
// Jeden Absatz im Dokument durchlaufen.
Loop ForEach(Pgf) In(vCurrentDoc) LoopVar(vPgf)
  // Prüfen, ob Absatzstil Heading1 ist.
  If (vPgf.Name = 'Heading1')
    // Text in die Konsole ausgeben.
    Write Console vPgf.Text;
  EndIf
EndLoop
*/
```

Zwei Arten von Skripts

Standardskripts

Skripts lassen sich in zwei Hauptformen unterteilen. Eine ist ein *Standardskript* aus einem oder mehreren Befehlen, auf die eine oder mehrere Subroutinen folgen können. Standardskripts werden bei jeder Ausführung in den Arbeitsspeicher geladen und bei Beendigung wieder aus diesem entfernt. In einem Standardskript wird der Code, der nicht Teil einer Subroutine ist, der *Skriptkörper* genannt. Der Skriptkörper muss stets vor allen Subroutinen oder Ereignissen im Skript eingefügt werden.

Codebeispiel 1--10

```
// Skriptkörper.
If (ActiveDoc = 0)
  MsgBox 'Kein aktives Dokument. ';
  LeaveSub;
Else
  Set vCurrentDoc = ActiveDoc;
EndIf

// Subroutine aufrufen.
Run ProcessMarkers;

// Ende des Skriptkörpers. Subroutinen folgen nach dem Skriptkörper.

Sub ProcessMarkers
//
...
//
EndSub
```

Ereignisskripts

Die zweite Form von Skripts sind die *Ereignisskripts*. Ein Ereignisskript umfasst ein oder mehrere Ereignisse bzw. Subroutinen. In einem Ereignisskript gibt es keinen Skriptkörper. Ereignisskripts werden nicht ausgeführt, sondern installiert. Beim Installieren eines Ereignisskripts wird es in den Arbeitsspeicher geladen. Dort »wartet« das Skript, bis es durch ein Ereignis ausgelöst wird. Bei einem Ereignis kann es sich um einen Menübefehl, einen Tastaturbefehl oder ein FrameMaker-Ereignis handeln, zum Beispiel das Öffnen oder Speichern eines Dokuments. Ein Ereignisskript bleibt so lange resident im Arbeitsspeicher, bis es deinstalliert wird oder FrameMaker beendet wird.

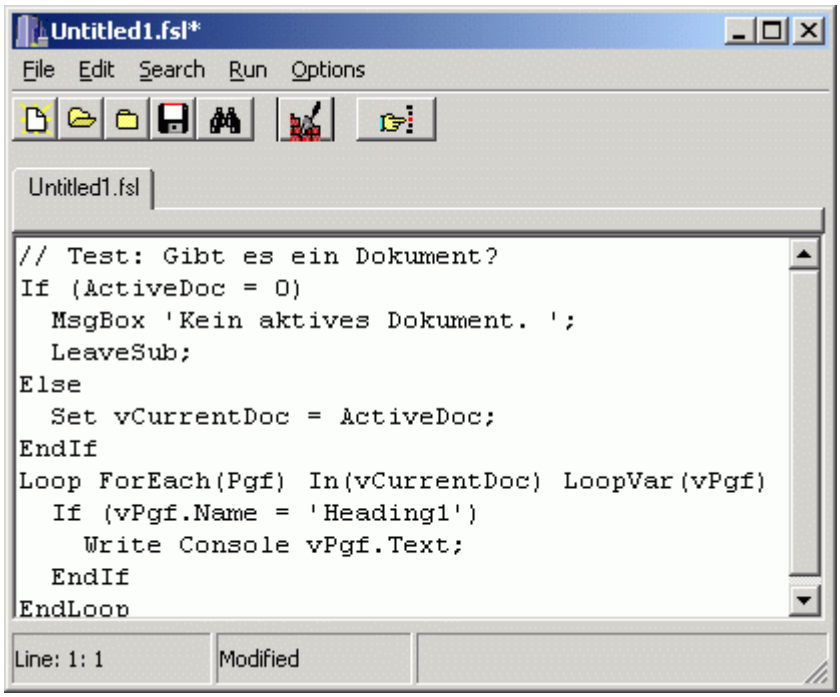
Es folgt ein Beispiel eines Ereignisskripts, das auf ein FrameMaker-Ereignis reagiert. Es wird beim Speichern eines Dokuments ausgelöst und speichert automatisch eine Kopie des Dokuments im MIF-Format. Beachten Sie, dass das Skript keinen Skriptkörper besitzt.

Codebeispiel 1--11

