

Auto-Complete and Live Error in LiveCode 8.2 – a first look

by hauke | Sep 13, 2017 | LiveCode tools and tips, | 2 comments

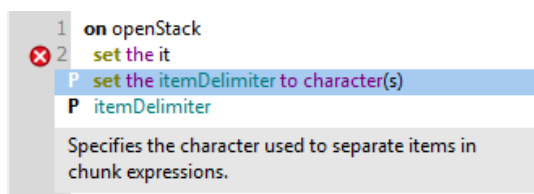
With the recently released Developer Preview (dp1) of LiveCode 8.2, a new major feature is included in the script editor of LiveCode. For years hoped and longed for by many, there is now a comprehensive code completion (code completion) including editable snippets, which can greatly simplify the programmers' lives, further increase the efficiency of the workflow and now places the LiveCode editor in a number with other significant programming editors. And that's not all: In the course of the implementation of these new possibilities, it was also possible to ensure that a context-based live check of the code can now be carried out during the input – called "Live Errors". This is a major development step – more important for the everyday life of programming than many other features that have recently been integrated into LiveCode.

I took a closer look at the first version. It is the dp1 – a preview version that should not be used as the main development system, as there are still some bugs included here. Nevertheless, if you want to know where the trip is going, you should definitely download the version and try these features!

How does auto-completion work in LiveCode?

Completion:

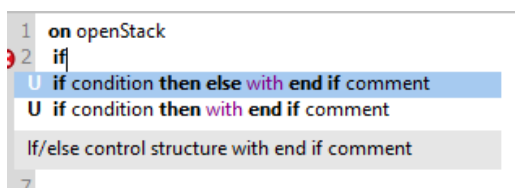
While you enter characters into the editor, all possible completions of the command appear immediately in a list below the input line. The more characters you enter, the more precise the list will be. Instead of continuing to type, you can now immediately select the desired command with the arrow keys and at the same time receive a short description of what the respective command or keyword in LiveCode does.



Here "set the it" is immediately proposed "set the itemDelimiter to ..."

The displayed suggestion with the tab key is selected and inserted. And not only the started word is completed, but usually a complete code snippet with active elements is inserted, which can then be easily and cleverly adjusted.

For example, when I enter the word "if", this immediately appears here:



The two possible variants of the use of "if" are proposed as snippet. Let's assume I want to insert a complete if / else structure, then I just press the tab key, and the code snippet is in the editor.

```

on openStack
  if condition then
    -- true code
  else # condition
    -- false code
  end if # condition
end openStack

```

The word “condition” is a placeholder for the condition I want to ask in my if structure. If I now simply press the condition without pressing any further keys, it is written into all three grayed areas.

```

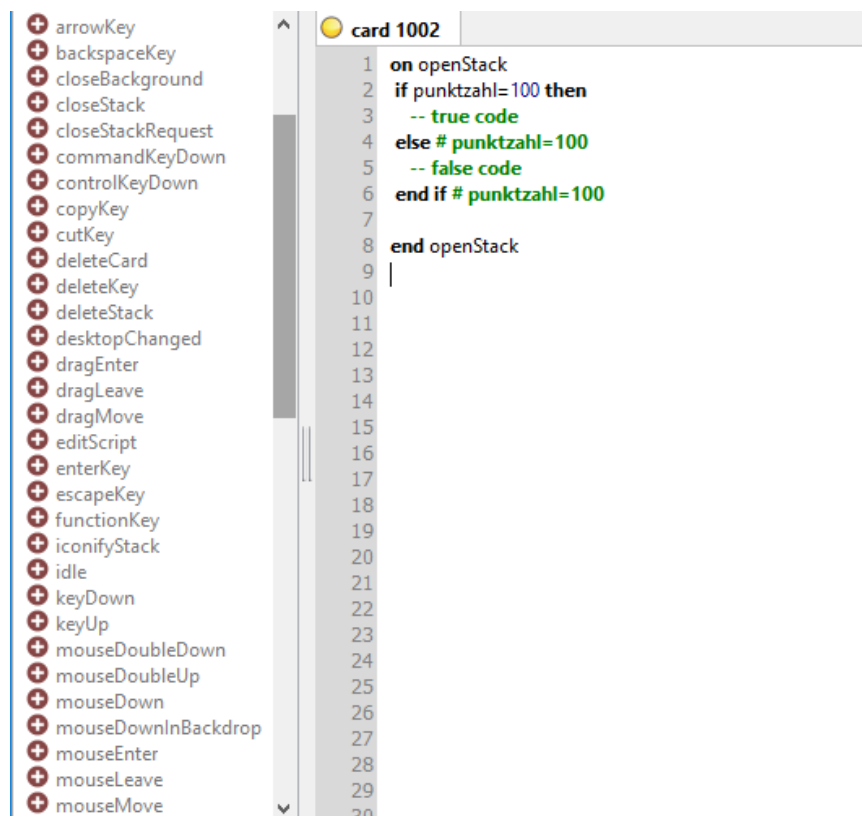
on openStack
  if punktzahl=100 then
    -- true code
  else # punktzahl=100
    -- false code
  end if # punktzahl=100
end openStack

