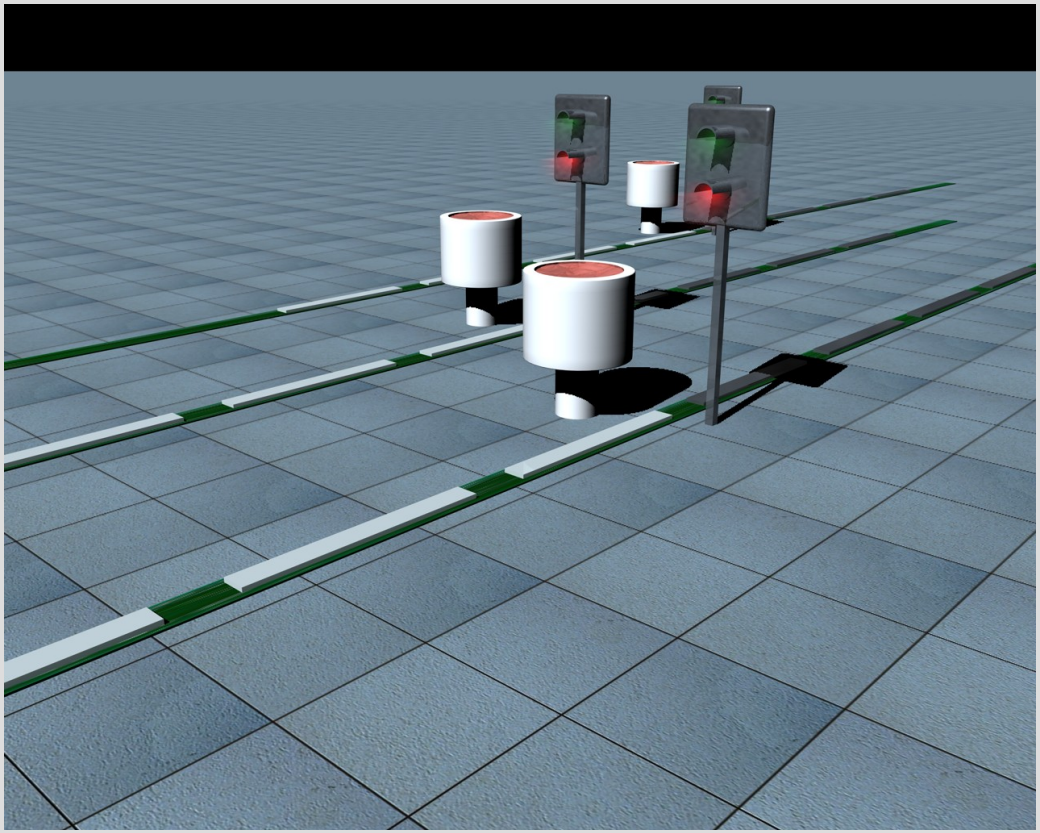
ThemaScript Workshop

von Jürgen Schmitz



ThemaScript Workshop

Übersicht

- Was ist ThemaScript und was macht man damit?
- ThemaScript Syntax, was ist dieses UPN?
- Wie funktioniert ThemaScript?

Was ist ThemaScript?

Das ist ThemaScript:

```
#DATE 28.03.2007
#DESC Standard Thema für alle neuen Züge, stdstarthook am Anfang, stdendhook am
Ende, stdreghook am Anfang (keiner am Ende)

*:bahnhof(stdstarthook,hook,stdreghook,regionhook);
winter:Start(B1,$flag,(nop),($verspaetung,$abstunde,4,/,$abstunde,rnd,
+,verspaetung),if);

herbst:Start($tempo,0,2,rnd,-,tempo
,B1,$flag,(nop),($verspaetung,$abstunde,8,/,$abstunde,2,/,rnd,+,verspaetung),if);

sommer:Start($verspaetung,10,/,verspaetung);

schwerer_winter:Start($tempo,0,2,rnd,-,tempo);
winter:Bahnhof($wurdegesteuert,(nop),($verspaetung,0,$abstunde,1,+,
$abstundev,-,3,*,rnd,+,verspaetung),if);

*:Bahnhof(B1,$flag,($verspaetung,10,/,verspaetung),(nop),if);
*:bahnhof(stdendhook,hook);
```

Was ist ThemaScript und was macht man damit?

ThemaScript ist eine simple Script-Sprache, die es erlaubt, einige Parameter von Zügen zu ändern, unter anderem:

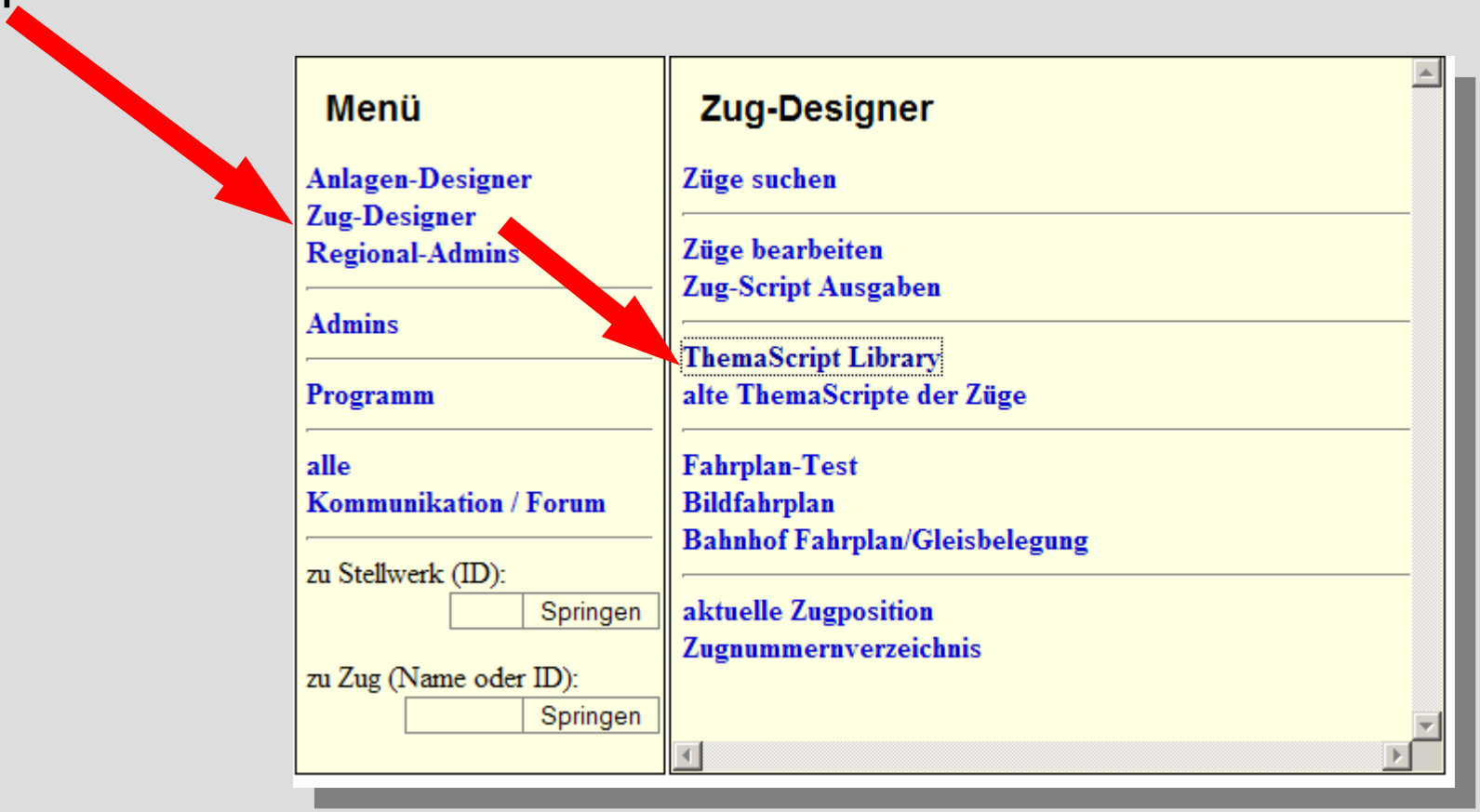
- die Verspätung
- das Tempo
- die Länge
- ob ein Zug aktiv ist

Was ist ThemaScript und was macht man damit?

- ThemaScript bietet Funktionalitäten, um Parameter eines Zuges abzufragen
- ThemaScript kann etliche mathematische und logische Operationen
- ThemaScript erlaubt strukturelle Programmierung durch Unterfunktionen und Hooks

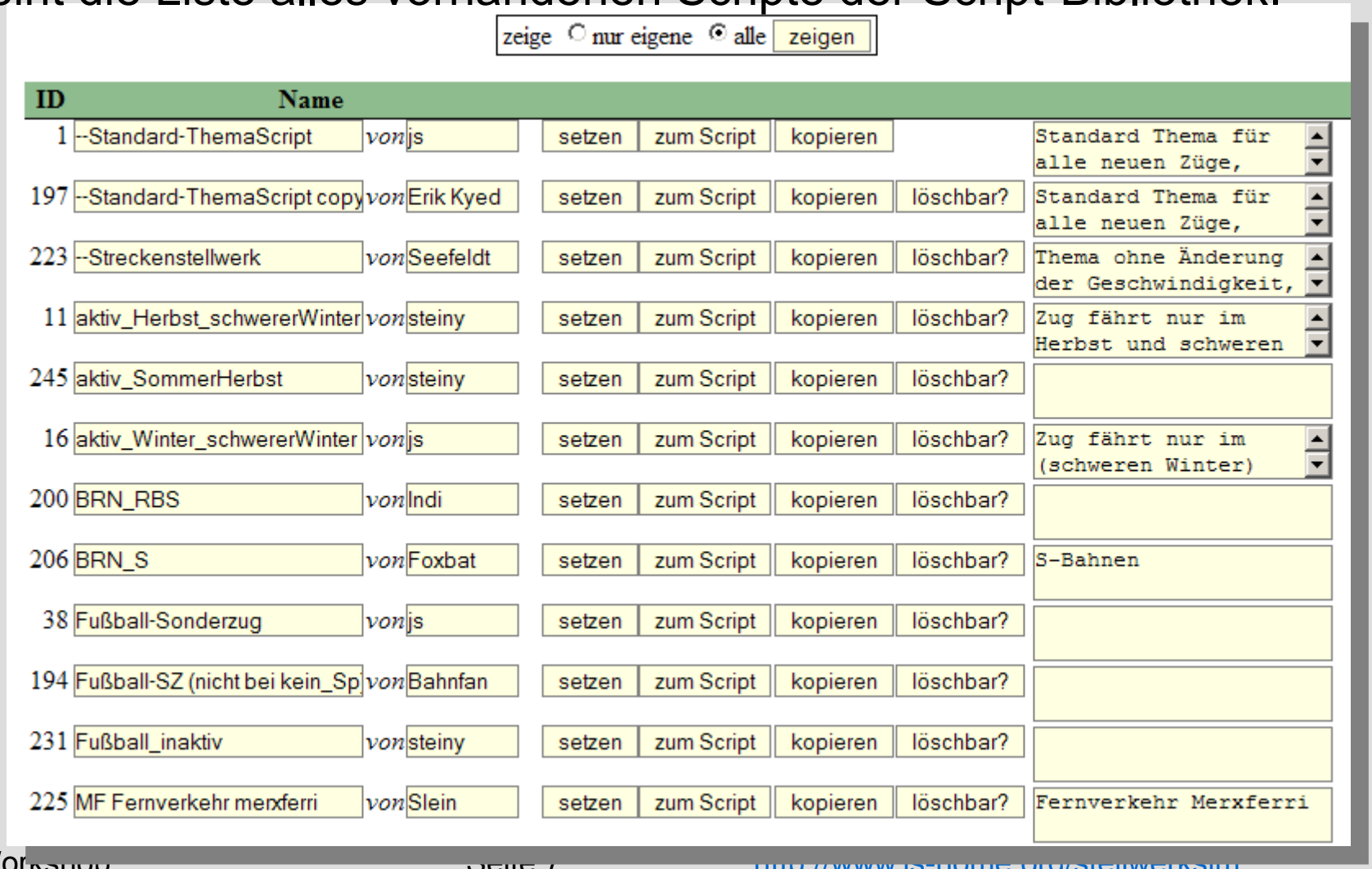
Wo ist dieses ThemaScript überhaupt?