```
Event NotePostSaveDoc
//
// Suchen nach der .fm-Erweiterung im Dateinamen.
Find String('.fm') InString(FileName) Backward NoCase
  ReturnStatus(vFound) ReturnPos(vPos);
If (vFound)
  // Falls Erweiterung gefunden, diese löschen und durch .mif ersetzen.
  Get String FromString(FileName) EndPos(vPos-1) NewVar(vNameNoExt);
  Set vMifName = vNameNoExt+'.mif';
Else
  // Falls keine Erweiterung vorhanden, .mif anfügen.
  Set vMifName = FileName+'.mif';
EndIf
// Dokument als MIF-Datei speichern.
Save Document DocObject(FrameDoc) File(vMifName)
  FileType(SaveFmtInterchange);
//
EndEvent
```

Skripts schreiben

Sie können Ihre Skripts im Skriptfenster schreiben. Standardskripts können Sie vom Skriptfenster aus auch ausführen. Zum Öffnen des Skriptfensters wählen Sie **FrameScript > Script Window**.



Mit den Befehlen und Schaltflächen im Skriptfenster können Sie Ihre Skripts speichern und ausführen. In der folgenden Tabelle finden Sie eine Beschreibung der Schaltflächen im Skriptfenster:

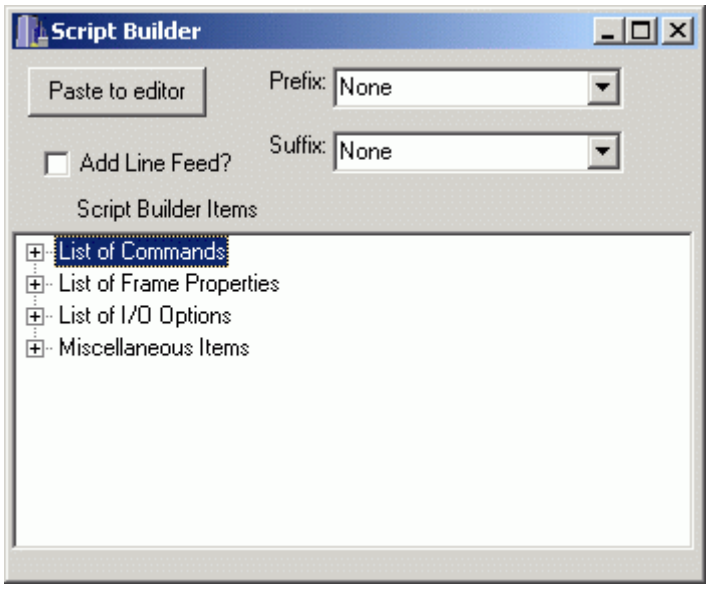
Schaltfläche	Funktion	Beschreibung	Entsprechender Menübefehl
	Neues Skript	Öffnet eine neue Skript-Registerkarte im Skriptfenster	File > New
	Skript öffnen	Zeigt ein Dialogfeld zum Öffnen eines vorhandenen Skripts	File > Open
	Skript schließen	Schließt das Skript im Vordergrund	File > Close (Strg+W)
	Skript speichern	Speichert das Skript im Vordergrund	File > Save (Strg+S)
	Text suchen	Durchsucht das Skript im Vordergrund nach Text	Search > Find (Strg+F)
	Script Builder	Öffnet das Fenster des Script Builders	Run > Script Builder (Strg+B)
	Skript ausführen	Führt das Skript im Vordergrund aus	Run > Script (Strg+R)

Das Skriptfenster ist eine gut geeignete Umgebung zum Entwerfen von kleinen Skripts und zur Fehlersuche. Beachten Sie bei Verwendung des Skriptfensters folgende Richtlinien:

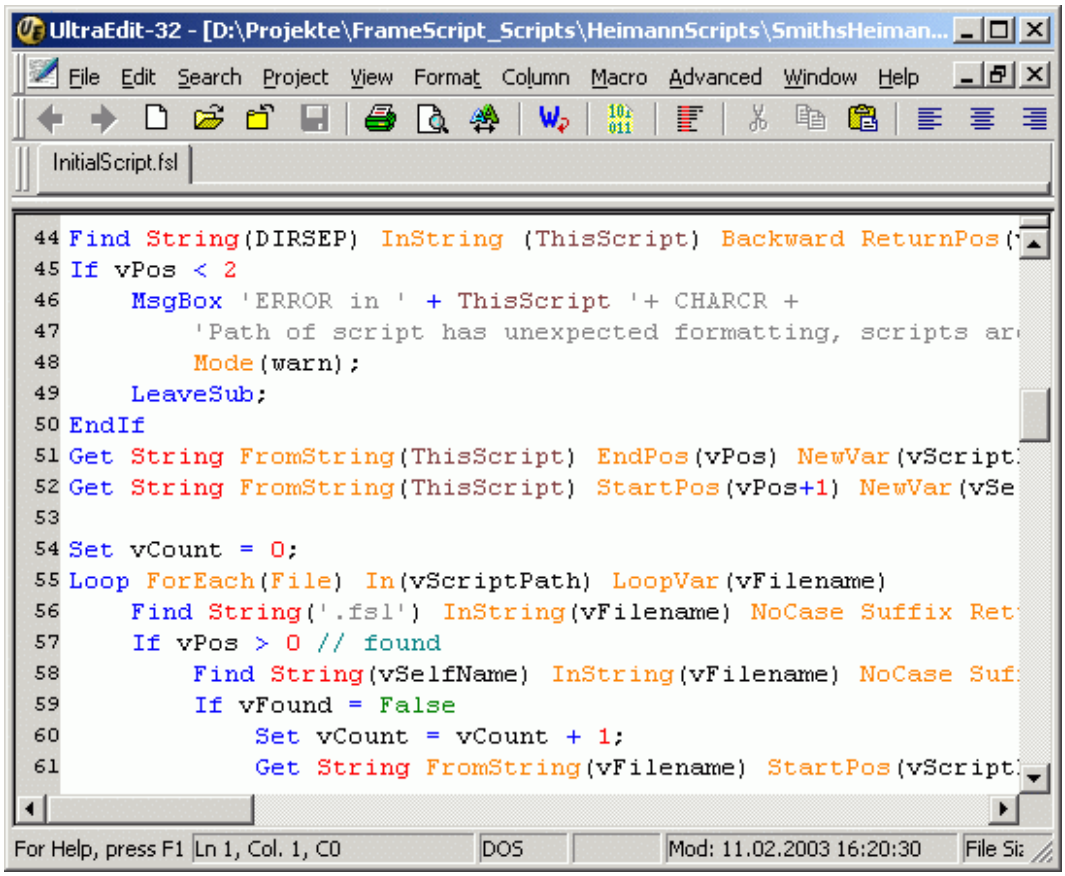
- **Speichern Sie Ihre Skripts regelmäßig**. Da Sie vom Skriptfenster aus auch ungespeicherte Skripts ausführen können, gehen alle nicht gespeicherten Änderungen verloren, falls das Skript FrameMaker zum Absturz bringt.

- **Speichern Sie Ihre Änderungen, bevor Sie zu einer anderen Registerkarte wechseln**. In einigen Fällen gehen die Inhalte einer Registerkarte beim Wechseln zu einer anderen Registerkarte verloren. Der Inhalt wird dann durch eine 0 (Null) ersetzt. Falls ein Skript durch eine 0 ersetzt wurde, schließen Sie das Skript, ohne es zu speichern.

- **Verwenden Sie den Script Builder als Unterstützung zur Skriptsyntax**. Öffnen Sie den Script Builder, indem Sie **Run > Script Builder** wählen oder auf das Script Builder-Symbol klicken. Der Script Builder enthält eine Liste mit FrameScript-Befehlen und -Eigenschaften, die Sie direkt in das Skriptfenster einfügen können.



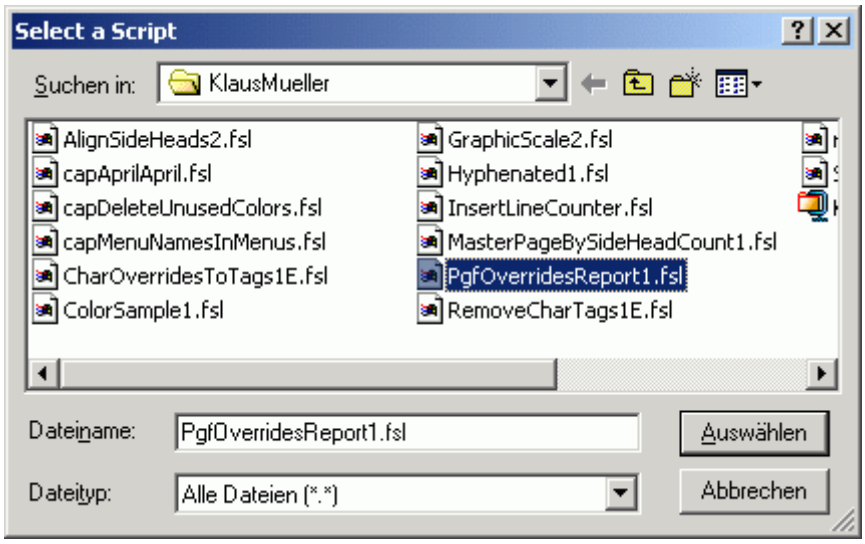