```

The condition “point number of 100” is now automatically available both as a condition according to the “if” and as a comment behind the “else” and the “end if”. You don’t have to explain to any programmer why this is tremendously practical. In complex scripts with numerous nested structures, understanding the code is considerably facilitated when you see at a glance where the else or the end if refers to, which is somewhere in line 325! And commenting on this oneself – let’s be honest – is often too arduous in the heat of the moment. How good that it is now happening automatically!

With the tab key you can now jump from placeholder to placeholder and enter the respective code. In this way, recurring code patterns can be created faster, easier and more securely, because in finished code sippets there are no typos errors that can often make life difficult.

By the way, event handling can now also be inserted directly by clicking, because all usual event queries for the respective object are now in the script editor in the left column and can be appended to the skill script directly with a mouse click.



The list in the left column is even more extensive and can be scrolled. For example, I click on

“mouseDoubleClick” immediately add the following snippet to the back of the code:

```
on mouseDoubleClick pButtonNumber  
|  
end mouseDoubleClick
```

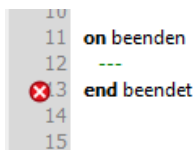
This not only saves typing work, but it can also help beginners in particular enormously by founding all available events for the object to which the script refers in the list and can be inserted directly. At the same time, one learns how this or that command is used correctly without looking up.

For all possible program structures, there are already predefined clever and partly interactive snippets – and therefore not enough: If you want, you can define your own snippets and add them to their editor. This allows everyone to create their individual programming habits according to the perfect workflow for the code. The editor still had a few bugs in the preview version – but this will certainly be fixed soon. The possibilities are great!

Auto completion is not only a feature that saves typing, but it also provides the writing of code on a completely new basis – typos are excluded, structures are immediately correct and a kind of help and instructions for programming in LiveCode is already included! For beginners a huge help, for professionals extremely practical and time-saving.

Live error:

The side effect of auto completion is not to be underestimated. From now on, the code is already pre-compiled and checked continuously when typing – which means that not only syntax errors (i.e. misspelled commands, etc.) are marked as incorrect, but also errors related to the context. For example, if I double declare a variable or finish a function with the wrong name or something else, which is syntactically correct as a single line, but in context makes an error, then the editor now immediately shows me as an error, without having to compile the script beforehand.



This error will be shown to me “live”. As a result, I save a lot of time, because I immediately see when typing that something is wrong before I continue writing and can eliminate so many mistakes that might later be caused by headaches.

Conclusion: The new editor functions in LiveCode are much more than just a small entry helper. They change the way we program with LiveCode, save time and a lot of look up, secure our code and help to avoid numerous errors in advance. I look forward to the first stable version with these features!

Note: LiveCode 8.2.0 can be downloaded here:

<http://downloads.livecode.com/livecode/>

2 comments



Torsten February 14, 2017 at 11:51

Already found a small bug: if you browse the list of possible commands, you will not get 1st shown how long the list is and 2. the view ends when the last option has been narrowed.

Here it would be better if at least the list were always visible without automatically

switching off.

[Sign up for answers](#)



hauke prelease announced on September 14, 2017 at 12:44

Of course, you can report the bugs to LiveCode. I think they will be fixed in the upcoming versions.

[Sign up for answers](#)

<input type="text"/>	Suche
----------------------	-------

Latest posts

Create real web apps

Create Android apps in Windows

Programming your own apps – the new version has been released

Auto-Complete and Live Error in LiveCode 8.2 – a first look

What can you do with LiveCode? (And what not?)

Archives

May 2021

February 2020

Nov 2019

September 2017

February 2017

January 2017

November 2016



topblogs



Impressum

Hauke Fehr 2020