

LiveCode Cheat Sheet

Comments

Comments allow you to add explanations and annotations to your code.

```
-- these  
# are  
// all  
/* commented  
out */
```

Variables

Variables are used to store information, the stored value can be changed or accessed when you need it.

```
local tVar  
put "str" into tVar  
put 1 into tVar  
  
put "val" into  
tVar["key"]
```

Constants

Constants store a value that is defined at the point of declaration and never changes.

```
constant kFoo = 15
```

Control Structures

Control structures are used to control what code is executed and how many times.

Introduction

(<https://livecode.com/docs/9-5-0/introduction/>)

Lessons

(<https://livecode.com/docs/9-5-0/lessons/>)

FAQ (<https://livecode.com/docs/9-5-0/faq/>)

Language

(<https://livecode.com/docs/9-5-0/language/>)

LiveCode Cheat Sheet

(<https://livecode.com/docs/9-5-0/language/livecode-cheat-sheet/>)

LiveCode Builder Cheat Sheet

(<https://livecode.com/docs/9-5-0/language/livecode-builder-cheat-sheet/>)

LiveCode Builder - LiveCode Cheat Sheet (<https://livecode.com/docs/9-5-0/language/livecode-builder-livecode-cheat-sheet/>)

LiveCode Script

(<https://livecode.com/docs/9-5-0/language/livecode-script/>)

Education Curriculum

(<https://livecode.com/docs/9-5-0/education-curriculum/>)

Deployment

(<https://livecode.com/docs/9-5-0/deployment/>)

Components

(<https://livecode.com/docs/9-5-0/>)

Comments

Variables

Constants

Control Structures

Operators

String Processing

Array Processing

Sorting

Files & Processes

User Input / Notification

Custom Handlers

Event Handlers

0/components/)

Tooling

(<https://livecode.com/docs/9-5-0/tooling/>)

Core Concepts

(<https://livecode.com/docs/9-5-0/core-concepts/>)

Language Comparison

(<https://livecode.com/docs/9-5-0/language-comparison/>)

Extending LiveCode

(<https://livecode.com/docs/9-5-0/extending-livecode/>)

Whats New?

(<https://livecode.com/docs/9-5-0/whats-new/>)

```
repeat for each  
char tChar in tVar  
end repeat
```

```
repeat 10  
end repeat
```

```
repeat with x = 1  
to 10  
end repeat
```

```
repeat while x > 1  
    subtract 1  
from x  
end repeat
```

```
if true then ...  
else ...
```

```
if tVar then  
else if tOther  
then  
else  
end if
```

```
switch tVar  
    case "a"  
        break  
    default  
        break  
end switch
```

Operators

Operators are ways of combining values such as boolean values, numbers or strings, to produce other values.

```
// Logical
true and false is
false
true or false is
true
not false is true
```

```
// String
"foo" & "bar" is
"foobar"
"foo" && "bar" is
"foo bar"
"string" begins
with "st"
"string" ends with
"g"
```

```
// Chunks
char 5 of "string"
is "n"
item 3 of "a,b,c"
is "c"
word 1 of "hi
there" is "hi"
line 2 of "a" &
return & "b" is
"b"
```

```
// Compound
chunks
char 1 of item 1 of
line 1 of "a,b,c"
is "a"
```

String Processing

These examples show how string values can be manipulated.

```
// General
put "a" before tVar
delete char 1 of
tVar
replace "_" with
 "-" in tVar

// Regex
matchText("1", "[0-9]", tN) is
true
tN is 1

filter lines of
tVar with regex
pattern tPattern
```

Array Processing

These examples show how array values can be manipulated.

```
// Split / combine  
put "a,b,c" into  
tVar  
split tVar by ","  
tVar[2] is "b"  
combine tVar with  
","  
tVar is "a,b,c"
```

```
// Iteration  
repeat for each  
key tKey in tArray  
-- Do something  
with tArray[tKey]  
end repeat
```

```
repeat for each  
element tElement in  
tArray  
end repeat
```

```
// Length  
the number of  
elements in tArray
```

Sorting

These examples show how
to sort items and lists.

```

local tList
put "5,2,3,1,4"
into tList
sort items of tList
ascending numeric
-> tList is
"1,2,3,4,5"
sort items of tList
descending numeric
-> tList is
"5,4,3,2,1"

local tData
put "6,1:8,3:2,2"
into tData
set the
lineDelimiter to
"."
sort lines of tData
ascending numeric
by item 2 of each
-> tData is
"6,1:2,2:8,3"

```

Files & Processes

These examples show how to read from and write to files and processes.

```

get url("file:/" &
tPath)
put "" into
url("file:/" &
tPath)

open process tProc
read from process
tProc for 5
close process tProc

```

User Input / Notification

These examples show how to pop up information dialogs, or prompts for user input.

```
ask "What is your  
name?"  
put it into tName  
  
answer "Something"
```

Custom Handlers

A custom handler is a function or command that you define yourself.

```
function foo pParam  
end foo  
// get foo(tVar)  
  
command bar pParam  
end bar  
// bar 5
```

Event Handlers

An event handler is a handler that is triggered when an event occurs, such as the use of the mouse or keyboard.


```
// Mouse
on mouseUp pButton
end mouseUp

on mouseDown
pButton
end mouseDown

on mouseMove
end mouseMove

// Keyboard
on keyDown pKey
end keyDown

on keyUp pKey
end keyUp
```