

KVS Einsteigertutorial

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
1.0	Jan 08		T

Contents

1	Voraussetzungen	1
2	Einleitung	1
2.1	Zu diesem Tutorial	1
2.2	Über KVIrc	1
2.3	Über KVIrc-Skripte (KVS)	1
3	Der Anfang	1
3.1	Befehle in der Eingabezeile	2
3.2	Aliasse	2
3.2.1	Was ist ein Alias	2
3.2.2	Aliaseditor	2
3.2.3	Ein Alias aufrufen	3
3.3	Ereignisse	3
3.3.1	Wozu Ereignisse?	3
3.3.2	Der Ereigniseditor	4
4	Grundlagen	4
4.1	Variablen	4
4.2	Befehle	6
4.3	Funktionen	6
4.4	Kommentare	7
4.5	Alias im Detail	7
4.5.1	Parameter	8
4.5.2	Rückgabewert	9
4.6	Ereignisse	10
4.6.1	Parameter	11
4.6.2	Neue Ereignisskripte anlegen	12
4.6.3	Standartereignisse unterdrücken	13
4.7	Rechnen mit Zahlen	13
4.8	Arbeiten mit Strings	14
5	Schluss	15
5.1	Dankansagungen	15
5.2	Kontakt	15
5.3	Copyright	15

1 Voraussetzungen

- Computer mit Betriebssystem ;-)
- Ein installiertes KVIrc (Version möglichst aktuell)
- Genug Zeit

2 Einleitung

2.1 Zu diesem Tutorial

Hallo!

Vielen Dank erstmal, dass du dich für dieses Tutorial entschieden hast. Dieses Tutorial ist entstanden, da ich selbst gerne KVIrc-Skripte schreibe und mir aufgefallen ist, dass es eigentlich nicht genug Einleitungen und Tutorials für Anfänger gibt. Es ist hauptsächlich an Anfänger gerichtet, die auch vom Programmieren noch keine Ahnung haben. Leute, die schon andere Programmiersprachen können, werden merken, dass sie einiges überspringen können.

Ich wünsche allen nun viel Spaß beim Weiterlesen und Skripten.

2.2 Über KVIrc

"After 7 years of development KVIrc is a mature and full featured IRC client, an excellent companion for your daily IRC sessions."

"KVIrc is a free portable Internet Relay Chat client based on the excellent Qt GUI toolkit. KVIrc is being written by Szymon Stefanek and the KVIrc Development Team with the contribution of many IRC addicted developers around the world."

— Pragma <http://www.kvirc.net/>

2.3 Über KVIrc-Skripte (KVS)

Was sind Skripte? Wozu brauchen wir sie?

Skripte sind einfach eine Reihe von Befehlen, die als Text geschrieben werden und dann wenn sie ausgeführt werden, vom Programm gelesen werden.

Skripte sind einfach dazu da, sich das Bedienen des Programmes einfacher zu machen, häufige Arbeiten zu automatisieren oder es auch um neue Funktionen zu erweitern.

KVS bietet eine Menge an Befehlen, mit denen man fast alles machen kann. Vom kleinen Mini-Skript bis hin zu einer komplett neuer Oberfläche.

3 Der Anfang

Es gibt mehrere Möglichkeiten KVIrc-Skripte auszuführen oder zu ändern. Ich erkläre euch nun kurz die 3 Hauptmethoden mit denen man Skripte aufruft und bearbeitet.

3.1 Befehle in der Eingabezeile

Die Eingabezeile hat bestimmt jeder von euch schon gesehen. Sie sieht so aus:



Doch wo ist hier die Verbindung zu KVS?

Ganz einfach: Normalerweise wird alles, was mit einem "/" anfängt zum Server geschickt. Davor prüft KVirc allerdings noch, ob die Eingabe KVS Befehle enthält.

Hier möchte ich zu einem kleinen Beispiel kommen. Wir schauen uns nun einmal den Befehl "echo" an. Der Befehl dient einfach dazu, Text im aktuellen Fenster anzuzeigen und wird deshalb nicht an den Server geschickt.

Also gehen wir dazu in die Eingabezeile und geben folgendes ein:

```
/echo "Hallo Welt!"
```

und drücken Anschließend [ENTER]. Es folgt folgende Ausgabe:



D.h. wir können einfach in irgendeinem Fenster KVS Befehle ausführen, indem wir die Eingabe mit "/" beginnen. Doch es ist einem Leid wenn man ein langes Skript hat alles immer so in die Eingabezeile eintippen muss. Dies bringt uns zu einer sehr praktischen Funktion in KVirc nämlich den Aliassen, die später im Tutorial noch genauer beschrieben werden.

3.2 Aliasse

3.2.1 Was ist ein Alias

Ein Alias enthält eine Reihe von KVS-Befehlen. In der Eingabezeile lassen sich zwar auch Befehle ausführen, allerdings wird das dann bei mehreren oder längeren Befehlen sehr aufwändig sie immer von Hand einzugeben.

3.2.2 Aliaseditor

Nur um kurz zu zeigen wie er aussieht:



Hier haben wir nun 2 Befehle, die wir auf einmal ausführen wollen.

Ich habe das Alias mal "hallowelt" genannt.

3.2.3 Ein Alias aufrufen

Ich habe ja nun im Beispiel ein Alias erstellt und das kann man nun folgendermaßen aufrufen:

```
/hallowelt
```

Die Ausgabe wäre dann:



Doch da die Aliasse ein ganzes eigenes Kapitel sind, werde ich sie später noch weiter erläutern und auch beschreiben wie man sie Schritt für Schritt erstellt.