Beim Entwickeln längerer und komplexer Skripts sollten Sie einen externen Texteditor zum Schreiben der Skripts verwenden. Ein gutes Shareware-Programm für den PC ist zum Beispiel UltraEdit (http://www.ultraedit.com), das etwa $30 kostet. Dieser Editor enthält viele nützliche Funktionen zum Schreiben von Skripts, zum Beispiel Zeilennummerierung und Syntaxmarkierung. Die Syntaxmarkierung stellt unterschiedliche Komponenten der FrameScript-Sprache in unterschiedlichen Farben dar. Kommentare werden beispielsweise grün angezeigt, Befehle sind blau, Parameter sind rot, usw. Die Abbildung unten ist ein Screenshot eines Skripts in UltraEdit.



Skripts ausführen

In der Regel speichern Sie Ihre Skripts in einer Datei. Hierbei muss es sich um eine reine ASCII-Textdatei handeln. Beim Speichern eines Skripts verwenden Sie normalerweise die Erweiterung .FSL. Ein Standardskript, das Sie bereits gespeichert haben, können Sie über den Befehl **FrameScript > Run** ausführen.

Wählen Sie das gewünschte Skript, und klicken Sie auf **Auswählen**.

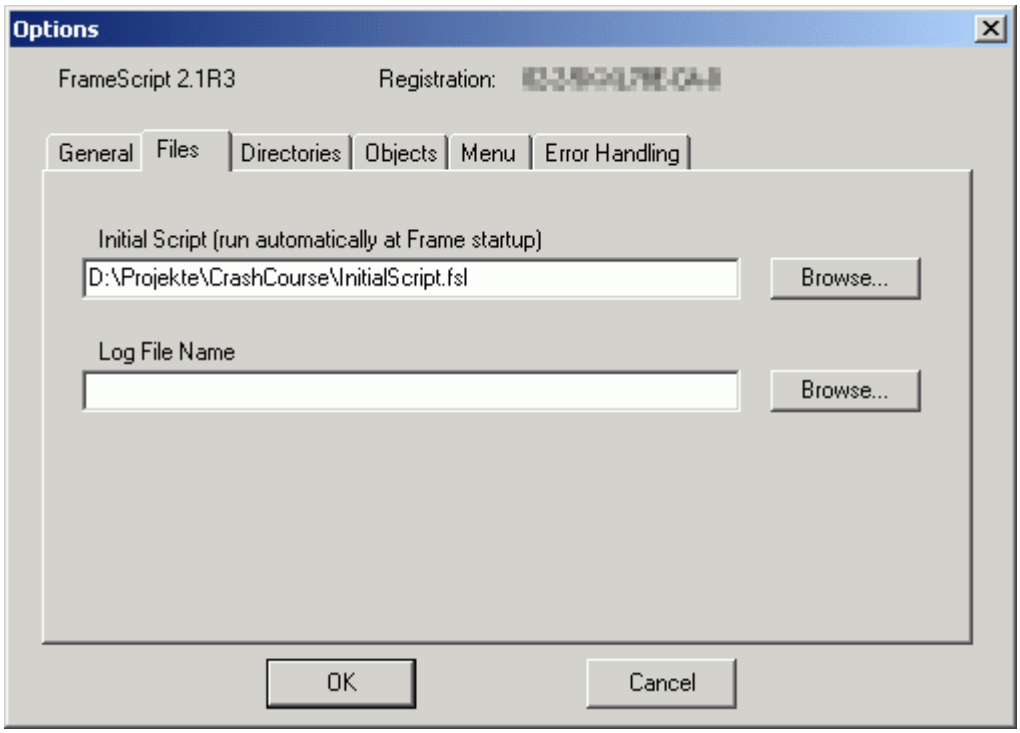


Sie können auch Standardskripts ausführen, die gerade im Skriptfenster geöffnet sind, indem Sie den Befehl **Run > Script** wählen oder auf die Schaltfläche **Run** klicken.

Skripts automatisch ausführen

FrameScript kann ein Skript beim Start von FrameMaker automatisch ausführen. Dieses Skript wird *Initialisierungsskript* genannt. Bei einem Initialisierungsskript kann es sich um ein beliebiges Standardskript handeln. Meistens wird dies verwendet, um beim Start von FrameMaker andere Skripts zu installieren. Siehe »Skripts automatisch installieren« auf Seite 1-16. Alle im Initialisierungsskript festgelegten Variablen sind globale Variablen und bleiben resident im Arbeitsspeicher, so lange FrameMaker ausgeführt wird.

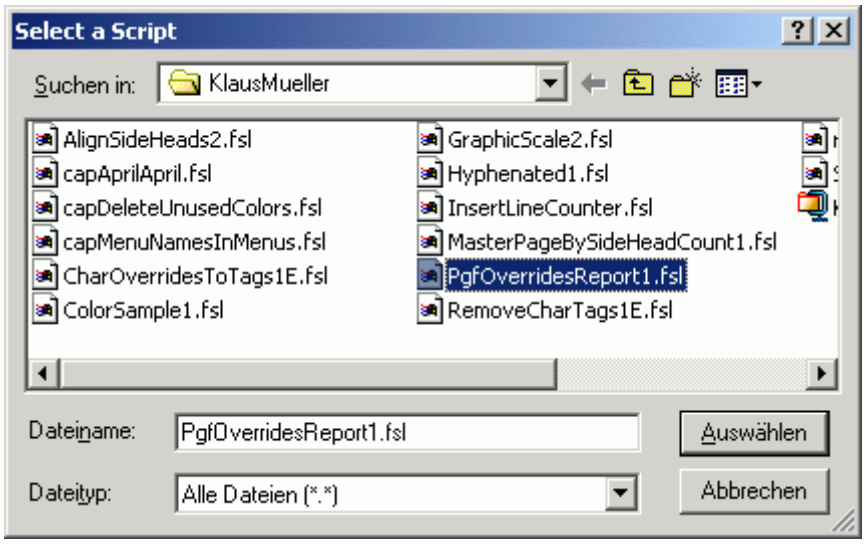
Um ein Skript als Initialisierungsskript zu definieren, wählen Sie **FrameScript > Options**, und klicken Sie auf die Registerkarte **Files**.



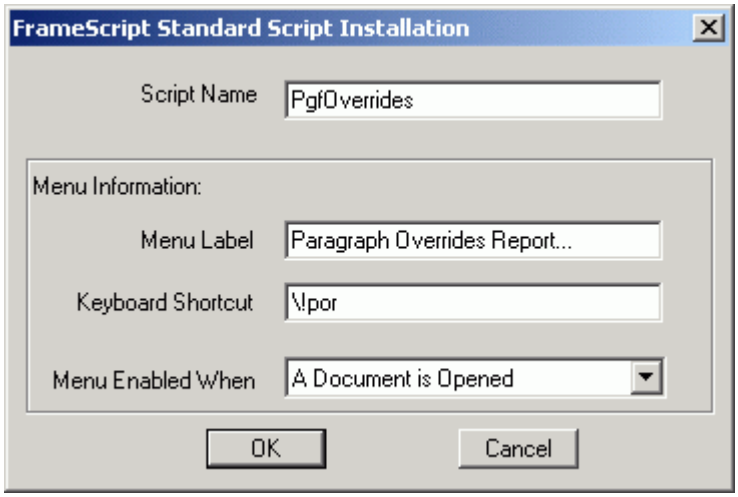
Klicken Sie neben dem Feld **Initial Script** auf die Schaltfläche **Browse**, und suchen Sie das Skript, das Sie als Initialisierungsskript verwenden möchten. Klicken Sie auf **OK**, um das Dialogfeld zu schließen. Das Initialisierungsskript wird beim nächsten Start von FrameMaker ausgeführt.

Skripts installieren und deinstallieren

Sie können Standardskripts installieren und diese dann über das Untermenü **FrameScript > Scripts** ausführen. Wählen Sie **FrameScript > Install**, und suchen Sie das zu installierende Skript.

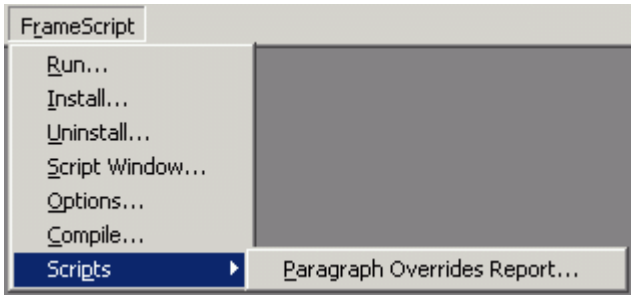


Das Dialogfeld **Standard Script Installation** wird angezeigt.



Wählen Sie einen Namen für das Skript. Wenn Sie im Feld **Script Name** nichts eingeben, vergibt FrameScript automatisch einen Namen (Script1, Script2, usw.). Das Feld **Menu Label** steht für den Namen, der im Untermenü **FrameScript > Scripts** angezeigt wird. Sie können auch einen Tastaturbefehl für das Skript auswählen und festlegen, unter welchen Bedingungen der Menübefehl verfügbar sein soll.

Sobald Sie auf **OK** klicken, ist Ihr Skript im Untermenü **FrameScript > Scripts** enthalten.



Falls Sie Ihre Skripts in einem externen Texteditor schreiben, können Sie die Skripts interaktiv bearbeiten und testen. Führen Sie hierfür unten stehende Schritte aus. Diese Vorgehensweise funktioniert nur bei Standardskripts, jedoch nicht bei Ereignisskripts.

