


Apple Script.

Eine Einführung.

Autor: Henning Grote (2008)


Was ist ein Skript?

- Ein Skript ist eine Abfolge von Anweisungen
- Skriptsprachen sind Programmiersprachen zur **schnellen** Lösung von Problemen auf **einfache** Weise
- Typische Merkmale einer Skriptsprache:
  - Wird interpretiert
  - Automatische Speicherverwaltung
  - Dynamische Typenverwendung

...und was ist dann Apple Script?

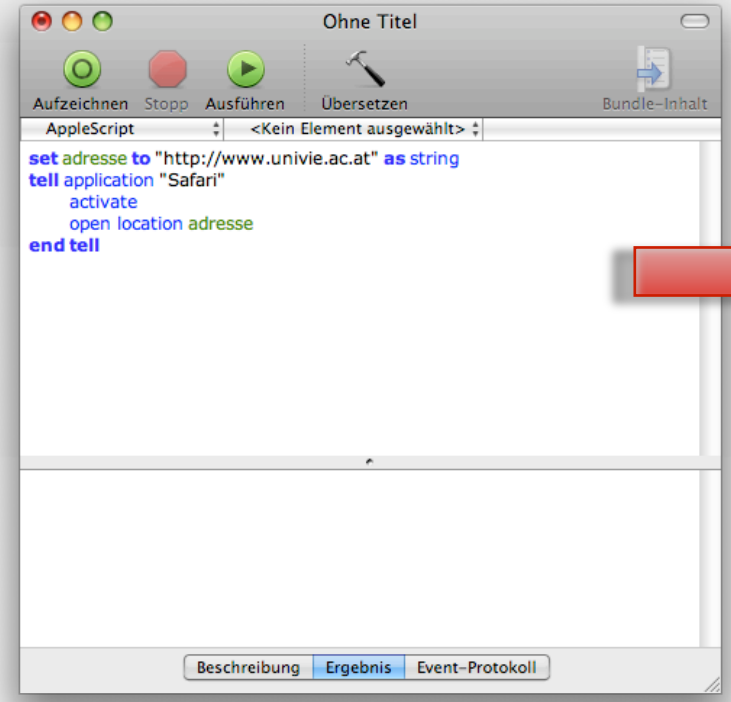
- Apple Script (AS) wurde von Apple zum Vereinfachen von Workflows entwickelt
- Orientiert sich syntaktisch am natürlich gesprochenem Englisch
- Soll für jedermann intuitiv zu benutzen sein
- Ist objektorientiert
- Anwendungen können bequem gesteuert und in ihrer Funktion erweitert werden

Die Geschichte von Apple Script

- Geht auf die multimediale Programmierumgebung *HyperCard* mit der Skriptsprache *HyperTalk* zurück
- Erstmals erschienen 1987, dank seiner intuitiven Art aber seiner Zeit weit vorraus
- Beliebt bei Lern- und Multimediasoftware
- Prominenter auf *HyperTalk* basierter Vertreter:

  Myst (1993) Broderbund

Der Skripteditor



Farbliche Unterscheidung:

Blau: Befehle

Grün: Variablen

Schwarz: Werte

Öffnen via: Finder / Dienste / Skripteditor / Neues Skript erstellen

Datentypen

- Apple Script kennt drei Arten von Daten:
  - Zahlen: **number**
  - Einfachen Text: **string**
  - Boolsche Ausdrücke: **boolean**
- Definition:

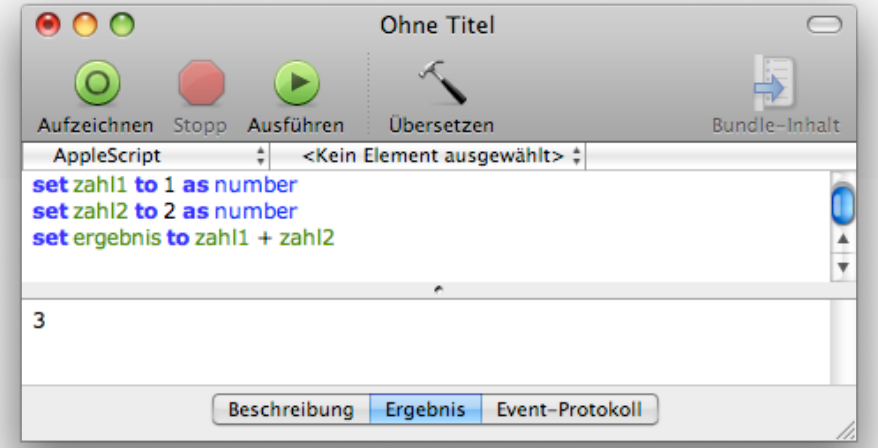
```
set meineZahl to 3.14183 as number
set meinText to "hallo!" as string
set meineEntscheidung to true as boolean
```

```
set meineZahl to 3.14183
set meinText to "hallo!"
set meineEntscheidung to true
```

Möglich, aber mit Vorsicht zu verwenden!