3.3 Ereignisse

3.3.1 Wozu Ereignisse?

Aliase sind schon besser als alles von Hand einzutippen, doch man kann folgendes mit ihnen nicht machen:

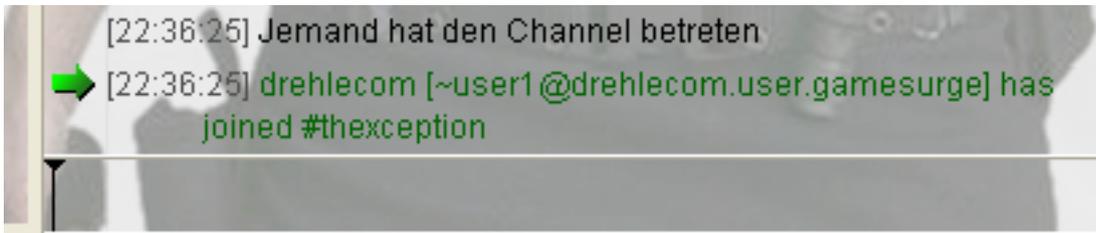
- Wie lasse ich z.B. ein Skript ausführen, wenn jemand in einen Channel kommt?
- Was soll KVIrc machen, wenn ...?

Die Lösung dafür sind Ereignisse. Es gibt von KVIrc vorgegebene Ereignisse, denen man Skripte zuweisen kann, die dann in diesem Fall ausgeführt werden.

3.3.2 Der Ereigniseditor



Ich habe hier mal das Ereignis gewählt, dass ausgelöst wird, wenn jemand einen Channel betritt (OnJoin, eine vollständige Beschreibung der Ereignisse findet ihr in der Hilfe.) Das Ganze ergibt dann folgende Ausgabe:



Zu beachten ist hier, dass das Skript ausgeführt wird, bevor die normale Betreten-Nachricht angezeigt wird. Dies ermöglicht einfach, dass man im Skript die original KVIrc Ausgabe unterdrücken kann und dann nur noch die eigene Ausgabe erscheint.

4 Grundlagen

4.1 Variablen

Variablen sind dazu da, dass man in ihnen Inhalte, wie Zahlen oder Texte, speichern kann. Schließlich wollen wir uns ja bestimmte Dinge, während eines Skriptdurchlaufs oder im ganzen Programm, merken. Variablen beginnen in KVIrc immer mit dem Zeichen "%".

KVIrc unterscheidet zwischen 2 Typen von Variablen:

- globale Variablen
- lokale Variablen

Globale Variablen können von überall im Programm aus erreicht werden. Ihre Namen beginnen mit einem Großbuchstaben. Globale Variablen bleiben solange gespeichert bis KVIrc beendet wird.

Lokale Variablen gelten nur so lange, bis das aktuelle Skript abgearbeitet wurde und werden danach geleert. Ihre Namen beginnen mit Kleinbuchstaben.

Es gibt in KVirc folgende Variablentypen:

Table 1: Variablentypen

Typ	Beschreibung	Beispiel
string	Enthält einen Text	<code>%var = "Hallo"</code>
integer	Eine Ganzzahl	<code>%var = 1</code>
real	Eine Zahl mit Nachkommastellen	<code>%var = 2.1987</code>
boolean	wahr/falsch true/false 1/0	<code>%var = \$true</code>
hobject	siehe Dokumentation	
arrays	siehe Dokumentation	
hashes	siehe Dokumentation	

Beispiele:

Globale Variablen

```
%Variable
%Tralla
```

Lokale Variablen

```
%variable
%nocheine
```

Variablenamen dürfen nur aus Buchstaben (A-Z und a-z) sowie "_" und "." bestehen.

Kommen wir dazu, wie wir etwas in eine Variable speichern. Man kann etwas in eine Variable speichern, in dem wir ihr einen bestimmten Wert/Text zuweisen. Dies geschieht mit Hilfe eines "=".

```
%Text = "Hallo Welt"
```

Um das ganze auszuprobieren, geben wir das ganze mal in die Eingabezeile ein:

```
/%Text = "Hallo Welt"
```

Nichts passiert. Haben wir etwas falsch gemacht? Nein :)

Es wird nur keine Ausgabe erzeugt. Doch wir haben ja unseren Befehl "echo" kennengelernt.

Also tippen wir:

```
/echo %Text
```

Und siehe da!



Unser Hallo Welt-Text, den wir vorher in der Variable %Text gespeichert haben, wird angezeigt.

Du kannst das ganze auch mit anderen Texten probieren.

Als Variation kannst du nun mal statt %Text einfach %text nehmen. Du wirst bemerken, dass, wenn du %text nimmst, er nichts anzeigt. Das liegt einfach daran, dass %text mit einem Kleinbuchstaben anfängt und damit eine lokale Variable ist. Diese gilt damit, in diesem Fall, nur für den Teil des Skriptes bis man mit [ENTER] bestätigt und wird danach wieder geleert.

Doch man kann nicht nur Text in Variablen speichern sondern auch Zahlen. Zahlen werden dann nicht in Anführungszeichen geschrieben. Probieren wir es mit:

```
%MeineZahl = 8
```

und danach

```
echo %MeineZahl
```

Hinweis

Beachte immer den Unterschied Globale/Lokale Variable am ersten Buchstaben des Namens.

Achte auch darauf, dass wenn du ein Befehl/Skript in die Eingabezeile eingibst, du mit einem "/" anfängst.

