Variablen

Die vier Variablentypen

```
stringVar = "Blindtext"
numVar = 1                  boolVar = true
arrayVar = {7,8,9}
```

Globale und lokale Variablen

```
globalVar = "Blindtext"
local lokalVar = "Blindtext"
```

Variablen-Wert löschen

```
meineVar=nil
```

Multidimensionale Arrays

```
multiArrayVar[1,2] = "Blindtext"
```

Array-Größe

```
table.getn(ArrayVar)      -- Variante 1
#ArrayVar                 -- Variante 2
```

Array erweitern

```
table.insert(ArrayVar, line)
```

Hashes

```
hashVar = { lua_1 = 10, lua_2 = 24, }
hashVar["lua_1"] = 10
```

Array oder Hash sortieren

```
table.sort(a, b)
```

Operatoren

Vergleichsoperatoren

```
a==b, a~=b, a<b, a<=b, a>b, a>=b
```

Strings

Zeichenketten ersetzen

```
a = "mein string"
b = string.gsub(a, "mein", "dein")
```

Strings verbinden

```
meinString = "Hallo ".."Welt"
```

Maskieren

```
meinString = "Er sagt: \"Stop!\""
```

Etwas in einen String konvertieren

```
meinString = tostring(meineZahl)
```

Stringlänge

```
string.len(stringVar)
```

In Klein- bzw. Großbuchstaben umwandeln

```
string.lower(stringVar)
string.upper(stringVar)
```

Substrings

```
stringVar = string.sub(stringvar,2,2)
Erstes Zeichen hat die Position 1, letztes
Zeichen -1.
```

Strings formatieren

```
string.format("pi=%.4f", PI) -- pi=3.1416
string.format("%02d“,d) -- 05
```

Einfaches Pattern matching

```
stringVar = "Hallo Welt"
i, j = string.find(stringVar, "Hallo")
print(i,j) -- 1,5
```

Ersetzen

```
s = string.gsub("Weld", "d", "t")
print(s) -- Welt
```

Zahlen

Etwas in eine Zahl konvertieren

```
meineZahl = tonumber(stringVar)
if (n==nil) then      -- ...keine Zahl
```

Maskieren

```
stringVar = "Er sagt: \"Stop!\""
```

Datum und Uhrzeit

Zeitpunkt in Epochensekunden umwandeln

```
os.time{year=1970, month=1, day=1, hour=0}
```

Epochensekunden in Zeitpunkt umwandeln

```
hashVar = os.date("*t", 906000490)
Erzeugt {year = 1998, month = 9, day = 16,
yday = 259, wday = 4, hour = 23, min = 48,
sec = 10, isdst = false}
```

Datums-/Zeitangabe erzeugen

```
print(os.date("Today is %A"))
-- Today is Tuesday
print(os.date("%x", 906000490))
-- 09/16/1998
%a      abbreviated weekday name (e.g., Wed)
%A      full weekday name (e.g., Wednesday)
%b      abbreviated month name (e.g., Sep)
%B      full month name (e.g., September)
%c      date and time (e.g., 09/16/98 23:48:10)
%d      day of the month (16) [01-31]
%H      hour, using a 24-hour clock (23) [00-23]
%I      hour, using a 12-hour clock (11) [01-12]
%M      minute (48) [00-59]
%m      month (09) [01-12]
%p      either "am" or "pm" (pm)
%S      second (10) [00-61]
%w      weekday (3) [0-6 = Sunday-Saturday]
%x      date (e.g., 09/16/98)
%X      time (e.g., 23:48:10)
%Y      full year (1998)
%y      two-digit year (98) [00-99]
```

Kontrollstrukturen

for do

```
for i=1,10 do
       -- irgendetwas
       if (bedingung) then break end
end
```

while do

```
while (a[i]) do i = i+1 end
```


Verfasser: Rüdiger Reichelt – reichelt@subreality.de
Quellen: lua.org, mkshb.de, siio.de, fibarouk.co.uk