Dort findet man die ThemaScript Funktion:



Wo ist dieses ThemaScript überhaupt?

Es erscheint die Liste alles vorhandenen Scripte der Script-Bibliothek.



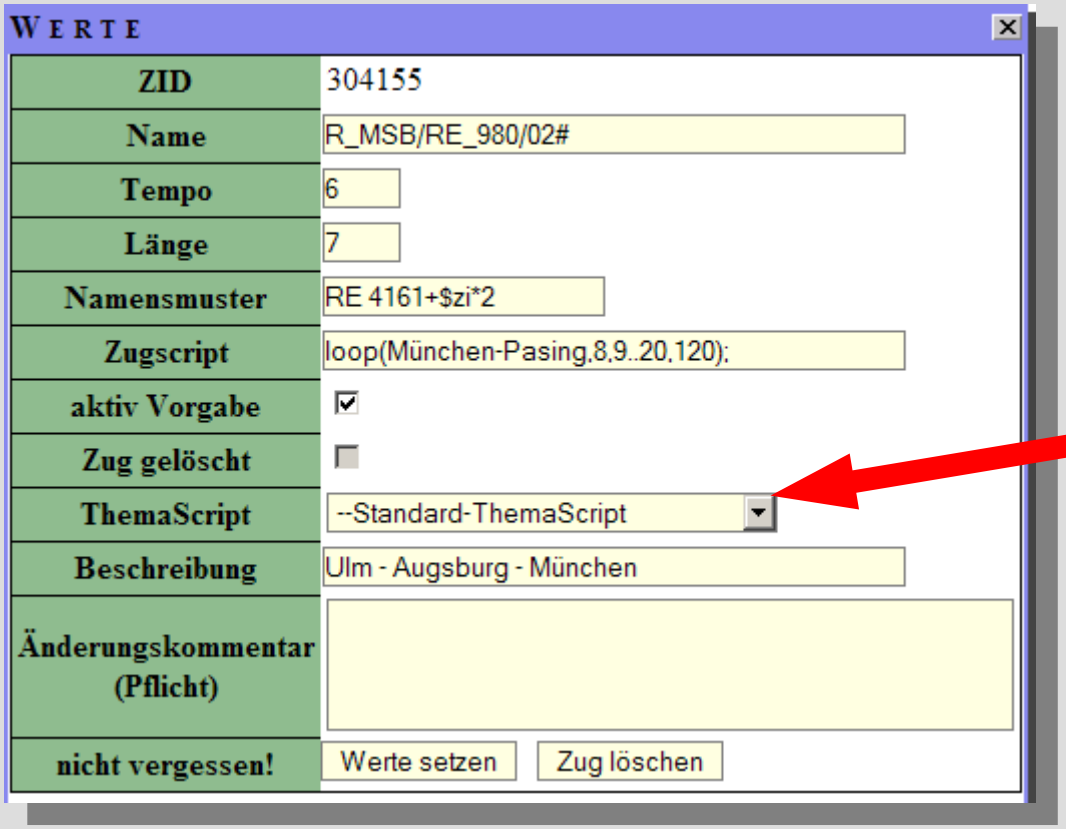
zeige ○ nur eigene ◉ alle [zeigen]

ID	Name							
1	--Standard-ThemaScript	von	js	setzen	zum Script	kopieren		Standard Thema für alle neuen Züge,
197	--Standard-ThemaScript copy	von	Erik Kyed	setzen	zum Script	kopieren	löschbar?	Standard Thema für alle neuen Züge,
223	--Streckenstellwerk	von	Seefeldt	setzen	zum Script	kopieren	löschbar?	Thema ohne Änderung der Geschwindigkeit,
11	aktiv_Herbst_schwererWinter	von	steiny	setzen	zum Script	kopieren	löschbar?	Zug fährt nur im Herbst und schweren
245	aktiv_SommerHerbst	von	steiny	setzen	zum Script	kopieren	löschbar?	
16	aktiv_Winter_schwererWinter	von	js	setzen	zum Script	kopieren	löschbar?	Zug fährt nur im (schweren Winter)
200	BRN_RBS	von	Indi	setzen	zum Script	kopieren	löschbar?	
206	BRN_S	von	Foxbat	setzen	zum Script	kopieren	löschbar?	S-Bahnen
38	Fußball-Sonderzug	von	js	setzen	zum Script	kopieren	löschbar?	
194	Fußball-SZ (nicht bei kein_Sp	von	Bahnfan	setzen	zum Script	kopieren	löschbar?	
231	Fußball_inaktiv	von	steiny	setzen	zum Script	kopieren	löschbar?	
225	MF Fernverkehr merxferri	von	Slein	setzen	zum Script	kopieren	löschbar?	Fernverkehr Merxferri

Wo ist dieses ThemaScript überhaupt?

Geht man dann zu einem Script, bekommt man das zusehen:

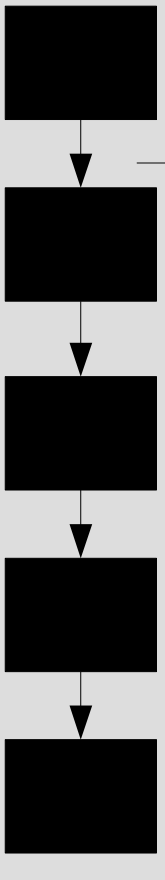
Wo ist dieses ThemaScript überhaupt?



Im Zugeditor in den Werten (Stammdaten) stellt man das Script für Züge ein.

Wann läuft ThemaScript?

Lebenszyklus eines Zuges (Beispielzugname Z1)

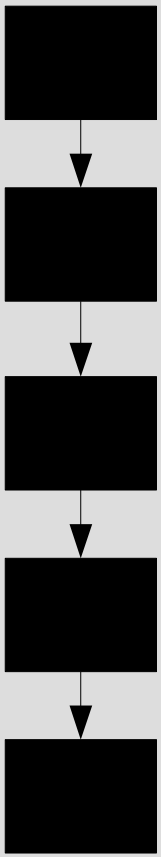


Start: Zug wird mit Defaultwerten belegt und auf 1. Bahnhof gesetzt

Zug verlässt Bahnhof, wird auf nächsten Bahnhof gesetzt

Zug verlässt letzten Bahnhof, Zug wird auf Start gesetzt

Wann läuft ThemaScript?



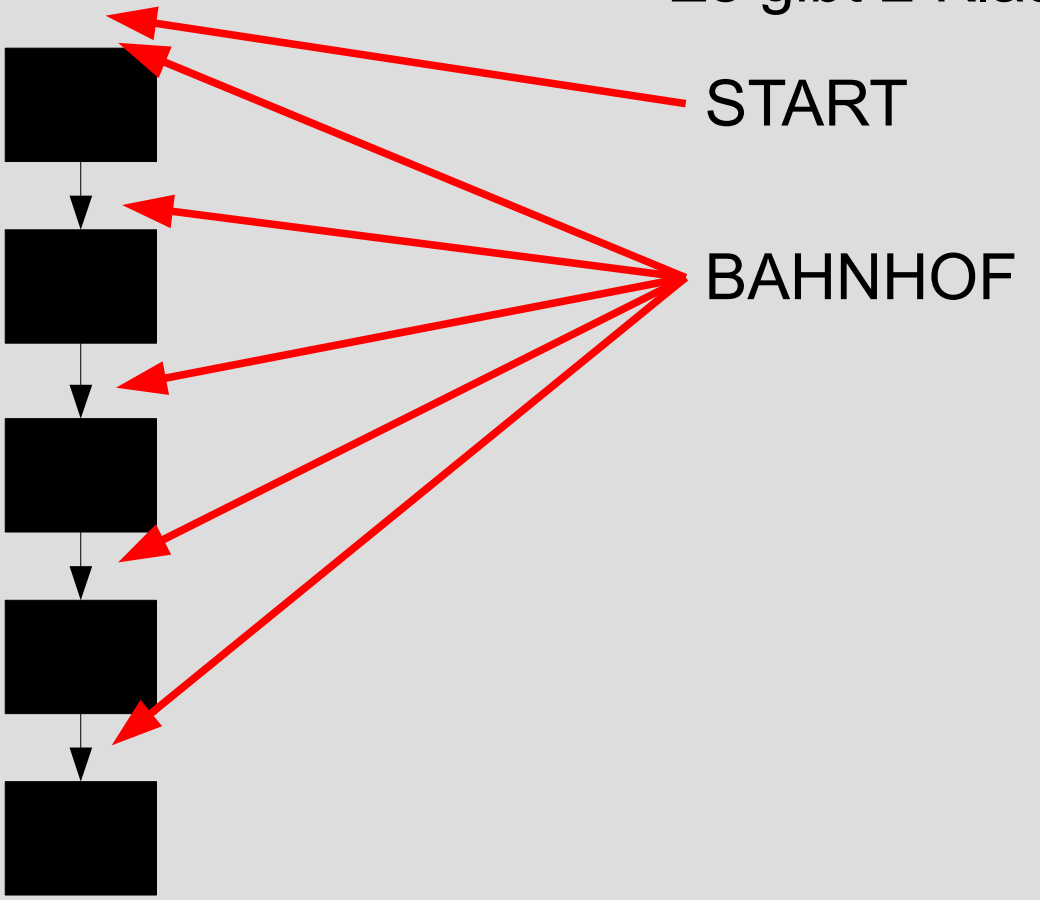
Um diese verschiedenen Punkte im Leben eines Zuges mit Scripten zu versehen, kennt ThemaScript „Klassen“. Klassen geben den Punkt an, wann ein Script ausgeführt werden soll.

