41.3 LUA Grundlagen - Texte

Autor: Goetz
Quelle: Mein EEP-Forum

In einem Programm oder Skript werden auch Texte benötigt. Lua muss unterscheiden können, was als Text gemeint ist und was beispielsweise eine Variable oder ein Befehl sein soll. Ein Computer kann nicht mitdenken. Der kann eigentlich überhaupt nicht denken. Ein Computer ist strunzdumm und tut nur stupide das, was man ihm sagt.

Wenn ich also, wie im ersten Beispiel, schreiben kann
John = 3
und damit festlege, dass eine Variable den Namen John bekommen soll und ich in dieser Variablen den Wert 3 speichern möchte, dann bedeutet das folgendes:
Die Entwickler von Lua haben festgelegt, dass ein Wort im Programm, welches in der Sammlung der Befehle von Lua nicht vorkommt, als Variable interpretiert wird. Völlig egal, wo im Skript dieses Wort auftaucht.

Die Funktion
print(**Modellbahn**)
schreibt also nicht das Wort Modellbahn ins Ausgabefenster.
Sondern sie schaut nach, ob in der Variablen **Modellbahn** ein Wert gespeichert wurde.
Und diesen Wert schreibt sie dann ins Ausgabefenster.

Falls dem Wort Modellbahn nie ein Wert zugewiesen wurde, schreibt die Funktion **print**() das Wort **nil** ins Ausgabefenster.
nil ist Englisch und meint nix oder nüscht.

Damit man Texte in einem Skript verwenden und beispielsweise ins Ausgabefenster schreiben kann, muss man sie als Text kennzeichnen. Dafür verwendet man die Anführungsstriche.
”Modellbahn”
zeigt Lua, dass man das Wort Modellbahn als Text benutzen möchte und nicht als Variable.

Will ich das Wort Modellbahn ins Ausgabefenster schreiben, dann lautet der richtige Code dafür
print(”Modellbahn”)

Alles ganz einfach, oder?

Von wegen!
Hier ist gar nichts einfach. Im Gegenteil.
Lasst euch nicht davon täuschen, dass das so einfach aussieht.

Denn wenn man sich das mal ganz genau ansieht, dann erkennt man folgendes:
Es steht ein Anführungszeichen **vor** dem Text und eins **dahinter**.
Wir betrachten das im Ganzen. Weil unser Gehirn das kann.
Für uns steht das Wort **in** Anführungszeichen.

So etwas geniales kann ein Computer nicht.
Der muss das Skript Buchstabe für Buchstabe und Zeichen für Zeichen lesen.

Lua sieht an einer Stelle plötzlich ein Anführungszeichen und weiß damit:
Ab hier kommt Text
Und wenn Lua dann wieder ein Anführungszeichen sieht, dann weiß es:
Hier hört der Text wieder auf.

Das **selbe** Anführungszeichen bedeutet also einmal Text-**Anfang** und einmal Text-**Ende**.
Je nachdem, ob vorher schon ein Anführungszeichen stand oder nicht.

Wer schon einmal Aufnahmen mit einer Videokamera gemacht hat, der kennt wahrscheinlich folgendes Problem:
Mit der Start-Stopp Taste kann man die Aufnahme abwechselnd starten und stoppen. Und wehe, man kommt dabei aus dem Rhythmus. Dann hat man später nichts von dem aufgenommen, was man wollte. Sondern nur den Himmel, die eigenen Füße, das Innenleben der Jackentasche … Je nachdem, was man mit der Videokamera gemacht hat, als man dachte, sie sei auf Pause.

Wir kämpfen bei der Videokamera und bei Texten in Programmier- und Skriptsprachen mit dem selben Problem:
Ein einziger Befehl - der Druck auf die Start-Stopp Taste bzw. das Anführungszeichen - hat zwei gegensätzliche Bedeutungen. Er steht **abwechselnd** für **An** und **Aus**.
Und man muss höllisch aufpassen, dass man nicht aus dem Tritt kommt.

Ein einziges Anführungszeichen zu viel oder zu wenig im Skript kann alles durcheinander bringen!

Okay, zurück zu Texten:
Ich kann also schreiben
print(”Lua rennt“)
und im Ausgabefenster steht dann
Lua rennt
und zwar ohne die Anführungszeichen!
Denn die gehören nicht zum Text, sondern zeigen Lua, dass alles, was dazwischen steht, als Text behandelt werden soll.