4.2 Befehle

Einen Befehl haben wir im Laufe dieses Tutorials schon kennengelernt "echo". Ein Befehl ist immer wie folgt aufgebaut:

```
befehl -switch1 -switch2 parameter1 parameter2 parameter3
```

Damit ihr auch gleich einmal was von der Hilfe aus KVIrc mitbekommt, dort steht zu echo folgendes:

```
Syntax Specification  
echo [-d] [-w=<window_id>] [-i=<icon_number>] [-n] <text>
```

[] stehen in der Documentation als optionale Parameter. D.h., sie müssen nicht verwendet werden.

< > stehen in der Documentation als benötigte Parameter, diese müssen auf jeden Fall berücksichtigt werden, sonst bekommen wir eine Fehlermeldung.

4.3 Funktionen

Funktionen sind sehr ähnlich zu Befehlen. Sie haben jedoch einen wichtigen Unterschied: Funktionen haben einen Rückgabewert.

```
%rueckgabe = $funktion(parameter1, parameter2, parameter3)
```

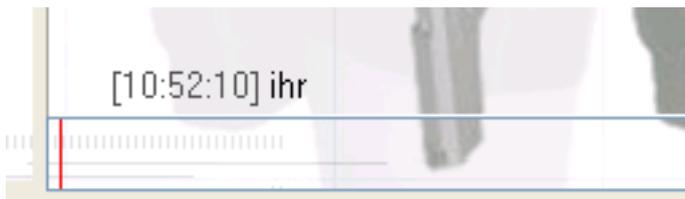
Während wir mit "echo" nur etwas anzeigen können, haben wir nun die Möglichkeit, dass das was wir aufrufen uns etwas zurückgibt. Auch hier wieder ein Stück aus der Dokumentation:

```
<string> $str.mid(<data>, <startidx>, <nchars>)
```

"\$str.mid" gibt uns einen Teil eines Strings zurück und zwar aus dem String "data" ab Position "startidx". Von dort ab werden die nächsten "nchars" Zeichen gelesen. Betrachtet hierzu folgendes Beispiel:

```
echo $str.mid("Hallo ihr da!", 6, 3)
```

Dies erzeugt folgende Ausgabe:



Im Ganzen betrachtet heißt das: Wir übergeben der Funktion mehrere Parameter und sie gibt uns dann wie z.B. hier das Wort "ihr" zurück.

4.4 Kommentare

Ohne Kommentare würde ein längeres Skript einfach nur unübersichtlich werden oder wenn man sich ein Skript nach einer Weile wieder anschaut, müsste man sich jedesmal immer wieder ganz genau den Code anschauen wo was war. KVIrc bietet hier die Möglichkeit Kommentare in ein Skript zu schreiben die nicht mit als Skript ausgewertet werden. Kommentiert eure Skripte so gut wie möglich, denn wenn ihr später Änderungen vornehmen müsst wird euch das eure Arbeit erleichtert.

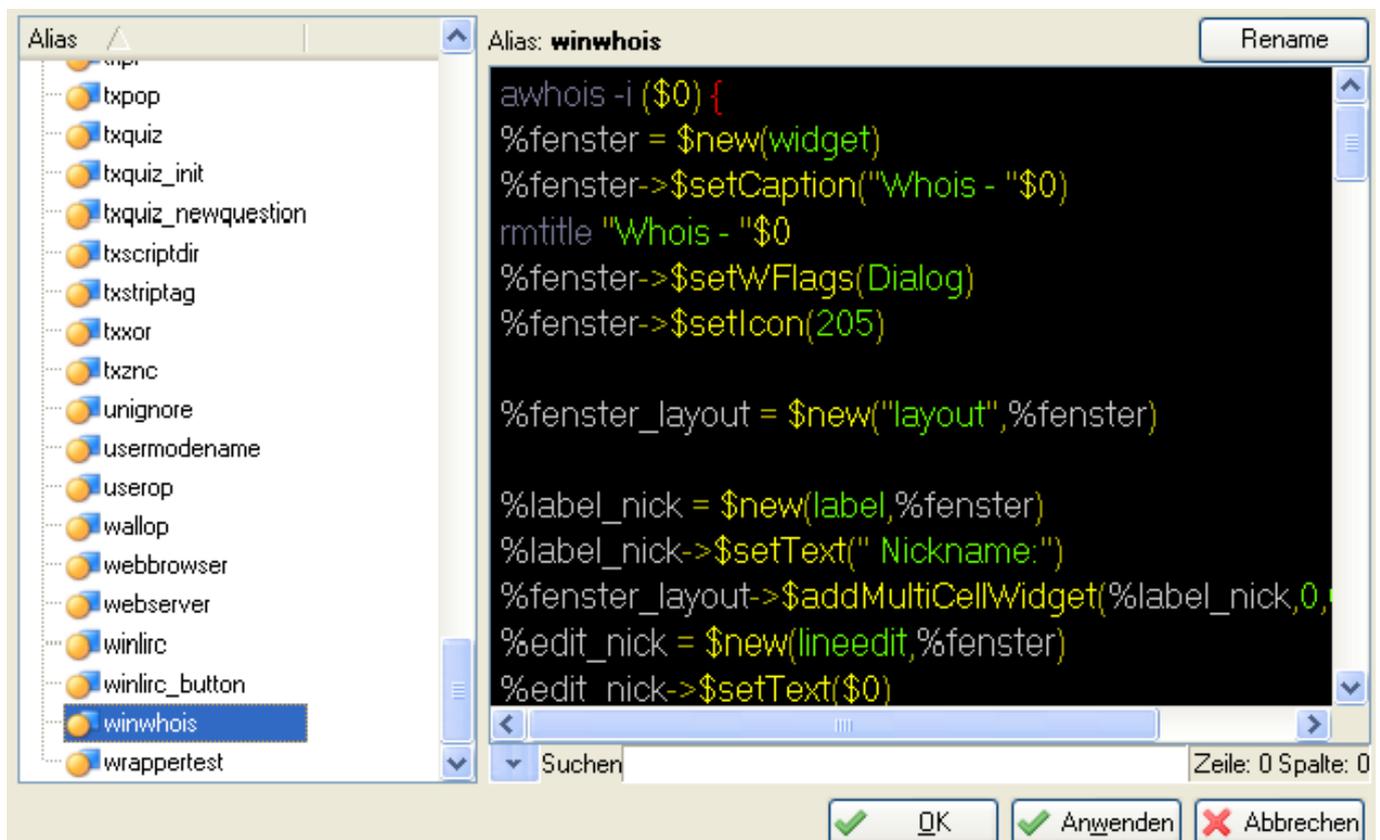
Kommentare können verschieden sein:

```
#Ein einzeliger Kommentar

//Noch ein einzeliger Kommentar

/* Ein
mehrzeiliger
Kommentar */
```