Es gibt 2 Klassen:

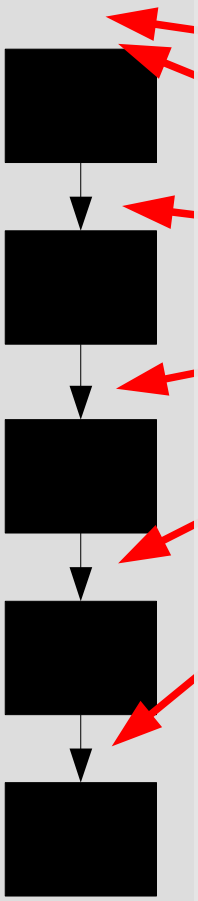
- Start
- Bahnhof

Wann läuft ThemaScript?

Es gibt 2 Klassen:



Wann läuft ThemaScript?



- Sind zwei Stellwerke per ÜP verknüpft und beide Stellwerke besetzt, dann sind viele Funktionen in der Klasse „Bahnhof“ unsinnig und irritierend!
- Zur Beispiel wäre eine Änderung der Zuglänge dort nicht sinnvoll und kann auch nicht garantiert werden!
- Ist ein Stellwerk hingegen nicht besetzt, macht ein Script manchmal schon Sinn!
- Scripte der Klasse „Bahnhof“ sind am besten zwischen Stellwerken mit langer Fahrzeit dazwischen.
- Scripte der Klasse „Start“ sind die Hauptverwendung, z.B. um Züge gar nicht erst fahren zu lassen oder deren Tageslänge oder -tempo zu ändern.

ThemaScript Syntax

Wie sage ich es nun als Programm?

- ThemaScript ist in Blöcke unterteilt
- Blöcke werden nur ausgeführt, wenn die Bedingungen des Block-Kopfes zutreffen
- Innerhalb eines Blocks steht der Programmcode als UPN-Code (umgekehrt polnische Notation, dazu später!)
- Aufbau:

```
blockkopf1(blockcode1);
blockkopf2(blockcode2);
```

ThemaScript Syntax

Der Block-Kopf.

Der Kopf eines Blocks, der Block-Kopf, besteht aus einer Liste von „Bedingungen“, die erfüllt sein müssen, damit der Block ausgeführt wird.

```
THEMA:KLASSE
```

- Das THEMA ist dabei das Tagesthema, für das der Block gilt, also z.B. „winter“, „sommer“, „schwerer_winter“.

Weitere Themen nach Bedarf/Ideen möglich.

- Die KLASSE ist eine der beiden genannten Klasse: „START“ oder „BAHNHOF“

- Beispiel:

```
winter:Start(BLOCK-CODE);
```

ThemaScript Syntax

Der Block-Kopf.

Soll ein Block bei jedem Thema ausgeführt werden, kann der Platzhalter „*“ als Themenname genutzt werden:

```
*:KLASSE
```

- Das gilt nur für das Thema, die Klasse muss **immer** angegeben werden.

- Mehrere zutreffende Block-Köpfe werden in der Reihenfolge ihres Eintrags im Script ausgeführt.

- Beispiel:

```
*:Start(blockcode1);
*:Start(blockcode2);
```

ThemaScript Syntax

Der erweiterte Block-Kopf.

Der „erweiterte“ Kopf enthält noch zusätzliche Bedingungen, die erfüllt sein müssen. Der Teil ist noch in der *Erprobungsphase*, erspart aber etlichen umständlichen Script-Code!

```
THEMA:KLASSE:AIDLISTE:ZIDLISTE
```

- Jede Liste (AIDLISTE/ZIDLISTE) besteht dabei aus einer durch Komma getrennten Liste von IDs: Stellwerk-IDs (AID) und Zug-IDs (ZID).
- Bei Zügen ist das die ID des Templates.
- Wird eine Liste nicht benötigt kann sie am Ende entfallen.
- Wird eine Liste in der Mitte nicht benötigt, kann/muss sie durch den Platzhalter „*“ beschrieben werden.

ThemaScript Syntax

Der erweiterte Block-Kopf.

- Durch die Angabe einer AID-Liste gilt ein Script nur für einen oder einige Bahnhöfe.
- Durch die Angabe einer ZID-Liste gilt ein Script nur für einen oder einige Züge.
- Durch Angabe von AID- und ZID-Liste gilt ein Script nur für einen oder einige Züge, wenn sie durch die angegebenen Bahnhöfe fahren.
- Eine AID-Liste macht nur in der Klasse „Bahnhof“ Sinn.

ThemaScript Syntax

Der erweiterte Block-Kopf.

Beispiele:

```
*:Bahnhof:4,7
```

Gilt für jedes Thema beim Bahnhof (die Klasse Start macht hier keinen Sinn) für die Bahnhöfe 4 und 7.

```
Winter:Start:*:42045
```

Gilt für das Thema „Winter“ beim Start für die Züge des Templates 42045.

```
*:Bahnhof:4:430117,430118
```

Gilt für jedes Thema beim Bahnhof wenn ein Zug vom Template 430117 oder 430118 den Bahnhof mit der AID 4 durchfährt.

ThemaScript Syntax

Der erweiterte Block-Kopf.

Die Freigabe der Funktionalität ist in Kürze geplant. Allerdings soll neben den ZIDs auch die Angabe der Templatenamen möglich sein. Wie das aussehen soll ist zur Zeit noch nicht geklärt.

Durch diese Erweiterung können etliche Scripte vereinfacht werden was nicht zuletzt der Verarbeitungsgeschwindigkeit zugute kommt.

ThemaScript Syntax

Der Block-Code.

ThemaScript ist eine auf der UPN (umgekehrt polnische Notation) basierende Stapelmaschine.

Vorteile:

- relativ leicht und schnell kann so ein Ausdruck vom System analysiert und verarbeitet werden
- durch die einfache Syntax werden viele Fehler vermieden, vor allem Folgefehler für andere Blöcke
- es gibt keinen Datenmüll am Ende, es muss nur der Stapel gelöscht werden, sonst nichts

ThemaScript Syntax

umgekehrt polnische Notation – Wikipedia

Wikipedia:
Die **Umgekehrte Polnische Notation** (kurz *UPN*), auf englisch Reverse Polish Notation (kurz RPN), auch *Postfixnotation* genannt, ist eine von der Polnischen Notation abgeleitete Schreibweise bzw. Eingabelogik für die Anwendung von Operationen. Bei der umgekehrten polnischen Notation werden zunächst die Operanden niedergeschrieben bzw. eingegeben und danach der darauf anzuwendende Operator.

In der Informatik ist die UPN deshalb von Interesse, weil sie eine stapelbasierte Abarbeitung ermöglicht: Operanden werden beim Lesen auf den Stapel gelegt, ein Operator holt sich die Anzahl an Operanden vom Stapel (Stack), die seiner Stelligkeit entspricht und legt das Ergebnis der Operation wieder auf dem Stapel ab. Am Ende liegt dann das Ergebnis des Terms oben auf dem Stapel. Deshalb bildet die UPN die Grundlage für stapelbasierte Programmiersprachen wie Forth, RPL, PostScript oder die Anweisungsliste im SPS-Bereich.

ThemaScript Syntax

umgekehrt polnische Notation – ein Beispiel

Es soll 3 und 4 addiert werden. Die Operanden sind dann 3 und 4, der Operator ist „+“.

In der Schule ist die erlernte Notation die (sequenziell organisierende) Infix-Notation „3+4“. In der umgekehrten polnischen Notation wird hingegen „3 4 +“ geschrieben (zwischen 3 und 4 ist ein Leerraum, damit die Zahlen von der Zahl 34 unterschieden werden können).

```
3 4 +
```

Für eine bessere Optik statt des Leerraums ein Komma in ThemaScript:

```
3,4,+
```

ThemaScript Syntax

umgekehrt polnische Notation – ein Beispiel

Weiteres Beispiel (in der Infix-Notation): „(3+4)×5“. Bei der UPN können die Klammern entfallen, alle Operationen (hier „+“ und „×“) arbeiten mit den beiden oberen Elementen des Stapels. Das Beispiel heißt in UPN dann: „3 4 + 5 ×“ (zuerst wird der Klammerausdruck „3 4 +“ausgerechnet, danach die hierbei entstandene Zahl 7 mit 5 multipliziert).

```
3,4,+,5,*
```

Dieses Prinzip ist auch in der deutschen Sprache zu finden:

- Die Wäsche waschen.
- Das Brot schneiden.
- Mehl und Eier mischen.