1. Öffnen Sie Ihr Skript im Texteditor. Prüfen Sie, ob mindestens ein Befehl im Skriptkörper vorhanden ist.

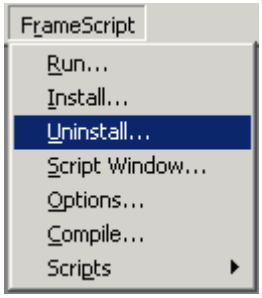
2. Lassen Sie das Skript im Texteditor geöffnet, und wechseln Sie zu FrameMaker. Wählen Sie **FrameScript > Install**, und wählen Sie das Skript aus, das im Texteditor geöffnet ist.

3. Wenn das Dialogfeld **Standard Script Installation** angezeigt wird, geben Sie ein **Menu Label** und bei Bedarf einen **Keyboard Shortcut** für das Skript ein.

Das Skript wird im Untermenü **FrameScript > Scripts** installiert.

Anschließend können Sie zwischen FrameMaker und dem Texteditor hin- und herwechseln. Wenn Sie Änderungen am Skript vornehmen, müssen Sie diese Änderungen im Texteditor speichern, bevor Sie das Skript wieder in FrameMaker ausführen. Wenn Sie das Skript über das Menü **FrameScript > Scripts** auswählen, wird stets die zuletzt gespeicherte Version des Skripts ausgeführt.

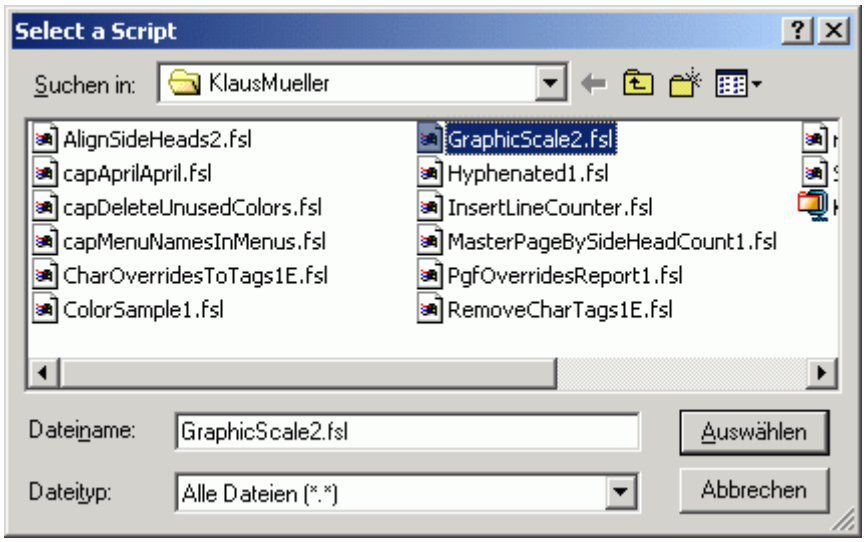
Zum Deinstallieren eines Skripts wählen Sie **FrameScript > Uninstall**.

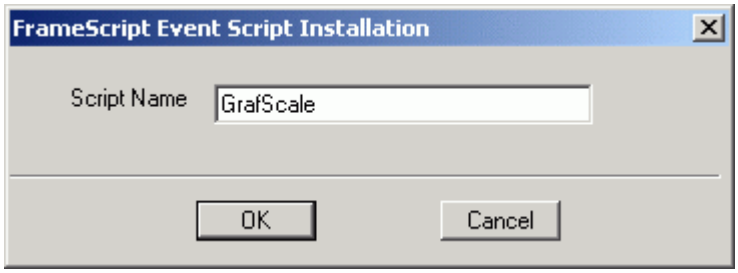


Wählen Sie das gewünschte Skript im Dialogfeld aus, und klicken Sie auf **OK**.

Ereignisskripts installieren und deinstallieren

Vor dem Testen eines Ereignisskripts müssen Sie dieses zunächst installieren. Zum installieren eines Ereignisskripts wählen Sie **FrameScript > Install**, und suchen Sie das gewünschte Skript. Sie werden in einem Dialogfeld aufgefordert, das Ereignisskript zu benennen.





Wird stattdessen das Dialogfeld **Standard Script Installation** angezeigt, haben Sie ein Standardskript statt eines Ereignisskripts gewählt. Wenn Sie sicher sind, dass Sie ein Ereignisskript ausgewählt haben, prüfen Sie das Skript, und stellen Sie sicher, dass keinerlei Code außerhalb von Ereignissen oder Subroutinen enthalten ist.

Wenn Sie Änderungen am Skript vornehmen, müssen Sie das Skript zunächst deinstallieren und anschließend erneut installieren. Zum Deinstallieren eines Skripts wählen Sie **FrameScript > Uninstall**. Wählen Sie das gewünschte Skript im Dialogfeld aus, und klicken Sie auf **OK**.

Mit FrameScript-Befehlen installieren und deinstallieren

FrameScript verfügt über Befehle zum Installieren und Deinstallieren von Skripts. Solche Befehle erleichtern zum Beispiel das Testen von Ereignisskripts. Sie können ein »Reset«-Skript erstellen, das ein Ereignisskript deinstalliert und wieder installiert. Wenn Sie Änderungen am Ereignisskript vornehmen, speichern Sie die Änderungen, und führen Sie anschließend das Reset-Skript aus, damit das Ereignisskript schnell deinstalliert und wieder installiert wird, sodass immer die neueste Version im Arbeitsspeicher geladen ist. Es folgt ein Beispiel, das in Verbindung mit dem weiter oben behandelten Skript SaveAsMif.fsl verwendet werden kann. Dabei gehen wir davon aus, dass sich das Skript im selben Ordner befindet wie die FrameScript .DLL.

Codebeispiel 1--12