Man kann Texte auch in Variablen speichern. Das ist sehr praktisch, wenn man sie öfter benötigt.
Wenn ich schreibe
John = ”Lua rennt”
dann merke ich mir den Text Lua rennt in der Variablen **John**.

print(**John**)
gibt dann im Ausgabefenster Lua rennt aus.

Wenn ich hingegen
print(”John”)
schreibe, dann wird das Wort John im Ausgabefenster erscheinen und nicht der Text, den ich in der Variablen **John** gespeichert habe.

Und was passiert, wenn ich
print(**john**)
schreibe?

Dann gibt Lua im Ausgabefenster folgenden Text aus:
nil: 0x0000000000000000

Denn der Variablen **john** - mit **kleinem** j am Anfang - habe ich nie einen Wert zugewiesen.
Da steht nix drin. Und das englische Wort für nix ist nil

Man muss bei Programmier- und Skriptsprachen peinlich genau aufpassen, was man schreibt.
Ein einziges falsches Zeichen bringt alles durcheinander.

Übrigens wird eine Software wie EEP genau so programmiert wie ihr es jetzt mit Lua kennen lernt. Nur, dass da eine unvorstellbare Menge an Code Zeilen nötig ist. Für jeden Pups, den das Programm machen soll, muss man zig Zeilen Code schreiben.
Dafür, dass etwas passiert, wenn man auf einen Button drückt.
Dafür, dass dieser Button überhaupt auf dem Bildschirm erscheint.
Dafür, dass auf diesem Button ein kleines Bildchen zu sehen ist.
Dafür, dass man ein Gleis anfassen kann.
Dafür dass sich die Darstellung dieses Gleises bewegt, wenn man es verschiebt …

Und ein einziges, falsches Zeichen in diesem Code bringt die Dinge durcheinander.
Das nennt man dann einen Bug.
Vielleicht könnt ihr euch nun besser vorstellen, wie schwer es ist einen Bug zu finden?
Vor allem dann, wenn die Bugmeldung nur lautet ”Das Programm stürzt dauernd ab!“ und der Programmierer jetzt theoretisch einen Text, der so lang wie Goethes Faust ist, nach einem einzigen fehlenden oder falschen Zeichen durchsuchen müsste, um die Ursache zu finden?

und weiter im Text
bzw. im Thema ”Text“

Benny hat mich gerade in einer PN darauf hingewiesen, dass ich manchmal die falschen Anführungszeichen benutze.
Das ist auch so ein Stolperstein, der einem den schönsten Code versauen kann.

Es gibt nämlich eine Reihe verschiedener Anführungszeichen.

solche zum Beispiel: „Für deutsche Texte, die Anführungszeichen unten und oben haben sollten.“
oder die sehr ähnlichen, englischen bzw. amerikanischen: ”Die Briten haben vorne und hinten die Anführungsstriche oben.“
und dann gibt es noch diese ganz einfachen: "Die sind gerade, nicht geschwungen und stehen ebenfalls vorne und hinten oben."

Das sind alles verschiedene Zeichen für Lua. So verschieden wie ! und % und #
Und nur die einfachen, geraden Anführungszeichen aus dem dritten Beispiel in der Liste akzeptiert Lua als Kennung für einen Text.

Wenn man seinen Code im vorgesehenen Skript Fenster schreibt, dann bekommt man ganz automatisch die richtigen Anführungszeichen. Aber mein Tutorial habe ich mit einem Schreibprogramm geschrieben und dann ins Posting eingefügt. Und die meisten Schreibprogramme gehen heute davon aus, dass man einen ”schönen Text“ schreiben möchte. Deshalb bekommt man die ”schönen“ Anführungszeichen.

Wenn ich also mit einem Programm wie Wordpad meinen Lua Code schreiben und dann mittels Kopieren und Einfügen in das Skript einfügen würde, dann hätte ich die falschen Anführungszeichen. Text würde

dann von Lua nicht als Text erkannt und ich bekäme ”Müll“.

Mit anderen Worten:

print("Lua rennt") funktioniert
print(„Lua rennt“) funktioniert nicht!
print(”Lua rennt”) funktioniert auch nicht!

und man muss sehr genau hinschauen, um den Unterschied zu erkennen.