ThemaScript Syntax

umgekehrt polnische Notation

In ThemaScript steht dieser Code innerhalb eines Blocks. Der Code steht dabei in Klammern, das sieht dann so aus:

```
(3,4,+)
(3,4,+,5,*)
```

ThemaScript Syntax

umgekehrt polnische Notation – als Bild

Was passiert da nun, wenn diese Befehle ausgeführt werden:

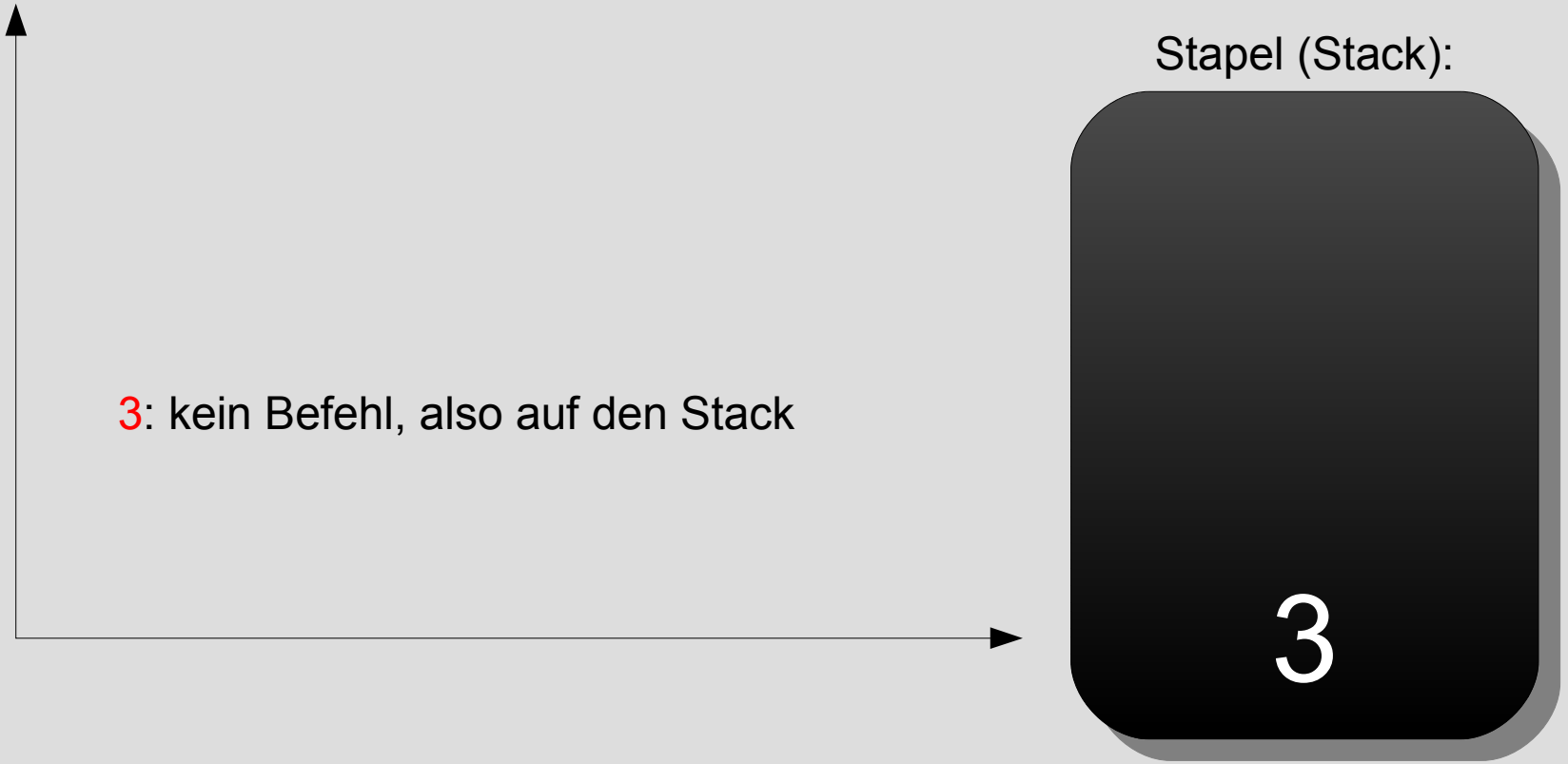
```
(3,4,+,5,*)
```

ThemaScript Syntax

umgekehrt polnische Notation – als Bild

Was passiert da nun, wenn diese Befehle ausgeführt werden:

```
(3,4,+,5,*)
```



ThemaScript Syntax

umgekehrt polnische Notation – als Bild

Was passiert da nun, wenn diese Befehle ausgeführt werden:

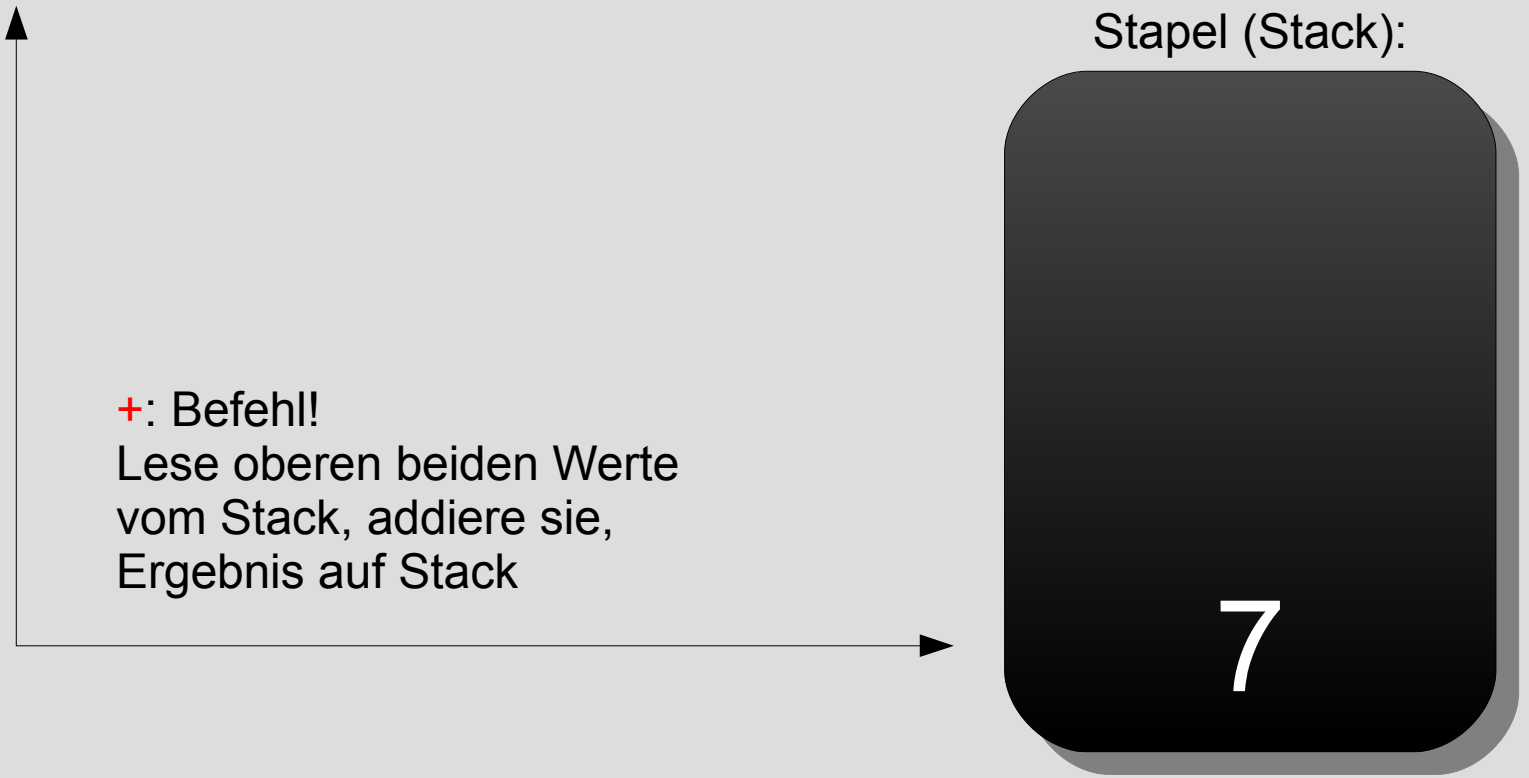
```
(3,4,+,5,*)
```

ThemaScript Syntax

umgekehrt polnische Notation – als Bild

Was passiert da nun, wenn diese Befehle ausgeführt werden:

```
(3,4,+,5,*)
```

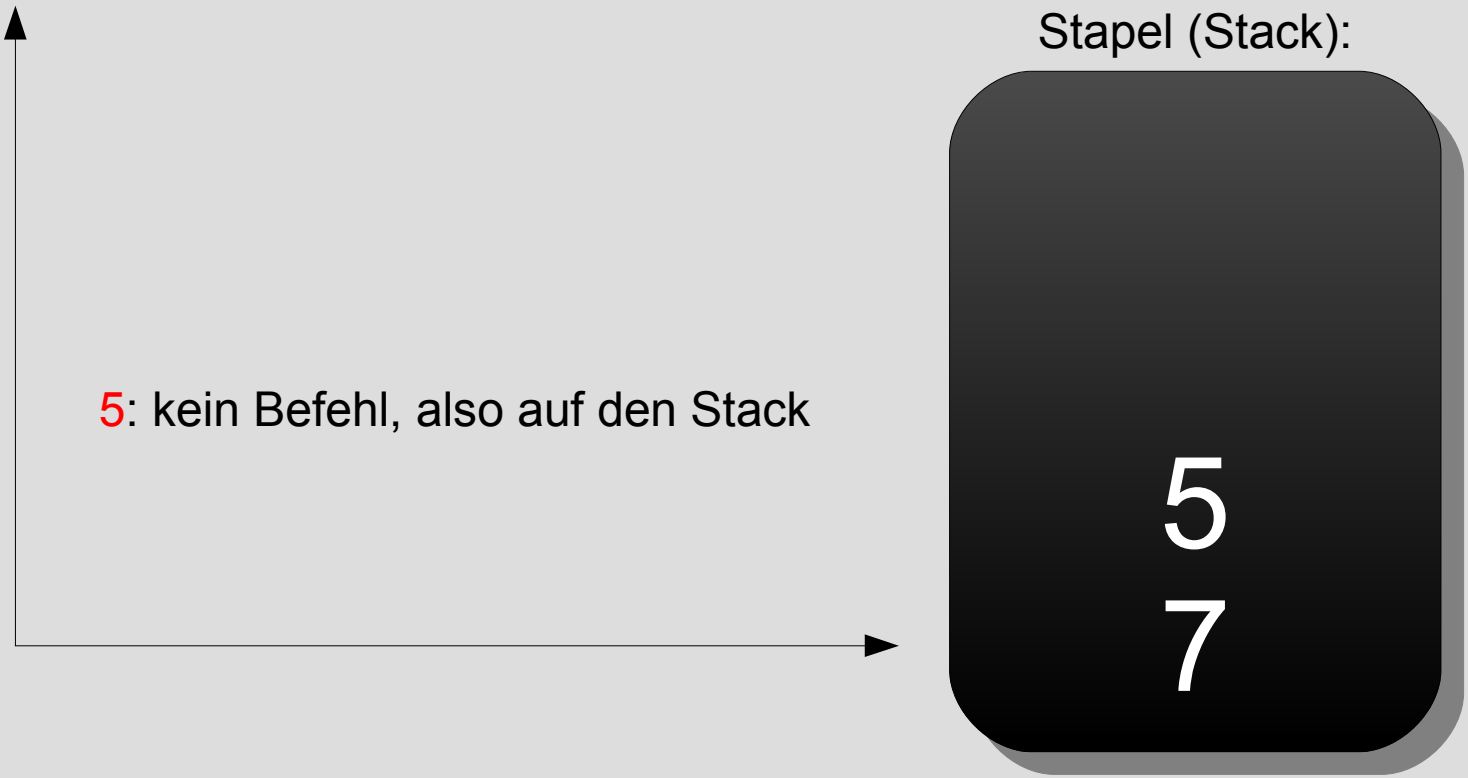


ThemaScript Syntax

umgekehrt polnische Notation – als Bild

Was passiert da nun, wenn diese Befehle ausgeführt werden:

```
(3,4,+,5,*)
```

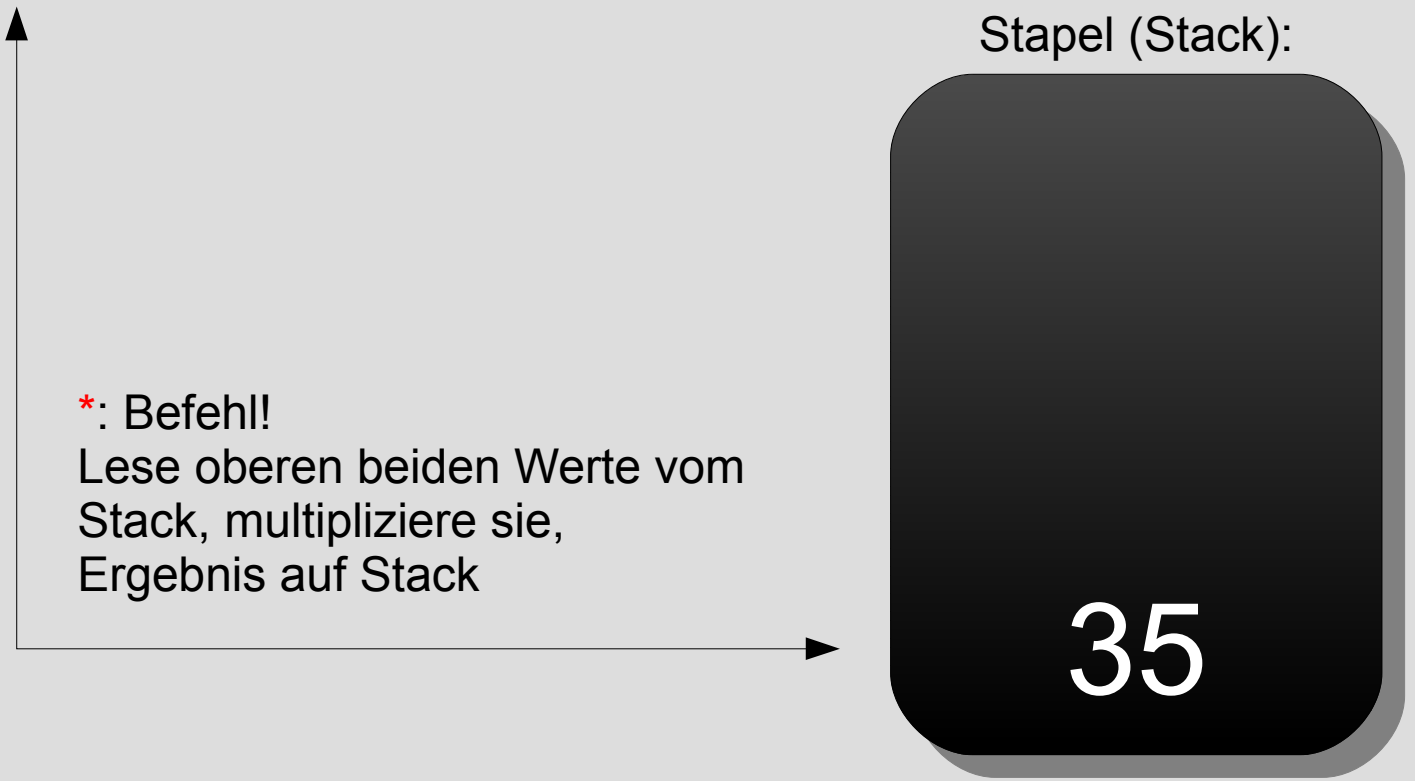


ThemaScript Syntax

umgekehrt polnische Notation – als Bild

Was passiert da nun, wenn diese Befehle ausgeführt werden:

```
(3,4,+,5,*)
```



ThemaScript Syntax

Der Block-Code.

Ein vollständiger Block sieht dann so aus:

```
winter:Start(3,4,+,5,*,print);
```

- dabei ist „winter“ das Thema und „Start“ die Klasse, d.h. der Block soll nur im Winter und nur zu Beginn einer Zugfahrt ausgeführt werden.
- es wird (3+4)*5 berechnet – in dem Zusammenhang zwar sinnlos, aber es geht
- der Befehl „print“ gibt den Wert aus, der auf dem Stapel liegt

Wie funktioniert ThemaScript?

Ein erstes Beispiel.

Das folgende Beispielscript erhöht die Verspätung eines Zuges in Abhängigkeit von der Tageszeit seiner nächsten (ersten) Abfahrt mit etwas Zufall:

```
winter:Start(
 $verspaetung,$abstunde,4,/,$abstunde,rnd,+,verspaetung
);
```

Gesprochen: Zu der jetzigen Verspätung wird zwischen der Abfahrtzeit ein viertel und der Abfahrtzeit ein Zufallswert hinzu gezählt

```
verspaetung = $verspaetung + rnd( $abstunde / 4 , $abstunde )
```

ThemaScript Workshop

10 Minuten Pause?

ThemaScript Kommentare

Kommentare sollen helfen, ein Script verständlich zu beschreiben – für andere aber auch für einen selbst um auch nach Wochen noch zu verstehen, was man da gemacht hat.

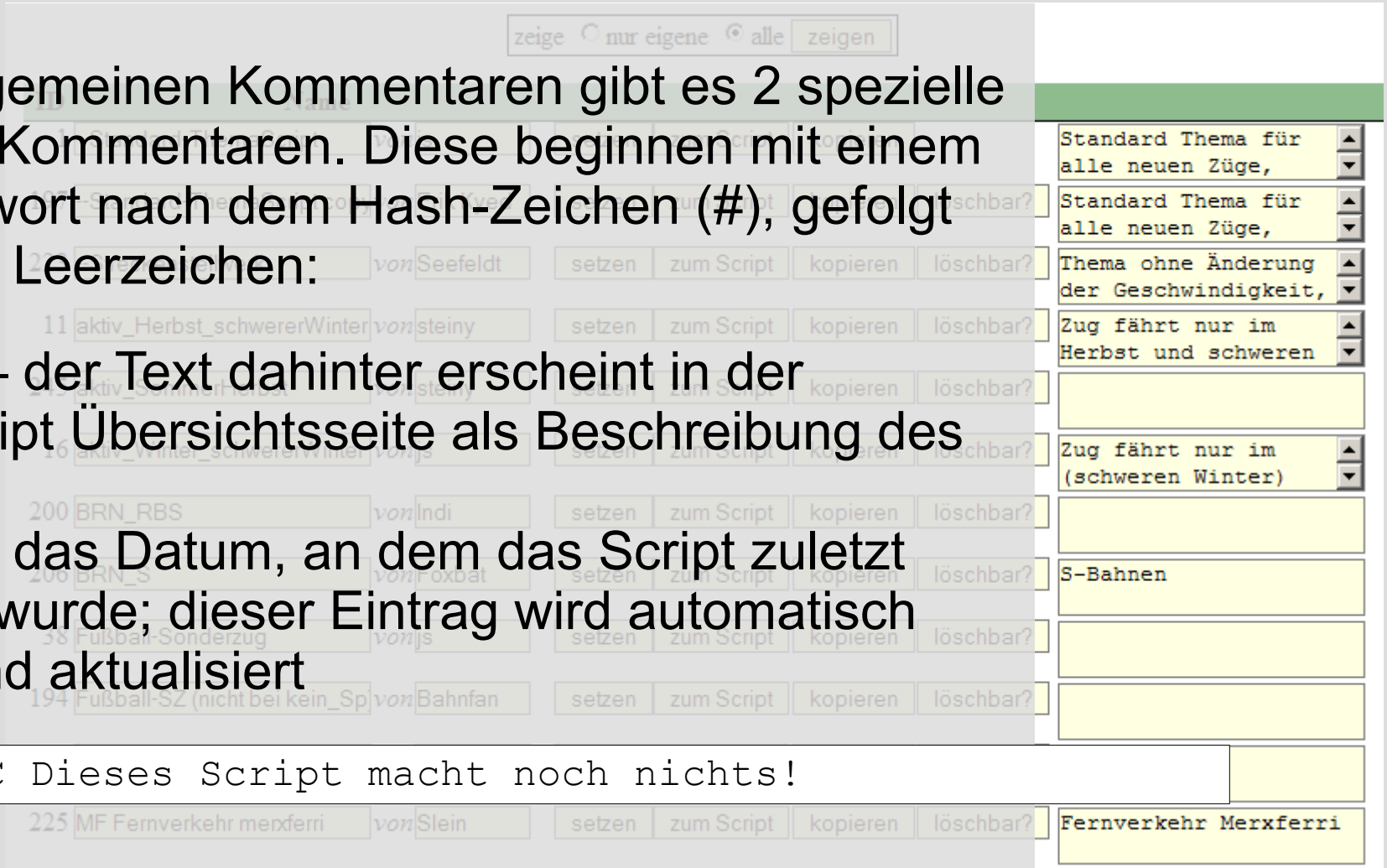
Kommentare sind nur zeilenweise möglich, Kommentarzeilen beginnen mit einem Hash-Zeichen (#).

```
# Die ist ein Kommentar!
```

ThemaScript spezielle Kommentare

Neben allgemeinen Kommentaren gibt es 2 spezielle Arten von Kommentaren. Diese beginnen mit einem Schlüsselwort nach dem Hash-Zeichen (#), gefolgt von einem Leerzeichen:

- #DESC – der Text dahinter erscheint in der ThemaScript Übersichtsseite als Beschreibung des Scripts
- #DATE – das Datum, an dem das Script zuletzt verändert wurde; dieser Eintrag wird automatisch erzeugt und aktualisiert




```
#DESC Dieses Script macht noch nichts!
```

ThemaScript Grundbefehle

Mathematische Befehle

Befehl	Funktion	Parameterzahl
+	addition	2
-	subtraktion	2
*	multiplikation	2
/	division	2
rnd	Zufallszahl zwischen Parametern inklusive Parametern, größerer Wert muß näher am Befehl stehen	2
oneof	Zufallswert aus Liste von Werten, Parameter direkt am Befehl gibt Anzahl an Werten an, die dann folgen	Mindstens 1