Zahlen sind nicht gern allein.

- Apple Script kennt folgende math. Operatoren:
  - \+ = Addition
  - \- = Subtraktion
  - / = Division
  - * = Multiplikation
  - ^ = Exponentialrechnung („hoch“)

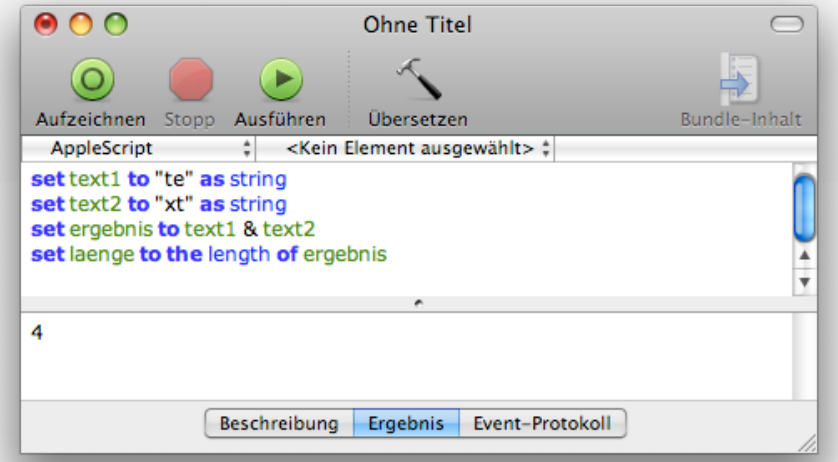


Text

- Auch Text lässt sich unterschiedlich einsetzen

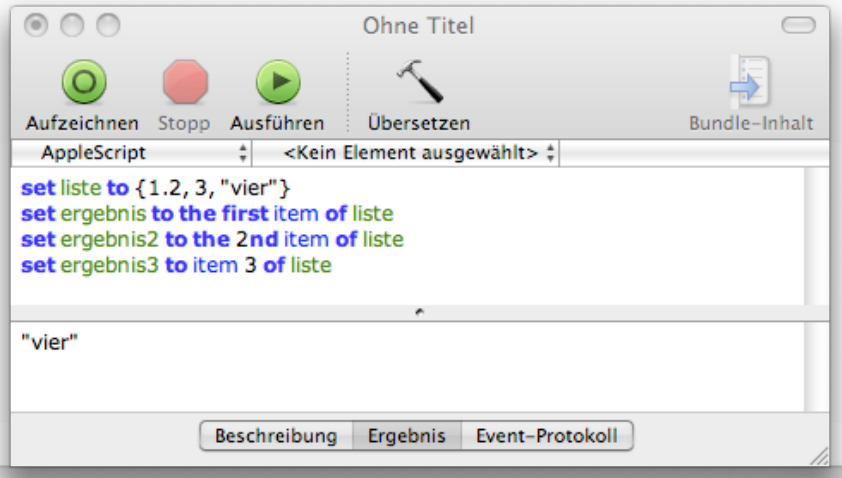
  Beispiele:

  - Verbinden von zwei Strings mit **&**
  - Länge eines Strings bestimmen mit **length**



Listen

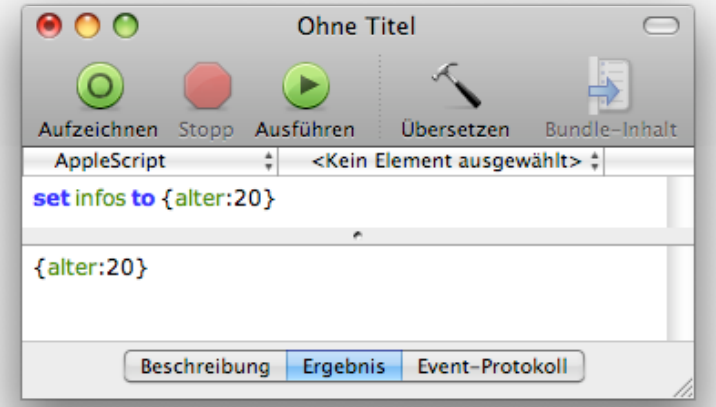
- Listen eignen sich zum Speichern mehrerer Werte
- Sind durch **{...}** zu erkennen
- Werte werden mit Kommata voneinander getrennt: **{1.2,3}**
- Aufruf über **get** liste **/ set ... item of ...**