```
// Reset-Skript für das Skript SaveAsMif.fsl.
// Skript deinstallieren.
Uninstall Script Name(SaveAsMif);
// Skript neu installieren.
Install Script File(ClientDir+DIRSEP+'SaveAsMif.fsl') Name('SaveAsMif');
```

Der Wert für den Parameter **Name** muss hierbei in beiden Befehlen identisch sein. Beim Parameter **File** im Befehl **Install Script** muss es sich um den Pfad zur Datei des Ereignisskripts handeln. In diesem Beispiel verwenden wir die FrameScript-Variable **ClientDir**, die den Pfad zur Datei FrameScript .DLL oder zum Modul zurückgibt. **DIRSEP** ist eine FrameScript-Variable, die das Trennzeichen für Ordner zurückgibt (Rückwärtsschrägstrich auf dem PC, Doppelpunkt auf dem Mac). Wenn das Skript nicht gefunden wird, meldet FrameScript einen Fehler beim Befehl **Install Script**. Dagegen gibt der Befehl **Uninstall Script** keinen Fehler aus, wenn Sie damit ein Skript deinstallieren, das gar nicht installiert ist.

Um das Reset-Skript zu verwenden, speichern Sie es, und installieren Sie es als Standardskript im Untermenü **FrameScript > Scripts**. Wenn Sie Änderungen am Ereignisskript SaveAsMif.fsl vornehmen, speichern Sie diese, und führen Sie das Reset-Skript aus. Die alte Version wird aus dem Arbeitsspeicher deinstalliert, anschließend wird die letzte gespeicherte Version installiert.

Skripts automatisch installieren

Mithilfe eines Initialisierungsskripts kann FrameScript Skripts automatisch bei jedem Start von FrameMaker installieren. Fügen Sie die Installationsbefehle in Ihr Standardskript ein, und definieren Sie dieses Skript als Initialisierungsskript. Siehe »Skripts automatisch ausführen« auf Seite 1-11.

Das folgende Skript installiert das Ereignisskript SaveAsMif.fsl und das zugehörige Reset-Skript. Wenn Sie dieses Skript speichern und als Initialisierungsskript definieren, wird das Skript SaveAsMif.fsl bei jedem Start von FrameMaker automatisch installiert. In diesem Beispiel muss das Initialisierungsskript im selben Ordner gespeichert sein wie die anderen Skripts.

Codebeispiel 1--13

