LiveCode Builder - LiveCode Cheat Sheet

Comments

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

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

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

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

Variables

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

LiveCode Builder LiveCode

```
variable
tVar
put
"str"
into
tVar
put 1
into
tVar
```

```
local
tVar
put
"str"
into
tVar
put 1
into
tVar
```

```
variable tArr
as Array
put "val"
into
tArr["key"]
```

```
put "val"
into
tVar["key"]
```

Constants

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

LiveCode Builder   LiveCode

```
constant
kFoo is
15
```

```
constant
kFoo =
15
```

Control Structures

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

LiveCode Builder   LiveCode

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


repeat
with tX
from 1 up
to 10
end
repeat


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

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

```
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.

LiveCode Builder LiveCode

```
// 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"

split "a,b,c" by "," into tItems
tItems[3] is "c"

split
```

```
// 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"
```

```
"hi
there" by
" " into
tWords
tWords[1]
is "hi"


split
"anb" by
"n" into
tLines
tLines[2]
is "b"
```

```
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"
```

```
split "a,b,c"
by "n" into
tLines
split tLines
by "," into
tItems
char 1 of
tItems[1] is
"a"
```

String Processing

These examples show how string values can be manipulated.

LiveCode Builder LiveCode

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

```
// 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.

LiveCode Builder LiveCode

```
// 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
```

```
// 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
```

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

Sorting

These examples show how to sort items and lists.

LiveCode Builder LiveCode

```
variable
tList
put
[5,2,3,1,4]
into tList
sort tList
in
ascending
numeric
order
 -> tList
is
[1,2,3,4,5]
sort tList
in
descending
numeric
order
 -> tList
is
[5,4,3,2,1]
public
handler
DoSort(in
pLeft, in
pRight)
returns
Integer
return
pLeft[2] -
pRight[2]
end
handler
```

```
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"
```

```
variable tData
as List
put [[6, 1],
[8, 3], [2, 2]]
into tData
sort tData
using handler
DoSort
-> tData is [[6,
1], [2, 2], [8,
3]]
```

Files & Processes

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

LiveCode Builder LiveCode

```
get the
contents
of file
tPath
set the
contents
of file
tPath to
""
```

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

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

Custom Handlers

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

LiveCode Builder   LiveCode

```
handler
foo(in
pParam)
end foo
// get
foo(tVar)
// foo 5
```

```
function
foo pParam
end foo
// get
foo(tVar)
```

```
command bar
pParam
end bar
// bar 5
```

Event Handlers

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

LiveCode Builder	LiveCode

```
// Mouse
handler
OnMouseUp()
    get the
click button
end handler
handler
OnMouseDown()
get the
click button
end handler


handler
OnMouseMove()
end handler
```

```
// Mouse
on
mouseUp
pButton
end
mouseUp
on
mouseDown
pButton
end
mouseDown


on
mouseMove
end
mouseMove


//
Keyboard
on
keyDown
pKey
end
keyDown
```

```
// Keyboard
handler
OnKeyPress(in
pText)
end handler
```

```
on keyUp pKey
end keyUp
```