4.5 Alias im Detail

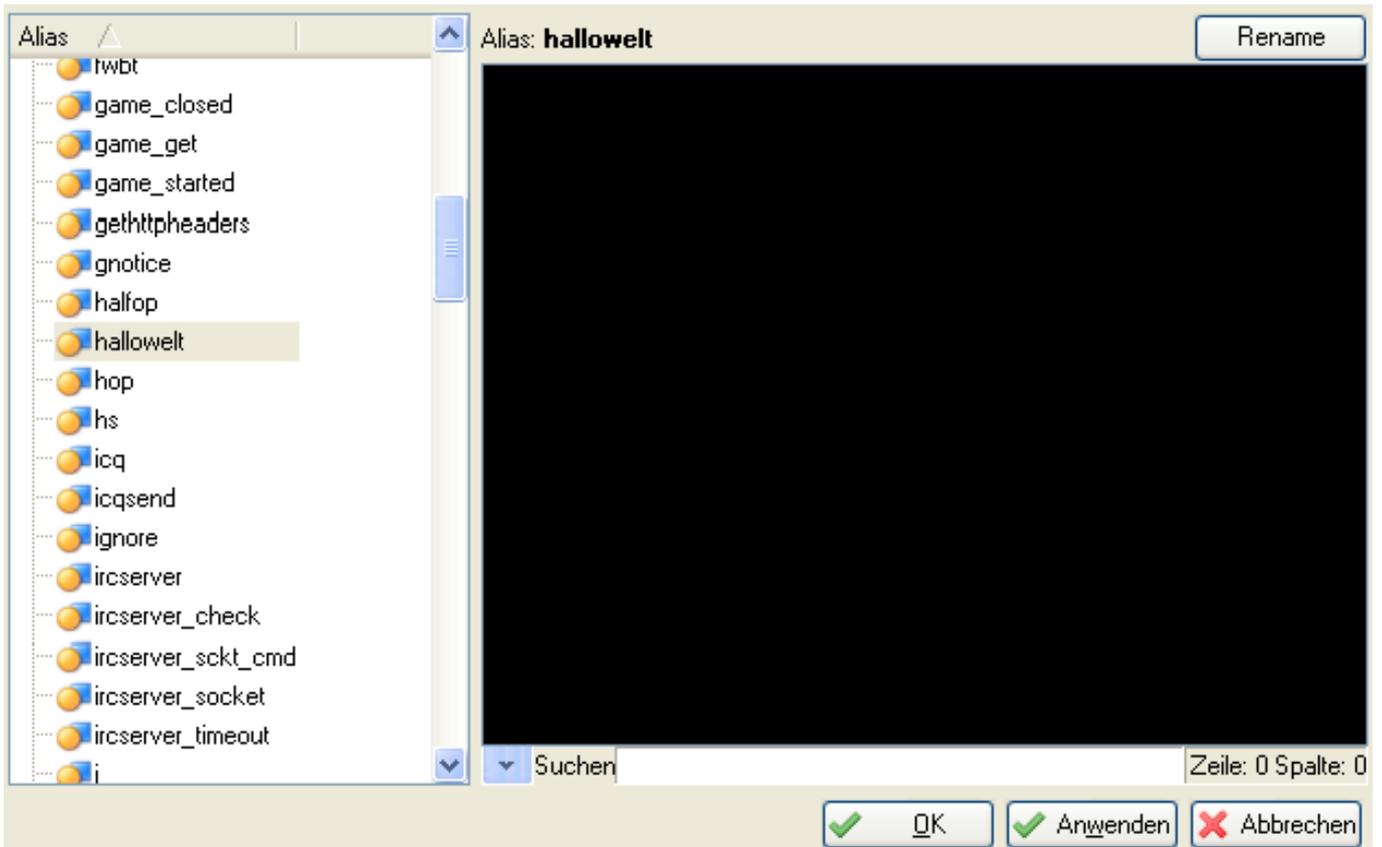


Hier noch mal ein Bild vom Aliaseditor. Ihr könnt ihn aufrufen indem ihr im Menü "Skripte" → "Aliasse bearbeiten" auswählt oder einfach "STRG+ALT+2" drückt. Nun seht ihr links eine Liste aller schon vorhandenen Aliasse. KVIrc bringt teilweise schon welche mit. Man kann nun links ein schon vorhandenes Alias auswählen oder ein neues anlegen. Wir beschäftigen uns gleich damit ein neues Alias anzulegen. Kurz vorweg: Mit "Rename" kann man das gerade ausgewählte Alias umbenennen. Änderungen werden erst übernommen wenn man auf "Anwenden" klickt. "OK" schließt das Fenster und speichert das Skript. "Abbrechen" bricht das Ändern ab.