ThemaScript Grundbefehle

Mathematische Befehle

Befehl	Beispiel
+	5,4,+
-	5,4,-
*	5,4,*
/	5,4,/
rnd	3,8,rnd
oneof	1,3,5,3,oneof

ThemaScript Grundbefehle

Boolsche Befehle

Befehl	Funktion	Parameterzahl
=	Gleichheit	2
<	kleiner als, größerer Wert näher am Befehl	2
>	größer als	2
<=	kleiner oder gleich	2
>=	größer oder gleich	2
and	logisches UND	2
or	logisches ODER	2
not	logisches NICHT	1

ThemaScript Grundbefehle

Ausgabe (Debug) Befehle

Befehl	Funktion	Parameterzahl
print	gibt obersten Wert von Stack aus, Wert ist damit gelöscht	1
spy	gibt obersten Wert von Stack aus, legt ihn dann wieder dort ab	1

ThemaScript Programmflussbefehle

Befehl	Funktion	Parameterzahl
sub	ruft ein anderes ThemaScript auf, dessen ID als Parameter übergeben wird, und führt dort die Blöcke mit gleichem Thema und Klasse aus	1
tsub	ruft ein anderes ThemaScript auf, neben der ID wie bei sub kann noch ein (beliebiges) Thema übergeben werden	2
if	bedingte Ausführung	3
nop	tut nichts, bei if oft benötigt	0

ThemaScript Programmflussbefehle

Befehl	Beispiel
sub	1,sub
tsub	meinthema,80,tsub
if	8,9,<,(nop),(4,print),if

if (8<9) : (nop) else (print 4)

ThemaScript Hooks/Haken

Befehlsliste

Befehl	Funktion	Parameterzahl
hook	ruft ein anderes ThemaScript auf, dessen Name mit „ZH-“ beginnt, gefolgt vom Regionskürzel des Zuges, gefolgt vom übergebenen Parameter	1
thook	wie hook, es wird aber noch ein Thema übergeben, wie bei tsub	2
regionhook	wie hook, allerdings muß der gesuchte Scriptname mit „RH-“ beginnen und es wird das Regionskürzel des durchfahrenen Bahnhofs verwendet	1
tregionhook	wie regionhook, es wird noch ein Thema übergeben, wie bei tsub	2

ThemaScript Hooks/Haken

Verwendung

Hooks erlauben recht flexibel eigenen Code für eine Region in generelle Scripte einzubauen (einzuhaken). Dazu muss im generellen Script nur ein Hook-Aufruf drin sein. Wird kein passendes Script gefunden, wird der Befehl einfach übersprungen.

Das Default-Script (ID 1) hat mehrere Hooks:

Am Anfang:

```
*:start(stdstarthook,hook);
*:bahnhof(stdstarthook,hook,stdreghook,regionhook);
```

Am Ende:

```
*:start(stdendhook,hook);
*:bahnhof(stdendhook,hook);
```

ThemaScript Hooks/Haken

Hooks im Default-Script

```
*:start(stdstarthook,hook);
```

- sucht beim Initialisieren des Zuges nach „ZH-*[1]-stdstarthook“
- *[1] ist das Regionskürzel des Zuges, zu dem der Zug gehört (Regionalzüge)
- ein Regionshook ist hier nicht möglich

ThemaScript Hooks/Haken

Hooks im Default-Script

```
*:bahnhof(stdstarthook,hook,stdreghook,regionhook);
```

- sucht beim Betreten des Bahnhofs nach „ZH-*[1]-stdstarthook“
- sucht beim Betreten des Bahnhofs nach „RH-*[2]-stdreghook“
- *[1] ist das Regionskürzel des Zuges, zu dem der Zug gehört (Regionalzüge)
- *[2] ist das Regionskürzel der Bahnhofsregion, in dem der Zug fährt gerade fährt

ThemaScript Hooks/Haken

Hooks im Default-Script

```
*:start(stdendhook,hook);
```

- sucht beim Initialisieren des Zuges nach „ZH-*[1]-stdstarthook“
- *[1] ist das Regionskürzel des Zuges, zu dem der Zug gehört (Regionalzüge)
- der Befehl wird nach alles anderen Script-Befehlen ausgeführt und **nicht etwa**, wenn der Zug ein Stellwerk verlässt

ThemaScript Hooks/Haken

Hooks im Default-Script

```
*:bahnhof(stdendhook,hook);
```

- sucht beim Betreten des Bahnhofs nach „ZH-*[1]-stdendhook“
- *[1] ist das Regionskürzel des Zuges, zu dem der Zug gehört (Regionalzüge)
- ein Regionshook ist hier nicht vorgesehen
- der Befehl wird nach alles anderen Script-Befehlen ausgeführt und **nicht etwa**, wenn der Zug ein Stellwerk verlässt

ThemaScript Zug-Befehle

Grundsätzliches

- es können Werte gelesen werden
- es können Werte gesetzt werden
- manche Werte können nur in der Klasse Bahnhof oder Start gelesen oder gesetzt werden
- die Werte können in den Klasse Bahnhof und Start unterschiedliche Bedeutung haben (Details später)
- Befehle um Werte zu lesen beginnen immer mit einem $ (Dollar-Zeichen)

ThemaScript Zug-Befehle

Einschränkungen

Im Handbuch werden folgende Kürzel genutzt, um die Verfügbarkeit der Befehle zu beschreiben:

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

ThemaScript Zug-Befehle

Befehl	`verspätung / verspaetung`
Einschränkung	`(L0) (S)`
Parameterzahl beim lesen	`0`
Parameterzahl beim setzen	`1`

Die Verspätung eines Zuges in Minuten.

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

ThemaScript Zug-Befehle

Befehl	`tempo`
Einschränkung	`(L0) (S)`
Parameterzahl beim lesen	`0`
Parameterzahl beim setzen	`1`

Die Geschwindigkeit eines Zuges.

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

ThemaScript Zug-Befehle

Befehl	`länge / laenge`
Einschränkung	`(L0) (S)`
Parameterzahl beim lesen	`0`
Parameterzahl beim setzen	`1`

Die Länge eines Zuges.
Im Zusammenhang mit ÜPs sollte die Länge nicht verändert werden.

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

ThemaScript Zug-Befehle

Befehl	aktiv
Einschränkung	(L0) (S)
Parameterzahl beim lesen	0
Parameterzahl beim setzen	1

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

Legt fest, ob ein Zug an dem Tag fährt. Als Parameterwerte sind „J“ und „N“ erlaubt. Nein ist dabei dominant, d.h. ist ein Zug einmal durch ein Script deaktiviert, bleibt er es bis zum Ende des Spieltags. Der Werte sollte nur in der Klasse „Start“ gesetzt werden, da es sonst zu Problemen kommen kann. Lesen geht immer.

ThemaScript Zug-Befehle

Befehl	anstunde
Einschränkung	(L0) (:SB)
Parameterzahl beim lesen	0
Parameterzahl beim setzen	-

Gibt in der Klasse „Start“ die früheste Ankunftsuhrzeit (davon deren Stunde) zurück, in der Klasse „Bahnhof“ die früheste Ankunftsuhrzeit (Stunde) im besuchten Bahnhof.

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

ThemaScript Zug-Befehle

Befehl	`abstunde`
Einschränkung	`(L0) (:SB)`
Parameterzahl beim lesen	`0`
Parameterzahl beim setzen	`-`

(L0) Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L) Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S) Befehl unterstützt Setzen
(:S) Befehl darf nur in Klasse Start
(:B) Befehl darf nur in Klasse Bahnhof
(:SB) Befehl gibt in den Klassen unterschiedliche Werte zurück

Gibt in der Klasse „Start“ die späteste Abfahrtsuhrzeit (davon deren Stunde) zurück, in der Klasse „Bahnhof“ die späteste Abfahrtsuhrzeit (Stunde) im besuchten Bahnhof.

ThemaScript Zug-Befehle

Befehl	`abstundev`
Einschränkung	`(L0) (:B)`
Parameterzahl beim lesen	`0`
Parameterzahl beim setzen	`-`

Gibt die späteste Abfahrtsuhrzeit (Stunde) im vorherigen besuchten Bahnhof.

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

ThemaScript Zug-Befehle

Befehl	`flag`
Einschränkung	`(L) (:B)`
Parameterzahl beim lesen	1
Parameterzahl beim setzen	-

Prüft, ob das als Parameter übergebene Flag an einem der Halte im Bahnhof vorkommt.
Unterstützte Flags:
R, E, K, F, A, B1, B2

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

ThemaScript Zug-Befehle

Befehl	`wurdegesteuert`
Einschränkung	`(L0) (:B)`
Parameterzahl beim lesen	`0`
Parameterzahl beim setzen	`-`

Gibt TRUE zurück, wenn das Stellwerk, durch das der Zug vorher gefahren ist, von einem Spieler gesteuert wird, der Zug also nicht nur vom System weitergereicht wurde.

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

ThemaScript Zug-Befehle

Befehl	`wirdgesteuert`
Einschränkung	`(L0) (:B)`
Parameterzahl beim lesen	`0`
Parameterzahl beim setzen	`-`

Gibt TRUE zurück, wenn das Stellwerk, durch das der Zug jetzt fahren soll, von einem Spieler gesteuert wird. Das garantiert allerdings nicht zwingend, dass der Spieler den Zug wirklich steuern wird!

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

ThemaScript Zug-Befehle

Befehl	`altaid`
Einschränkung	`(L0) (:B)`
Parameterzahl beim lesen	`0`
Parameterzahl beim setzen	`-`

Gibt die Stellwerks-ID zurück, durch das der Zug vorher gefahren ist – oder 0 wenn es das 1. Stellwerk ist.

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

ThemaScript Zug-Befehle

Befehl	`neuaid`
Einschränkung	`(L0) (:B)`
Parameterzahl beim lesen	`0`
Parameterzahl beim setzen	`-`

Gibt die Stellwerks-ID zurück, durch das der Zug jetzt fahren wird.

(L0)	Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L)	Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S)	Befehl unterstützt Setzen
(:S)	Befehl darf nur in Klasse Start
(:B)	Befehl darf nur in Klasse Bahnhof
(:SB)	Befehl gibt in den Klassen unterschiedliche Werte zurück

ThemaScript Sim-Befehle

Ausblick

Befehl	`simparam`
Einschränkung	`(S) (:B)`
Parameterzahl beim lesen	`-`
Parameterzahl beim setzen	`2 ?`

(L0) Befehl unterstützt Lesen, es werden dann keine Parameter benötigt
(L) Befehl unterstützt Lesen, es werden Parameter benötigt, Details dann im Text
(S) Befehl unterstützt Setzen
(:S) Befehl darf nur in Klasse Start
(:B) Befehl darf nur in Klasse Bahnhof
(:SB) Befehl gibt in den Klassen unterschiedliche Werte zurück

Dieser Befehl wird die Veränderung einiger Parameter eines Zuges am Simulator erlauben, u.a. der Mindest- und Maximalaufenthaltszeit.