Datensätze

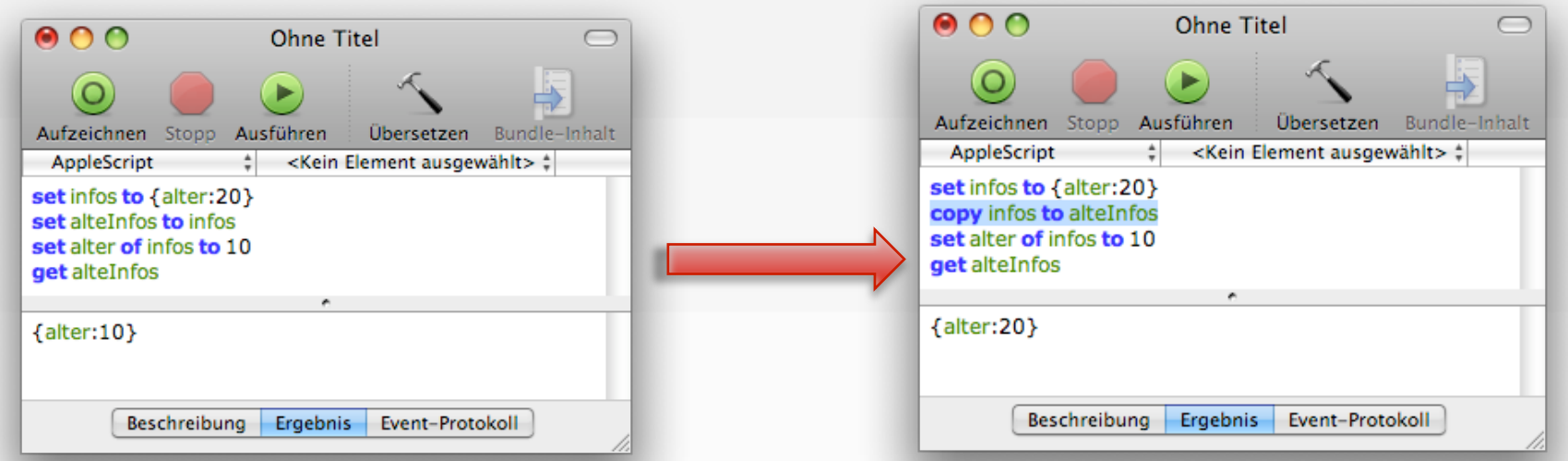
- Wie ich Datensätze richtig setze:
  - Datensätze (records) ähneln einer Liste
  - Bilden Zugehörigkeitspaare
  - Einzelne DS sind nicht änderbar

    -> DS muss neu erstellt werden!
  - Bsp: **set infos to {alter:20}**



Datensätze

- Achtung Falle:



set speichert keine Kopie, sondern nur eine Referenz. Darum **copy** benutzen!

Zeit für eine Pause!

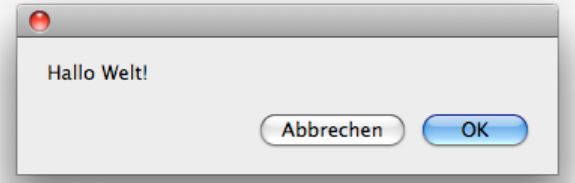
Gibt es Fragen? Feedback?

Da war doch was...

- Wir kennen bisher:
  - Den Editor: Befehle, Variablen, Werte
  - Datentypen: number, string, boolean
  - Operatoren: + - / * ^ &
  - Listen: {1,2,“drei“}
  - Datensätze: {alter:20}

Dialoge

- Dialoge sind Ein- bzw. Ausgabefenster
- Werden mit **display dialog of** eingeleitet
- Sie können beliebigen Text und Schaltflächen enthalten




1 + 1 = ?

- **Aufgabe**: Programmiert ein Skript welches 2 Zahlen miteinander addiert / multipliziert / ... und das Ergebnis in einem Dialog ausgibt!
- **Ziel:** Das zuvor gelernte Wissen anwenden!