Um kurz Klarheiten zu schaffen: Ein Alias mit Rückgabewert wird auch Funktion genannt. Ein Alias ohne Rückgabewert auch Befehl.

Also beginnen wir erstmal damit ein neues Alias anzulegen. Hierzu macht ihr einen Rechtsklick in die Liste der Aliasse im Aliaseditor, wählt "Add Alias / Alias hinzufügen" und gebt den Namen ein, den ihr dem Alias geben wollt. Am besten nur Buchstaben ohne Leer und Trennzeichen.

Ich habe mich für "hallowelt" entschieden. Ihr solltet nun folgendes vor euch sehen:



Wie ihr seht haben wir nun rechts die Möglichkeit unseren Code einzugeben. Gehen wir unserem Anfangsbeispiel nach:



Vergesst aber nicht auf Anwenden zu klicken, erst dann wird das Skript übernommen. Wechselt nun am besten zu einem beliebigen Fenster das eine Eingabezeile besitzt und gebt "/" und dann euren Aliasname ein. Wenn ihr alles richtig gemacht habt seht ihr folgendes:



4.5.1 Parameter

Das Alias funktioniert schon recht gut, doch Aliase bieten noch mehr Möglichkeiten. Parameter übergibt man an ein Alias in dem man sie beim Aufrufen des Alias einfach nachfolgend übergibt.

```
/aliasname Parameter1 Parameter2 .....
```

Die Verarbeitung der Parameter macht KVirc automatisch. Abgefragt werden können sie mit \$ und dann der Nummer des Parameters:

\$0	#Erster Parameter
\$1	#Zweiter Parameter
\$0-	#Alle Parameter
\$1-	#Den zweiten Parameter und alle darauf folgenden
\$4-6	#Den 5. bis zum 7. Parameter

Nun ändern wir unser Beispiel folgendermaßen:



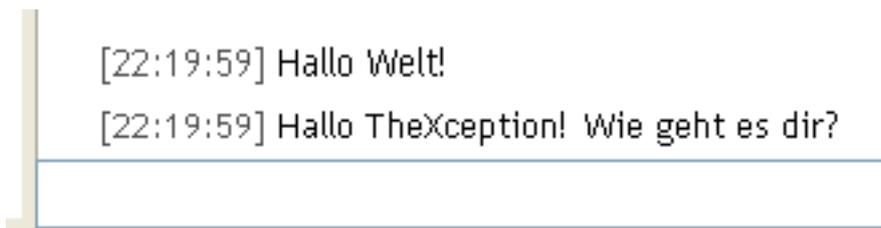
```
Alias: hallowelt
echo "Hallo Welt!"
echo "Hallo $0-! Wie geht es dir?"
```

Um kurz zu erklären wie es funktioniert:

\$0- steht für alle Parameter. An der Stelle an der das \$0- steht setzt KVS die Parameter ein. Nun können wir unsere Funktion wieder aufrufen:

```
/hallowelt TheXception
```

Die dazugehörige Ausgabe:



```
[22:19:59] Hallo Welt!
[22:19:59] Hallo TheXception! Wie geht es dir?
```

4.5.2 Rückgabewert

Machen wir eine kleine Überlegung: Wir wollen ein eigenes Alias aufrufen und eine Rückgabe erhalten.

Um einen Rückgabewert aus einem Alias zu erhalten, gibt es in KVirc einen Befehl namens "return". Dieser braucht als Parameter das, was die Funktion zurückgeben soll. z.B.:

```
return 78
return "Hallo"
```

Lassen wir unser Alias doch einfach mal die Zahl 99 zurückgeben:



```
Alias: hallowelt  
echo "Hallo Welt!"  
echo "Hallo $0-! Wie geht es dir?"  
return 99
```

Doch damit ist es noch nicht getan. Das Alias muss jetzt auch anders aufgerufen werden.

```
$aliasname (Parameter1, Parameter2, ...)
```

Das Ganze hat nun die folgenden Vorteile:

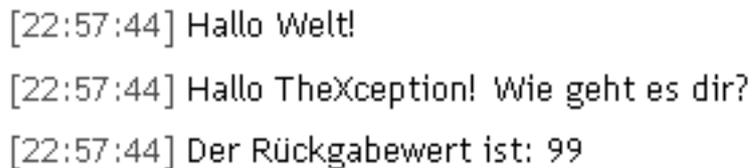
Wenn wir ein Alias nach der bisherigen Methode aufrufen würden werden die Parameter nicht mehr bearbeitet da sie dann für einen normalen Text gehalten werden. Lasst mich hierzu ein Beispiel zeigen:

```
/echo hallowelt TheXception
```

Es würde einfach nur "hallowelt TheXception" als Ausgabe erscheinen. Wenn wir nun die neue Methode nehmen:

```
/echo "Der Rückgabewert ist: "$hallowelt(TheXception)
```