```
// Initialisierungsskript zum Laden von SaveAsMif.
// Speicherort des Skripts ermitteln.
Find String(DIRSEP) InString(ThisScript) Backward ReturnPos(gPos);
Get String FromString(ThisScript) EndPos(gPos) NewVar(gScriptsPath);

// SaveAsMif-Skript installieren.
Install Script File(gScriptsPath+'SaveAsMif.fsl')
  Name('SaveAsMif');

// Reset-Standardskript installieren.
Install Script File(gScriptsPath+'ResetSaveAsMif.fsl')
  Name('ResetSaveAsMif') Label('Reset SaveAsMif Script');
```

Beachten Sie Folgendes bei diesem Skript:

- Der Befehl **Find String** verwendet die FrameScript-Variable **ThisScript**. Diese Variable gibt den vollständigen Pfad des Skripts zurück, das diesen Befehl enthält. So ermitteln wir den Pfad zu den anderen Skripts. Dies bedeutet, das Initialisierungsskript kann an einem beliebigen Speicherort installiert sein, sofern es sich im selben Ordner wie die anderen Skripts befindet.

- Die in diesem Skript erstellten Variablen erhalten das Präfix **g**, um sie als globale Variablen zu kennzeichnen. Dies weist darauf hin, dass die in einem Initialisierungsskript erstellten Variablen global gültig sind und im Arbeitsspeicher bleiben, so lange FrameMaker ausgeführt wird. Globale Variablen können nicht durch andere Skripts geändert werden. Zum Vermeiden von Konflikten sollten Sie die Namen der globalen Variablen nicht in anderen Skripts verwenden.

Strategien zum Schreiben von Skripts

Teilen Sie die Einzelaufgaben auf

Wenn Sie ein neues Skript schreiben möchten, werden Sie meistens zunächst an die Gesamtaufgabe des Skripts denken. Zum Beispiel können Sie folgende Gedanken haben: »Mein Skript soll alle verankerten Rahmen im Dokument zentriert ausrichten. Außerdem soll jeder Rahmen einen Rand mit Linienstärke 0,5 Punkt erhalten.« Der beste Ansatz liegt darin, die Gesamtaufgaben in kleinere Einzelschritte aufzuteilen. Dies wird erleichtert, wenn Sie eine Liste der Aufgaben erstellen.

- Verankerten Rahmen zentriert ausrichten

- Rand mit Linienstärke 0,5 Punkt hinzufügen

- Alle verankerten Rahmen des Dokuments durchlaufen

Sie sollten für jede Aufgabe je ein Skript schreiben. Wenn Sie sicher sind, dass jedes Einzelskript wie gewünscht funktioniert, können Sie die Skripts zu einem großen Skript zusammenfügen, das alle Aufgaben umfasst.

Beginnen Sie mit einzelnen Elementen

Wenn Sie verankerte Rahmen bearbeiten möchten, experimentieren Sie zunächst mit einem einzigen Rahmen. Der grundlegende Code ist immer derselbe, unabhängig davon, ob Sie einen oder tausend Rahmen bearbeiten, aber es ist definitiv sicherer, zunächst mit einem Rahmen zu beginnen! FrameMaker verfügt über Objekte, mit denen Sie markierte Elemente in Ihrem Dokument bearbeiten können. So gibt es zum Beispiel die Eigenschaft **FirstSelectedGraphicInDoc**, die ein markiertes Objekt im Dokument zurückgibt. Wenn Sie einen verankerten Rahmen markieren, können Sie dessen Eigenschaften über **FirstSelectedGraphicInDoc** bearbeiten.

Codebeispiel 1--14

```
// Variable für aktives Dokument setzen.
Set vCurrentDoc = ActiveDoc;

// Variable für die markierte Grafik setzen.
Set vAFrame = vCurrentDoc.FirstSelectedGraphicInDoc;

// Eigenschaften des verankerten Rahmens anzeigen.
Display vAFrame.Properties;
```

Verwenden Sie die Dokumentation

Nun, da Sie ein Objekt für einen verankerten Rahmen besitzen, können Sie mit dessen Eigenschaften experimentieren. In diesem Beispiel schlagen Sie das Objekt **AFrame** in der *FrameScript Scriptwriter's Reference* nach. Wenn Sie nicht sicher sind, wie ein bestimmtes Objekt heißt, können Sie mit folgendem Code den **ObjectName** anzeigen.

Michaels Tipp Die Lesezeichen in der *FrameScript Scriptwriter's Reference* (fslref.pdf) sind nicht optimal konfiguriert. Schieben Sie alle Einträge von »Command Format« (der zweite Eintrag) bis einschließlich »Write Command« unter »Command Reference« (und dann zuklappen) und alle Einträge von »General Information« bis einschließlich »Variables« unter »Object Reference«. So wird das Dokument schon viel übersichtlicher.

Codebeispiel 1--15

```
// Variable für aktives Dokument setzen.
Set vCurrentDoc = ActiveDoc;

// Variable für die markierte Grafik setzen.
Set vAFrame = vCurrentDoc.FirstSelectedGraphicInDoc;

// ObjectName-Eigenschaft für den verankerten Rahmen anzeigen.
Display vAFrame.ObjectName;
```

In Kapitel 2 der *FrameScript Scriptwriter's Reference* sind die Eigenschaften für alle Arten von FrameMaker-Objekten beschrieben. Wenn Sie unter **AFrame** nachsehen, finden Sie die Eigenschaft **Alignment** sowie eine Liste der möglichen Werte. Versuchen Sie dies nun mit dem markierten verankerten Rahmen.

Michaels Tipp Probieren Sie aus, ob es Ihnen nicht leichter fällt, die Objekteigenschaften mit dem *FOS Browser* zu analysieren, der von mir auf http://elmscript.de zum Download angeboten wird.

Codebeispiel 1--16

```
// Variable für aktives Dokument setzen.
Set vCurrentDoc = ActiveDoc;

// Variable für die markierte Grafik setzen.
Set vAFrame = vCurrentDoc.FirstSelectedGraphicInDoc;

// Verankerten Rahmen zentriert ausrichten.
Set vAFrame.Alignment = AlignCenter;
```

Der markierte Rahmen sollte nun zentriert ausgerichtet sein. Suchen Sie nun die Eigenschaft, die einen Rand um den Rahmen definiert. Tatsächlich benötigen Sie hierfür zwei Eigenschaften: **BorderWidth** stellt die Linienstärke ein, und **Pen** legt den Linienstil fest. Fügen Sie den Code in Ihr Skript ein, um den Rand zu bearbeiten.

Codebeispiel 1--17