Details stehen jedoch noch nicht fest. Bitte deshalb auch keine Fragen!

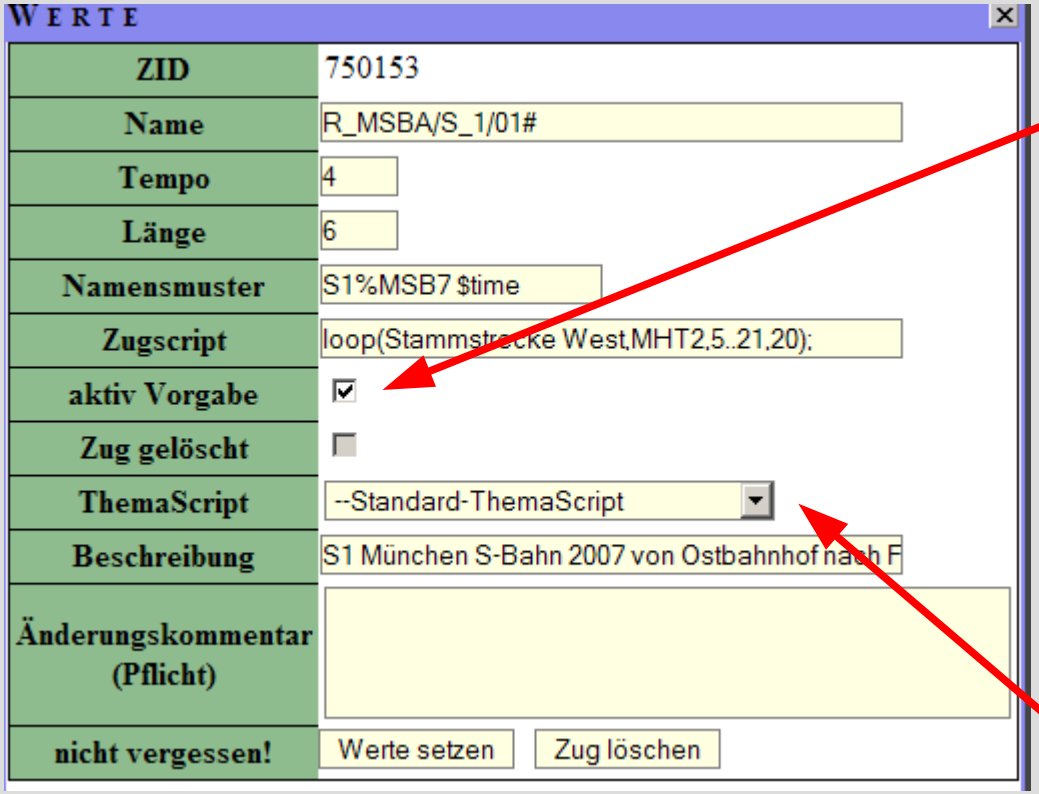
ThemaScript Anwendung

Wie wendet man ThemaScript an?

Einige praktische Beispiele.

Züge nur an bestimmten Tagen

einen Zug abschalten



Ein Zug ist nun sichtbar, wenn dieser Schalter gesetzt ist.

Der Schalter kann auch per ThemaScript gesetzt werden.

```
winter:start:*:750153(N,aktiv);
```

Hier muss nun noch das richtige Script eingetragen werden.

Züge nur an bestimmten Tagen

Grundsätzliches

Um nun auch die restlichen Vorteile des Standard-Scripts zu nutzen, rufen wir das natürlich auch noch auf:

```
winter:start:*:750153(N,aktiv);
*(1,sub);
```

Züge nur an bestimmten Tagen

Grundsätzliches

Alternativ und viel eleganter kann man das ganze auch als Start-Hook speichern und dann beim Zug das normale Standard-Script eingetragen lassen.

Speichern als „ZH-MSBA-stdstarthook“:

```
winter:start:*:750153(N,aktiv);
```

Der Vorteil wäre, dass man im selben Script dann auch gleich andere Züge so aktivieren oder deaktivieren kann.

Und natürlich kann man in beiden Fällen auch alle anderen Funktionen nutzen.

Anfangsverspätung

Ein Beispiel

```
winter:Start(
 $verspaetung,$abstunde,4,/,$abstunde,rnd,+,verspaetung
);
```

- die vorhandene Verspätung wird ausgelesen um sie zu erhöhen
- die letzte Abfahrtzeit dient als Referenz für die Zufallszahl
- Zufallszahl zwischen ¼ letzte Abfahrtzeit und Abfahrtzeit

```
verspaetung = $verspaetung + rnd( $abstunde / 4 , $abstunde )
```

ThemaScript Editor

Der neue Editor

ThemaScript 'editor-test' (246)

[verfügbare Themen zeigen]
[Script Tester öffnen]

Code:

```
#DATE 11.07.2008
*:start:*:786169($laenge,3,+,laenge,$tempo,print,$aktiv,print);
```

Beispiel:

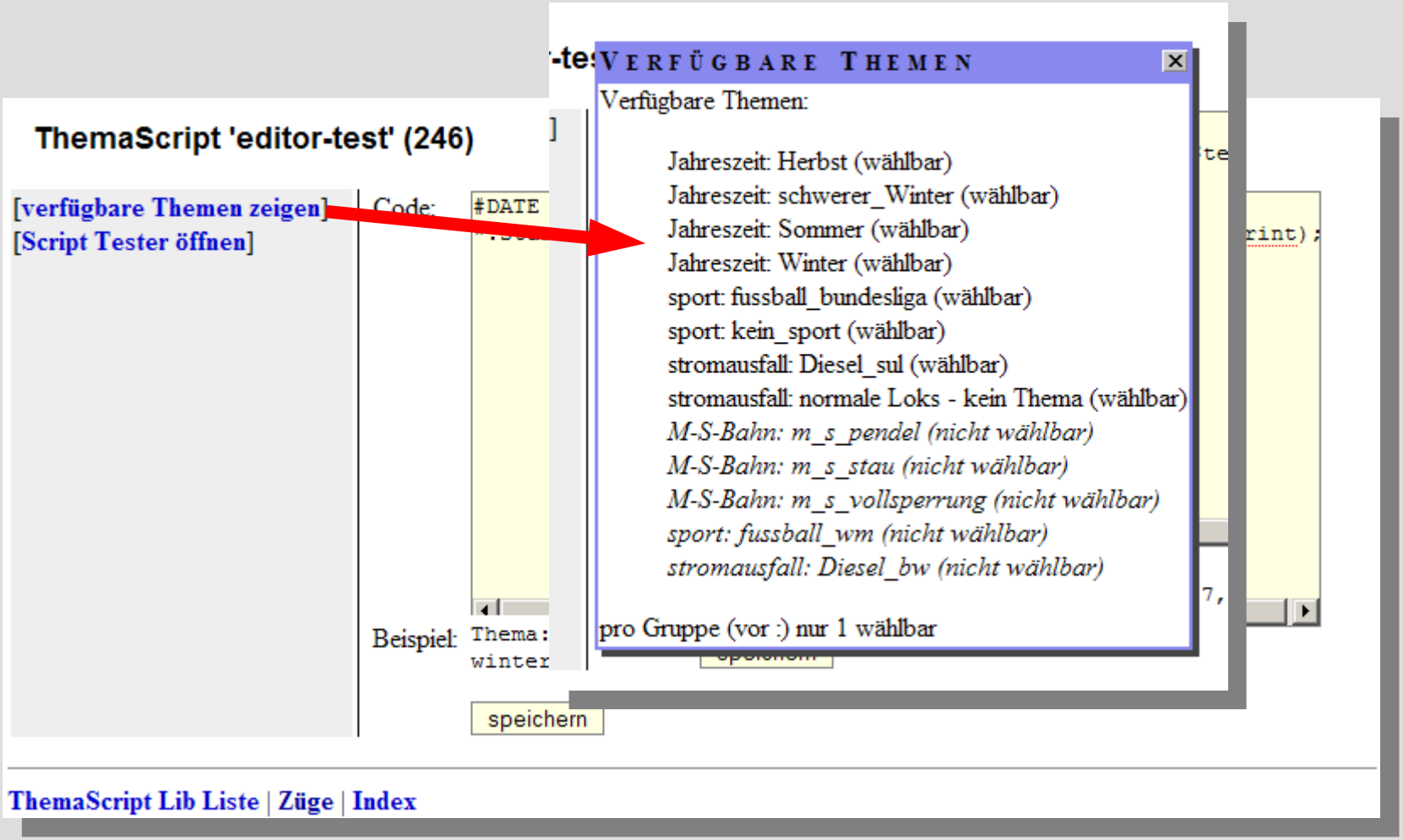
```
Thema:
winter:winter
start(5,20,rnd,verspätung,5,6,7,3,oneof,tempo);
```