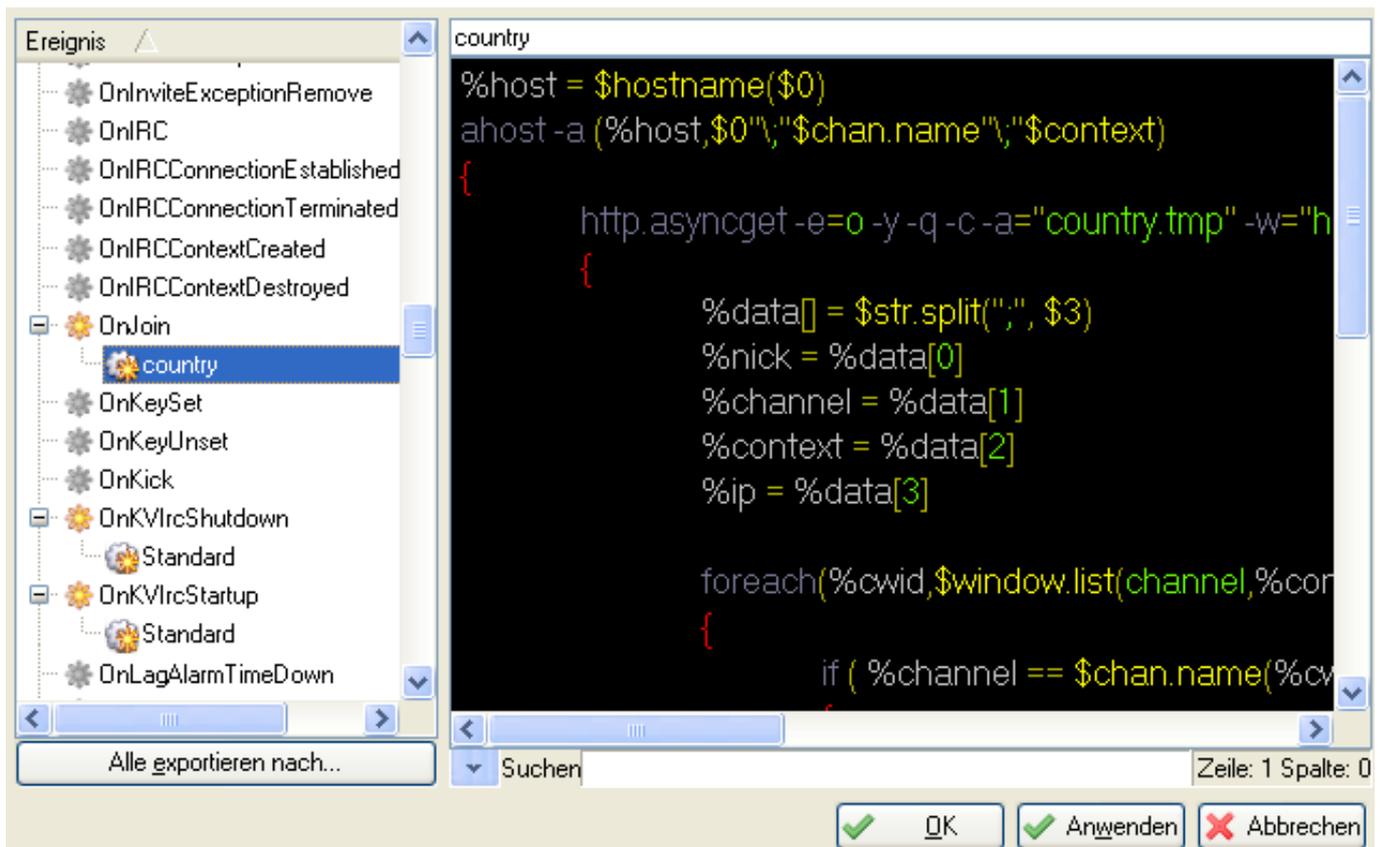
Bekommen wir die richtige Ausgabe. Das \$ signalisiert KVS das es sich um eine Funktion handelt und das die Parameter in Klammer angegeben werden müssen.



```
[22:57:44] Hallo Welt!  
[22:57:44] Hallo TheXception! Wie geht es dir?  
[22:57:44] Der Rückgabewert ist: 99
```

4.6 Ereignisse

Ich denke der Name "Ereignis" ist schon ziemlich selbsterklärend. Ereignisse sind in KVIrc Skripte die gestartet werden wenn ein bestimmtes Ereignis ausgelöst wird. Den Ereigniseditor findet ihr im Menü "Skripte" → "Ereignisse bearbeiten" oder mit der Tastenkombination "STRG+ALT+3".



Die Knöpfe haben hier die gleichen Funktionen wie beim Aliaseditor. Der einzige Unterschied ist, dass ein Ereignis einfach in dem Eingabefeld oben umbenannt werden kann. Links seht ihr eine Liste aller möglichen Ereignisse. Jeweils als Unterpunkt eines Ereignisses finden sich die schon dazu angelegten Handler. Handler sind nichts anderes wie Skripte, die zu einem Ereignis gehören. Hier auf dem Bild ist es das Skript "country", das zum Ereignis OnJoin gehört. Der gelbe Stern vor "country" heißt, dass dieser Handler aktiv ist, d.h. es wird ausgeführt wenn das OnJoin Ereignis eintritt. Mit einem Rechtsklick auf einen Handler kann man über ein Popup den Handler aktivieren oder deaktivieren.