Subreality.de

if then else

```
if (wert==1) then
      elseif (wert==2) then      -- ...
      else                       -- ...
end
```

Repeat

```
repeat
      i = i+1
until (i>10)
```

Funktionen

Funktionen

```
function addiere(zahl1,zahl2)
      return irgend_etwas
end
```

Fibaro

Nachricht an die Debug-Konsole

```
fibaro:debug("Nachricht")
Kann mit CSS (<span>) formatiert werden.
```

Einfache Push-Nachricht an ein Smartphone

```
fibaro:call(ID, "sendPush", "Nachricht")
```

Funktionaler Push-Nachricht an ein Smartphone

```
api.post('/mobile/push',
{["mobileDevices"]={g_id},
["message"]=push_nachricht,
["title"]=push_betreff,
["category"]='RUN_CANCEL',
["data"]={["sceneId"]=s_id}})
-- Zeilenumbruch mit \n
```

Pause

```
fibaro:sleep(Anzahl_Millisekunden)
```

Ein- und ausschalten / Öffnen und schließen

```
fibaro:call(ID,“turnOn“)  bzw. “open“
fibaro:call(ID,“turnOff“) bzw. “close“
```

Wert setzen und auslesen

```
fibaro:call(ID,“setValue“,15)
-- Dimmer auf 15% setzen
fibaro:getValue(id, "Value")
-- Liefert auch Zahlen als String
```

Wert auslesen

```
fibaro:getValue(id, "value")
-- Liefert auch Zahlen als String
```

Gerätename, Raum und Raumname ermitteln

```
fibaro:getName(g_id)
fibaro:getRoomID(g_id)
fibaro:getRoomName(r_id)
```

Gerätetyp ermitteln

```
theType = fibaro:getType(id)
```

Bereich ermitteln

```
bereich = fibaro:getSectionID(id)
if (section == 0) then
      fibaro:debug('Kein Bereich.')
else
      fibaro:debug('Nr.' ..bereich.. '.')
end
```

Geräteeigenschaft als Trigger

```
%% properties
59 value
-- Der Sensor mit Id 59 wird überwacht.
```

Szene automatisch starten und regelmäßig ausführen

```
%% autostart
setTimeout(function, millisekunden)
```

Trigger ermitteln

```
trigger = fibaro:getSourceTrigger()
```

Entfernung berechnen

```
-- Ort des Benutzers mit ID 12
userLocation=fibaro:getValue(12,'Location')
-- Ort der HC2
Home="52.4325295140701;16.8450629997253"

-- Entfernung berechnen und ausgeben
result=fibaro:calculateDistance(
userLocation, Home)
fibaro:debug('Entfernung:'..result..'m.')
```

Globale Fibaro-Variable managen

```
-- Wert und letzte Änderung von isNight
value, modificationTime =
fibaro:getGlobal('isNight')
-- modificationTime ist optional.
fibaro:setGlobal('isNight', 1) -- setzen
```

Zeitpunkt der letzten Zustandsänderung

```
-- Zeitpunkt für Id 11
lastModified =
fibaro:getModificationTime(11, 'value')

if ((os.time()-lastModified)>=10) then
      fibaro:debug('10 oder mehr Sek.')
end
```

Szenen managen

```
-- Szene 3 aktiv? Dann Szene 5 aktivieren.
if (fibaro:isSceneEnabled(3)) then
      fibaro:setSceneEnabled(5, true)
end
fibaro:startScene(id) -- Starten
fibaro:abort()        -- aktuelle Instanz
fibaro:killScenes(id) -- Alle Instanzen
```

Batterie von Sensoren managen

```
sArray = fibaro:getDevicesId({interfaces
={"battery"}, visible = true, enabled =
true})
batterie = fibaro:get(sArray[i],
'batteryLevel')
```


Verfasser: Rüdiger Reichelt – reichelt@subreality.de
Quellen: lua.org, mkshb.de, siio.de, fibarouk.co.uk

Subreality.de