Bedingte Anweisungen

- Bedingungen können mit **if ... then ... else** geprüft werden
- Anweisungen stehen in einem **if ... end if** Block

```
set zahl1 to 1 as number
set zahl2 to 2 as number
if zahl1 is greater than zahl2 then
      display dialog "Zahl1 ist größer als Zahl2"
else
      display dialog "Zahl2 ist größer als Zahl1"
end if
```

Vergleichsoperatoren für Zahlen:

=	is (bzw. 'is equal to')
>	is greater than
<	is less than
>=	is greater than or equal to
<=	is less than or equal to

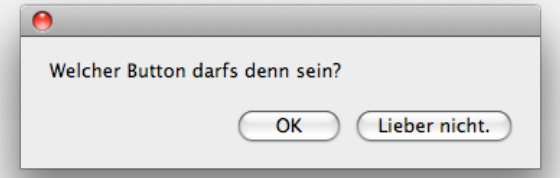
Vergleichsoperatoren für Text:

begins with (bzw. 'starts with')
ends with
is equal to
comes before
comes after
is in
contains

Auch hier gilt: Es können sowohl Symbole als auch englischer Text genutzt werden.

Bedingte Anweisungen und Dialoge

- Abfrage der Buttons über Dummy-Variable




Schleifen

- Schleifen dienen zum wiederholten Ausführen von Anweisungen
- Mehrere Typen:
  - **repeat** x **times** ... **end repeat**
  - **repeat until** bedingung **is** boolean ... **end repeat**
  - **repeat while** bedingung **is** boolean ... **end repeat**
  - **repeat from** start **to** ende ... **end repeat**

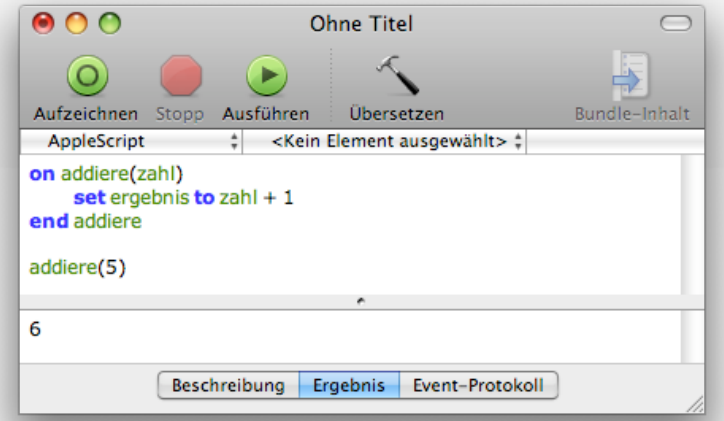

Schleifen

Beispiel: „99 bottles of beer on the wall“

```
set bob to 99 as number
repeat with counter from 1 to 99
      set bottles_of_beer to bob as string
      say bottles_of_beer & " bottles of beer on the wall, " & bottles_of_beer & " bottles of beer."
      set bob to bob - 1
      set bottles_of_beer to bob as string
      say "Take one down and pass it around," & bottles_of_beer & " bottles of beer on the wall."
end repeat
```

Eigene Methoden

- Eigene Methoden können definiert werden
- Auslagern von Funktionalität
- Bessere Lesbarkeit des Skripts
- **on methode(...) ... end methode**




I‘ve got to tell you something...

- **tell application** “App“ dient zum Steuern einer AS fähigen Anwendung
- Alle Anweisungen in einem **tell** ... **end tell** Block werden an die Applikation geleitet

```
tell application "Safari"
	activate
	open location "http://www.apple.de"
	say "Hi, this is Safari speaking. I just opened the apple website
		for you."
end tell
```

Feedback-Runde

- Was habt ihr für Ideen?
- Gibt es Fragen?
- Ausprobieren!!!

Quellen & Literatur

Schulz, D.: *HyperTalk der Ursprung von Apple Script*,
http://www.ratschlag24.com/index.php/hypertalk-der-ursprung-von-Apple Script/ , Stand 04/2008

Altenburg, B.: *Apple Script für Absolute Starter*,
http://www.fischer-bayern.de/as/as4as/AS4AS_g.pdf , Stand 04/2008

Apple Inc., *Umgekehrte Entwicklungshilfe*, in Fallstudie der Universität Zürich,
http://www.apple.com/chde/education/profiles/softwaretankstelle , Stand 04/2008