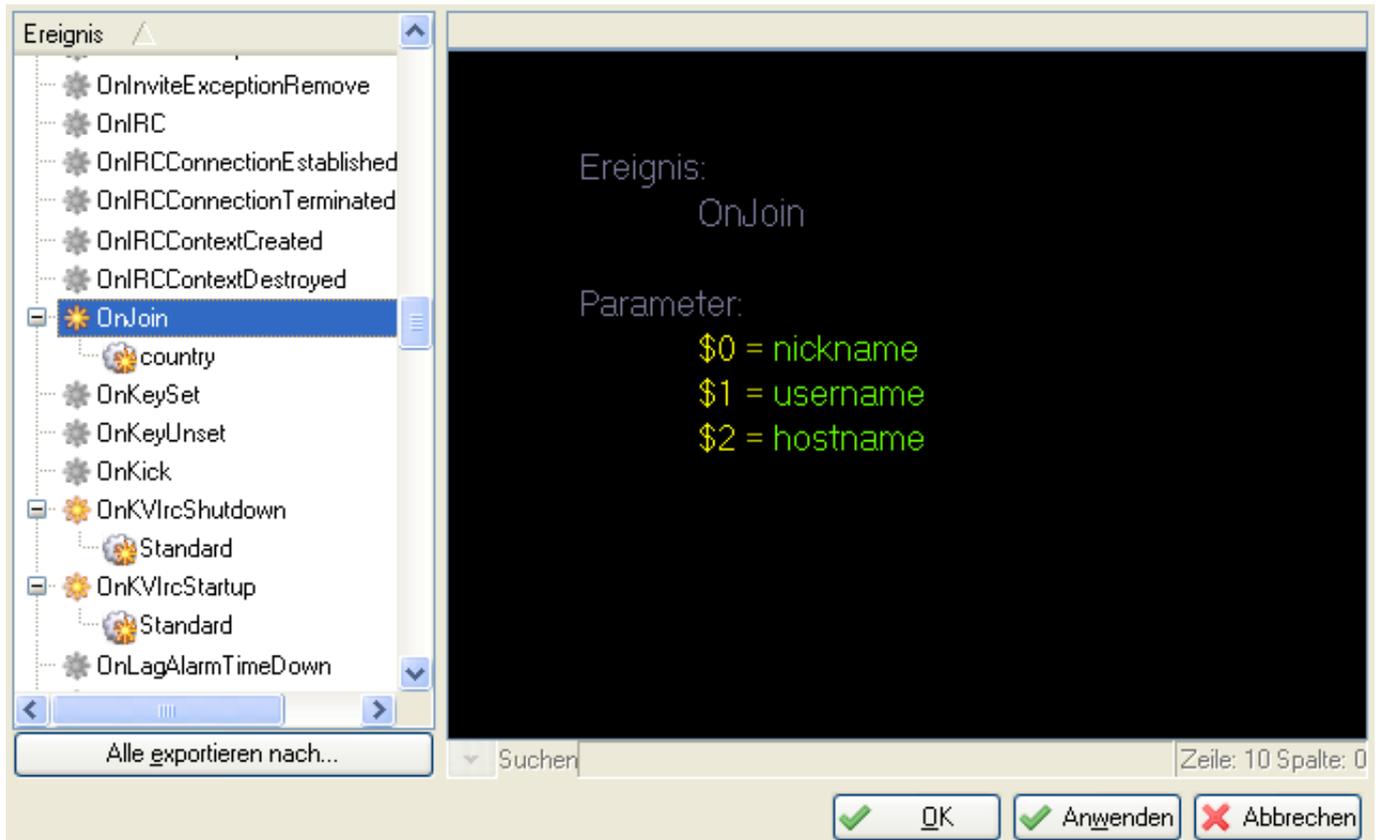
Hinweis

KVIrc deaktiviert einen Handler, wenn ein Fehler in seinem Skript vorkommt. Man kann ihn dann aber einfach wieder aktivieren.

4.6.1 Parameter

Zu jedem Ereignis übergibt KVIrc eine Reihe von Parametern, wie z.B. Nickname, Channel, ...

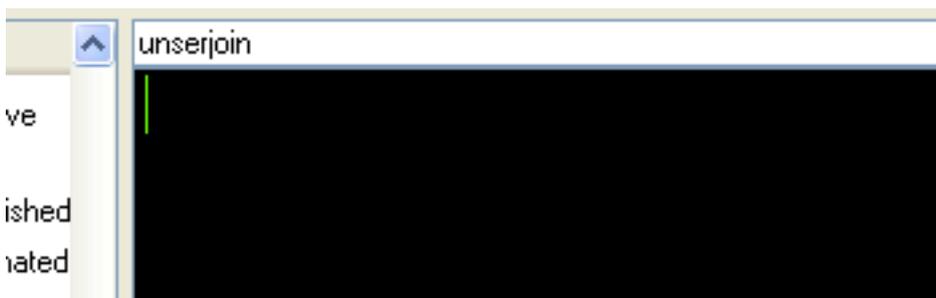
Diese können dann während des Skripts abgefragt und weiterverarbeitet werden.



Die Parameter, die zu einem Ereignis gehören, findet ihr, indem ihr einfach auf das Ereignis links in der Liste klickt. Wie gleich zu sehen ist, arbeiten die Parameter genauso wie bei den Aliassen.

4.6.2 Neue Ereignisskripte anlegen

Um ein neues Ereignis anzulegen macht ihr einen Rechtsklick auf ein Ereignis, auf das ihr reagieren wollt und wählt "Neuer Handler". Ich nehme hier "OnJoin". Nun erhalten wir einen neuen Handler der aber noch "Standard" heißt. Also wählen wir diesen Handler und geben für ihn oben einen neuen Namen ein.



Ein einfacher Klick auf "Anwenden" übernimmt die Änderung. (Auch wenn nicht sofort in der linken Liste sichtbar.)