speichern

ThemaScript Lib Liste | Züge | Index

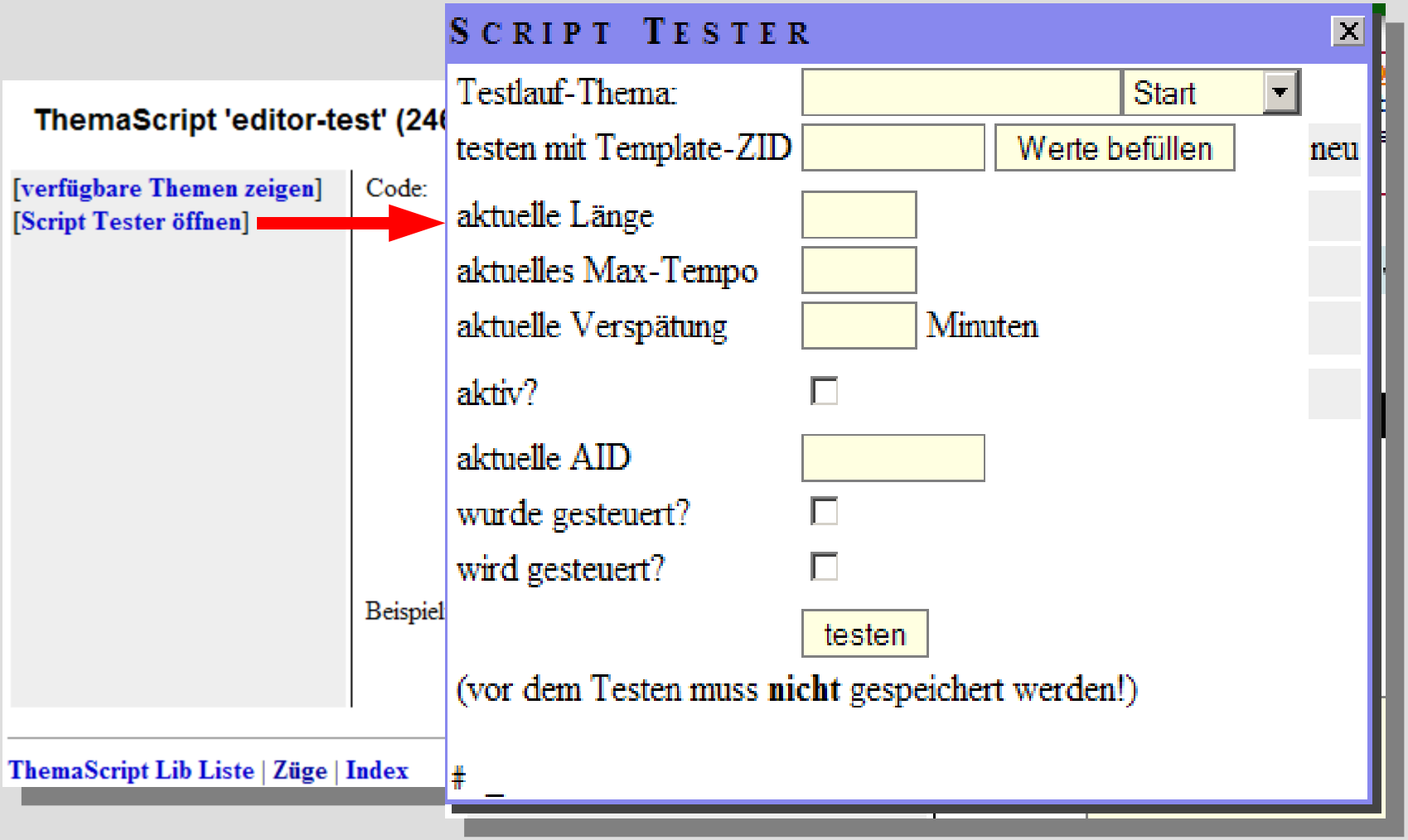
ThemaScript Editor

Menüfunktionen im Editor



ThemaScript Editor

Mit Menüfunktionen im Editor zur Fehlersuche

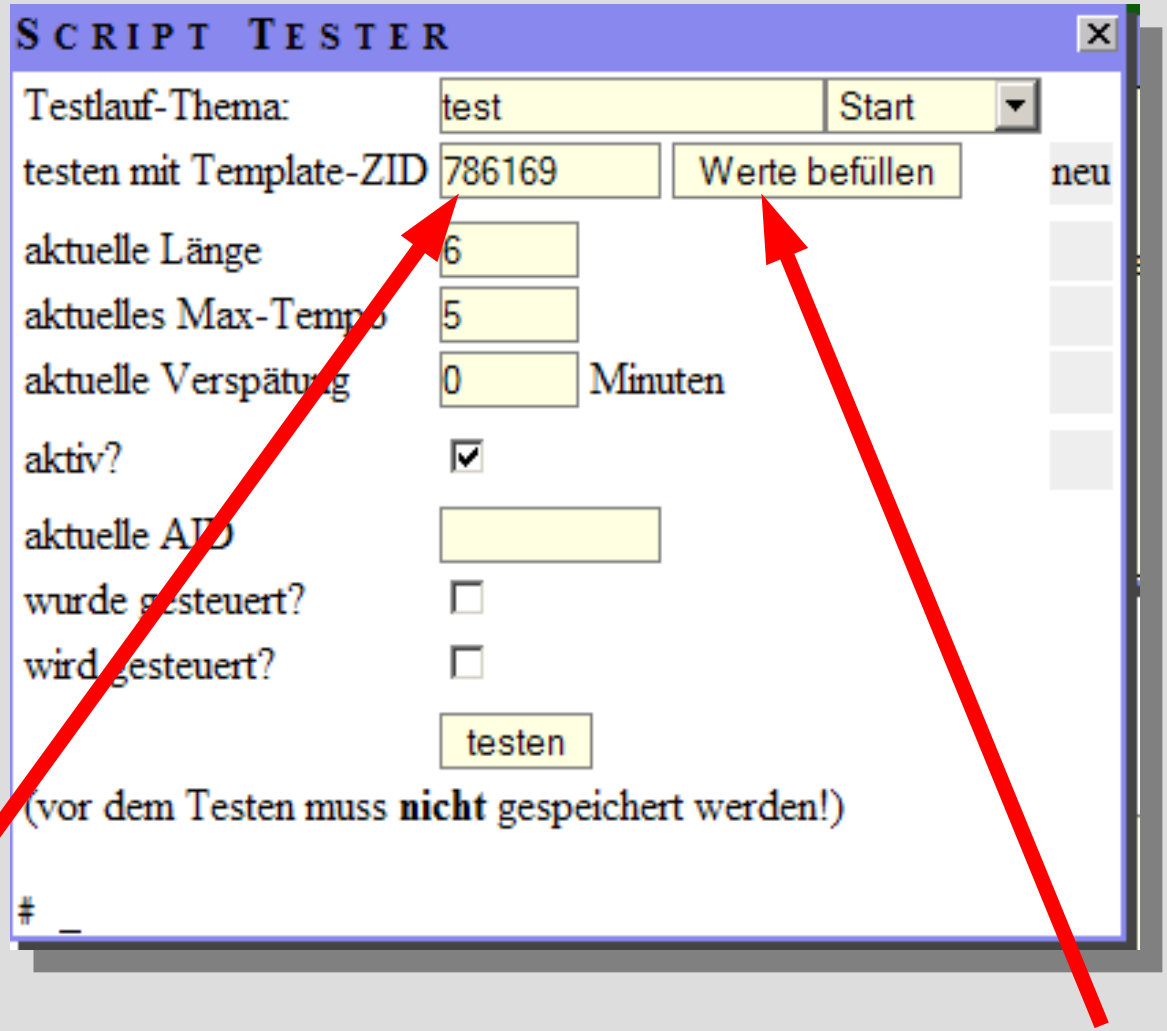


ThemaScript Editor

Fehlersuche im „Script-Tester“

- eine Template ZID eintragen
- auf „Werte befüllen“ klicken

Die Felder Länge, Tempo, Verspätung und Aktiv? Werden mit den Werten aus den Templatedaten gefüllt.



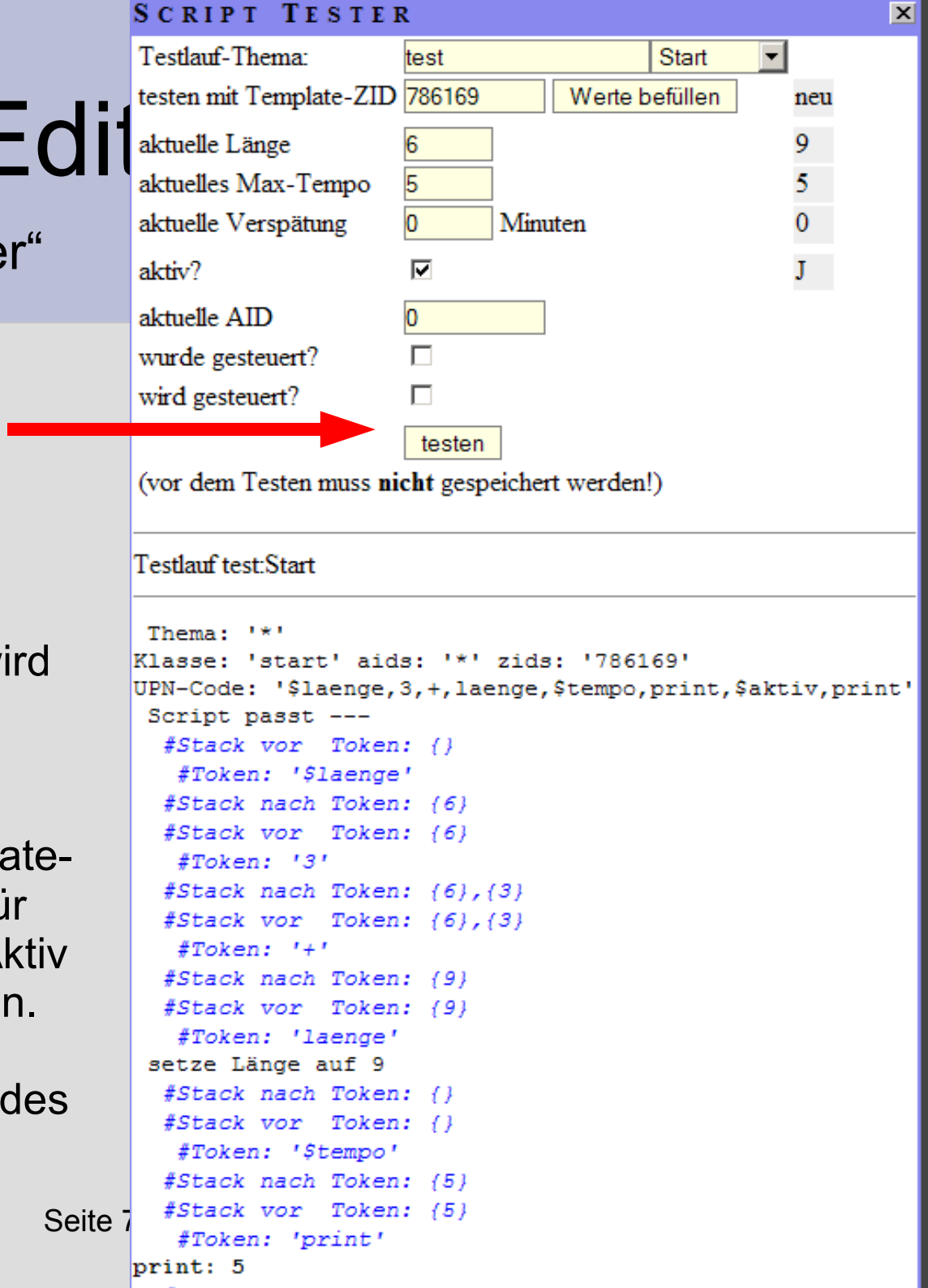
ThemaScript Edit

Fehlersuche im „Script-Tester“

- Ein Thema eintragen
- Ggf. Klasse ändern
- Optional eine AID eintragen
- Optional einen der Werte Länge, Tempo, Verspätung oder Aktiv? Ändern
- Optional wurde gesteuert oder wird gesteuert ändern
- Auf „testen“ klicken

Das Script wird nun für den Template-Zug ***simuliert***. Die neuen Werte für Länge, Tempo, Verspätung oder Aktiv stehen dann hinter den Startwerten.

Unterhalb ist der komplette Trace des Scriptes zu sehen.



Das war's!

Vielen Dank für das Interesse.