```
// Variable für aktives Dokument setzen.
Set vCurrentDoc = ActiveDoc;

// Variable für die markierte Grafik setzen.
Set vAFrame = vCurrentDoc.FirstSelectedGraphicInDoc;

// Verankerten Rahmen zentriert ausrichten.
Set vAFrame.Alignment = AlignCenter;

// Schwarzen Rand um den verankerten Rahmen ziehen.
Set vAFrame.BorderWidth = .5pt;
Set vAFrame.Pen = FillBlack;
```

Speichern Sie Ihre Einzelschritte

Sie sollten jede Einzelkomponente Ihres Skripts speichern, sobald diese wie gewünscht funktioniert. Vielleicht kommt es Ihnen albern vor, so kleine Skriptabschnitte zu speichern, aber so haben Sie bei großen Skripts stets einen Punkt, an den Sie zurückkehren können, falls unerwartete Probleme auftreten. Mit zunehmender Erfahrung mit FrameScript werden Sie weniger häufig speichern.

Fahren Sie mit dem nächsten Schritt in der Liste fort

Nun können Sie mit dem nächsten Schritt in Ihrer Liste fortfahren. Während der Lernphase sollten Sie jeden Schritt in einem separaten Skript entwickeln. In unserem Beispiel sollen alle verankerten Rahmen im Dokument durchlaufen werden. Schlagen Sie das Objekt **Doc** (Dokument) in der *FrameScript Scriptwriter's Reference* nach, um eine geeignete Eigenschaft zu finden. Siehe »Objekte und Eigenschaften« und »Steuerstrukturen«.

Es folgt ein einfaches Skript, das alle Grafiken im Dokument durchläuft.

Codebeispiel 1--18

```
// Variable für aktives Dokument setzen.
Set vCurrentDoc = ActiveDoc;

// Variable für erste Grafik im Dokument setzen.
Set vGraphic = vCurrentDoc.FirstGraphicInDoc;
// Alle Grafiken im Dokument durchlaufen.
Loop While(vGraphic)
  // Grafik in die Konsole ausgeben.
  Write Console vGraphic;
  // Zur nächsten Grafik wechseln.
  Set vGraphic = vGraphic.NextGraphicInDoc;
EndLoop
```

Jedes Grafikobjekt wird in die FrameMaker-Konsole geschrieben. Auf diese Weise können Sie den Code testen, bevor Sie tatsächlich Aktionen ausführen. Das Skript soll nur verankerte Rahmen verarbeiten, also müssen Sie innerhalb der Schleife eine Prüfung einfügen.

Codebeispiel 1--19

```
// Variable für aktives Dokument setzen.
Set vCurrentDoc = ActiveDoc;

// Variable für erste Grafik im Dokument setzen.
Set vGraphic = vCurrentDoc.FirstGraphicInDoc;
// Alle Grafiken im Dokument durchlaufen.
Loop While(vGraphic)
  // Prüfen, ob die Grafik ein verankerter Rahmen ist.
  If (vGraphic.ObjectName = 'AFrame')
    // Grafik in die Konsole ausgeben.
    Write Console vGraphic;
  EndIf
  // Zur nächsten Grafik wechseln.
  Set vGraphic = vGraphic.NextGraphicInDoc;
EndLoop
```

Erstellen Sie eine Bibliothek mit wiederverwendbarem Code

Wenn Sie dieses Skript ausführen, werden nur verankerte Rahmen (**AFrame**) in die Konsole geschrieben. Diese Aufgabe ist also abgeschlossen, und Sie können den Code als separates Skript speichern. Speichern Sie es unter einem aussagekräftigen Namen, zum Beispiel AlleVerankRahmenInDokVerarb.fsl. Wenn Sie ein weiteres Skript zum Verarbeiten aller verankerten Rahmen in einem Dokument benötigen, können Sie dieses Skript aus Ihrer »Codebibliothek« als Ausgangspunkt verwenden.

Setzen Sie den Code zusammen

Nun können Sie die beiden Skripts zu einem einzigen zusammensetzen, damit alle Aufgaben ausgeführt werden. Erstellen Sie ein Beispieldokument mit einigen verankerten Rahmen, an dem Sie Ihr Skript gefahrlos testen können. In unserem Beispiel sollen alle verankerten Rahmen im Dokument bearbeitet werden, also müssen Teile des ersten Skripts in die Schleife des zweiten Skripts eingefügt werden. Behalten Sie

unbedingt die beiden einzelnen Skripts, und erstellen Sie für die Kombination aus beiden ein drittes Skript. Unten finden Sie das kombinierte Skript. Sehen Sie sich das Skript an, und versuchen Sie, die Komponenten der Einzelskripte zu identifizieren. Sie werden bemerken, dass geringfügige Änderungen an den Variablennamen vorgenommen wurden, damit beide Codes miteinander funktionieren.

Codebeispiel 1--20

```
// Variable für aktives Dokument setzen.
Set vCurrentDoc = ActiveDoc;