Tragen wir, wie schon beim Alias unseren Code in das rechte Feld ein.

```
echo $0" hat den Channel betreten"
```

\$0 enthält, wie in der Parametererklärung beschrieben, den Nickname desjenigen, der gerade den Channel betreten hat. Die Ausgabe ist wie folgt:

```
[11:09:19] |thexception| hat den Channel betreten
➡ [11:09:19] |thexception| [n=kvirc@          .dip0.t-ipconnect.de] has joined
#thexception
```

Wie schon am Anfang des Tutorials gesagt, wird ein Handler schon vor der eigentlichen Ausgabe/Verarbeitung ausgeführt. Das hat ganz einfach den Sinn, dass wir die Originalausgabe von KVirc unterdrücken können wenn wir sie z.B. nicht benötigen.

4.6.3 Standardereignisse unterdrücken

Wenn wir unser neu angelegten Handler und seine Ausgabe anschauen, haben wir wann immer jemand einen Channel betritt die doppelte Ausgabe. Die unseres Skriptes und die von KVirc

KVS bietet einen einfachen Befehl um die Originalausgabe zu unterdrücken:

```
halt
```

Wir müssen in unserem Skript nur noch den "halt" Befehl aufrufen.

```
unserjoin
echo "$0" hat den Channel betreten"
halt
```

Das Ergebnis:

```
[11:18:17] |thexception| hat den Channel betreten
```

4.7 Rechnen mit Zahlen

Das Rechnen ist grundlegend recht einfach. KVS kennt die Grundrechenarten und über Aliase/Funktionen auch sin, cos etc.

Table 2: Grundoperatoren

*	Multiplizieren
/	Dividieren
+	Addieren
-	Subtrahieren

Wichtig

Damit ein Rechenausdruck auch als Rechnung verstanden wird muss er in $\$(...)$ stehen. Ansonsten würde KVS es als String werten.

Also z.B. so:

```
/echo $(1+1)
/%var = 2; echo ${%var*2}
```

4.8 Arbeiten mit Strings

Bei Strings handelt es sich um Texte. In KVirc gibt es 2 Arten Strings anzugeben. Die Eine ist den Text in Anführungszeichen zu packen und die Andere ist den Text einfach so zu schreiben. Der besondere Unterschied zwischen beiden Möglichkeiten ist, dass, wenn keine Anführungszeichen angegeben werden, mehrere aneinanderhängende Leerzeichen zu einem Zusammengefasst werden. Bei Texten in Anführungszeichen achtet KVirc auf jedes einzelne Zeichen und lässt auch mehrere Leerzeichen nacheinander zu.

```
/echo Text mit vielen           Leerzeichen
/echo "Text mit vielen         Leerzeichen"
```

Geben dementsprechend folgende Ausgabe:

```
[17:59:51] Text mit vielen Leerzeichen
[17:59:58] Text mit vielen           Leerzeichen
```

Wir können in Strings aber auch die Rückgabewerte aus Funktionen einbauen. Betrachten wir es anhand der einfachen Funktion `$me()`. Sie braucht keine Parameter und gibt uns unseren Nickname zurück, der zu dem Fenster bzw. IRC-Server gehört, bei dem wir die Funktion aufgerufen haben.

```
/echo "Hallo $me()!"
```

```
[18:05:32] Hallo theexception!
```

Das Zusammenhängen einzelner Strings in KVirc ist, im Gegensatz zu anderen Skript/Programmiersprachen, stark vereinfacht.

So genügt es einfach Texte aneinanderzuschreiben.

```
/echo $me() " ist da"
/echo "Leerzeichen          beachtet" und nicht beachtet
```

Um Strings zu Durchsuchen oder in anderer Weise zu bearbeiten, werden einem von KVS einige nützliche Funktionen bereitgestellt. Diese beginnen in der Regel mit "\$str.". Natürlich können so auch Variablen miteinander verbunden werden. Hierbei ist es bei Zahlen und Text egal ob sie gemischt werden. KVS wandelt automatisch Zahlen in Strings um und umgekehrt, vorausgesetzt der Inhalt der Variable besteht aus reinen Zahlen.

```
/%Var = "Hallo Welt"
/echo Der Inhalt meiner Variable ist: %Var
```

Wie euch bestimmt aufgefallen ist, können wir in unserem String nicht einfach Anführungszeichen setzen, das % und \$ Zeichen nicht verwenden.

Damit diese nicht von KVS interpretiert werden, müssen wir ihnen einen "\" voranstellen.

```
/echo Der Inhalt der Variable \%Var ist: %Var
```

Auch können auf diesem Weg Sonderzeichen in einen String geschrieben werden. Ein "\n" entspricht zum Beispiel einem Zeilenumbruch.

5 Schluss

Wenn das Tutorial gut ankommt, werde ich zukünftig noch Nachfolger bringen. In diesem Sinne wünsche ich euch noch viel Spaß mit KVirc.

TheXception

5.1 Dankansagungen

Vielen Dank an:

mren

für das Probe-/Korrekturlesen

5.2 Kontakt

Falls ihr noch Fragen, Anregungen oder Ähnliches habt, findet ihr mich im IRC auf dem Freenode Server in den Channels #kvirc,#kvirc.de und #thexception oder ihr schreibt mir eine email: [kvirc@thexception\(REMOVE-ME\).net](mailto:kvirc@thexception(REMOVE-ME).net)

5.3 Copyright

© 2008 TheXception

Dieses Dokument darf frei kopiert, verändert und weitergegeben werden, solange ein Hinweis auf den ursprünglichen Autor mit eingebunden wird.