// Variable für erste Grafik im Dokument setzen.
Set vGraphic = vCurrentDoc.FirstGraphicInDoc;
// Alle Grafiken im Dokument durchlaufen.
Loop While(vGraphic)
  // Prüfen, ob die Grafik ein verankerter Rahmen ist.
  If (vGraphic.ObjectName = 'AFrame')
    // Verankerten Rahmen zentriert ausrichten.
    Set vGraphic.Alignment = AlignCenter;
    // Schwarzen Rand um den verankerten Rahmen ziehen.
    Set vGraphic.BorderWidth = .5pt;
    Set vGraphic.Pen = FillBlack;
  EndIf
  // Zur nächsten Grafik wechseln.
  Set vGraphic = vGraphic.NextGraphicInDoc;
EndLoop
```

Testen und optimieren Sie das Skript

Während der Entwicklung von Skripts werden Sie auf neue Aufgaben stoßen, die erledigt werden müssen. In unserem Beispiel liegen zwei Aufgaben auf der Hand.

- Es sollen nur verankerte Rahmen bearbeitet werden, die unterhalb der aktuellen Zeile verankert sind. Mit anderen Worten: Verankerte Rahmen, die direkt an der Einfügemarke, außerhalb der Spalte usw. verankert sind, sollen übersprungen werden.

- Außerdem soll vor dem Ausführen des Skripts geprüft werden, ob ein aktives Dokument vorhanden ist.

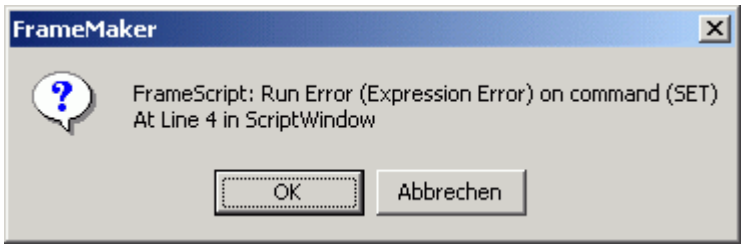
Möglicherweise werden Sie die Bedeutung der ersten Aufgabe recht leidvoll erfahren: Sie führen das Skript aus und stellen fest, dass es nicht ausreichend genau festlegt, welche Elemente bearbeitet werden sollen. Versuchen Sie vorherzusehen, zu welchen Ergebnissen die Befehle in Ihrem Skript unter den verschiedensten Bedingungen führen. Im Allgemeinen sollten Sie den Wirkungsbereich der Befehle so eng wie möglich eingrenzen. Für unser Beispielskript können Sie dies folgendermaßen umsetzen:

Codebeispiel 1--21

```
// Variable für aktives Dokument setzen.
Set vCurrentDoc = ActiveDoc;

// Variable für erste Grafik im Dokument setzen.
Set vGraphic = vCurrentDoc.FirstGraphicInDoc;
// Alle Grafiken im Dokument durchlaufen.
Loop While(vGraphic)
  // Prüfen, ob die Grafik ein verankerter Rahmen ist.
  If (vGraphic.ObjectName = 'AFrame')
    // Prüfen, ob der Rahmen unterhalb der aktuellen Zeile verankert ist.
    If (vGraphic.AnchorType = 'AnchorBelow')
      // Verankerten Rahmen zentriert ausrichten.
      Set vGraphic.Alignment = AlignCenter;
      // Schwarzen Rand um den verankerten Rahmen ziehen.
      Set vGraphic.BorderWidth = .5pt;
      Set vGraphic.Pen = FillBlack;
    EndIf
  EndIf
  // Zur nächsten Grafik wechseln.
  Set vGraphic = vGraphic.NextGraphicInDoc;
EndLoop
```

Auf die zweite Aufgabe stoßen Sie, wenn Sie das Skript ausführen, ohne dass ein Dokument geöffnet ist. Sie erhalten folgende Fehlermeldung:



Ihr Skript sollte *Fehlerbehandlungsroutinen* enthalten, um Bedingungen abzufangen, die zu einem Fehler führen könnten. Wenn Ihr Code eine bestimmte Bedingung erfordert, müssen Sie in Ihrem Skript prüfen, ob diese Bedingung auch erfüllt ist. Mit folgendem Code wird geprüft, ob ein Dokument geöffnet ist:

Codebeispiel 1--22

```
// Prüfen, ob aktives Dokument vorhanden.
If (ActiveDoc = 0)
  MsgBox 'Kein aktives Dokument. ';
  LeaveSub; // Skript beenden.
Else
  // Variable für aktives Dokument setzen.
  Set vCurrentDoc = ActiveDoc;
EndIf
```

Fehlerbehebung in Skripts

Wenn FrameScript einen Fehler in einem Skript feststellt, wird normalerweise die Nummer der Zeile ausgegeben, die den Fehler verursacht. Nun können Sie diese Zeile überprüfen, um den Fehler zu finden. Deshalb ist es hilfreich, einen Texteditor zu verwenden, der die Zeilennummern anzeigt. Wenn Sie das Skriptfenster verwenden, wird links unten die Nummer der Zeile angezeigt, in der sich die Einfügemarke befindet. Sie können mit **Search > Go To Line** (Strg+L) zu einer bestimmten Zeile im Skriptfenster wechseln.

Viele Skriptfehler werden von falsch geschriebenem Code verursacht. Vergessen Sie nicht, dass Programmiersprachen sehr präzise sind. Ein einziger falscher Buchstabe kann zu einem Fehler führen. Andere Fehler können dadurch entstehen, dass für bestimmte Eigenschaften die falschen Datentypen verwendet werden. Wenn Sie zum Beispiel eine Zeichenfolge verwenden und diese nicht in Anführungszeichen setzen, führt dies zu einem Fehler.

Beim Ausführen eines Skripts kann es zu Kompilierungsfehlern kommen. Diese werden normalerweise durch falsch angeordnete Steueranweisungen verursacht, zum Beispiel eine **If**-Anweisung ohne eine zugehörige **EndIf**-Anweisung. Möglicherweise wird bei Kompilierungsfehlern nicht exakt die Zeile mit dem Fehler gemeldet, aber in der Regel wird der Fehler innerhalb weniger Zeilen vor der in der Fehlermeldung angegebenen Zeile verursacht.

Eine besondere Art von Fehler ist ein so genanntes »Runaway-Skript«. Dieser Fehler wird durch Endlosschleifen verursacht. Bei einem Runaway-Skript wird das Skript erst beendet, wenn FrameMaker abstürzt. Sie können versuchen, ein Runaway-Skript zu beenden, indem Sie die Esc-Taste drücken. Das folgende Beispiel ist eine Schleife, die zu einem Runaway-Skript führt:

```
// Diesen Code nicht ausführen!
Set vCurrentDoc = ActiveDoc;

// Variable für erste Grafik im Dokument setzen.
Set vGraphic = vCurrentDoc.FirstGraphicInDoc;
// Grafiken durchlaufen.
Loop While(vGraphic)
  // Grafik in die Konsole ausgeben.
  Write Console vGraphic;
  // Hier fehlt etwas!
EndLoop
```

Innerhalb der Schleife ist kein Befehl vorhanden, zur nächsten Grafik im Dokument zu wechseln, also endet die Schleife niemals. Dies ist die fehlende Zeile:

```
Set vGraphic = vGraphic.NextGraphicInDoc;
```

Weitere Informationen über Schleifen finden Sie unter »Loop/EndLoop« auf Seite 3-3.