JavaScript - LiveCode Builder Cheat Sheet

Comments

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

JavaScript

```
// These
/* are

commented
 out */
```

LiveCode Builder

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

Literals

A literal is a notation for creating a particular type of value.

JavaScript

LiveCode Builder

```
"string literal"
'string literal'
["array", "of", "literals"]
{"object": "literal"}
```

```
"string literal"
["list", "of", "literals"]
{"array": "literal"}
```

Variables

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

JavaScript

```
var myVar;
myVar = "str";
myVar = 1;
```

```
var arr = {};
arr["key"] = "val";
```

LiveCode Builder

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

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

Constants

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

changes.

JavaScript LiveCode Builder

```
const
FOO =
15;
```

```
constant
kFoo is
15
```

Control Structures

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

JavaScript LiveCode Builder

```
for (var i=0; i
< text.length;
i++) {
    char =
text.charAt(i);
}
for (var i=0; i
< 10; i++) {
}


while (x > 1) {
x--;
}


if (value) {
} else if
(other) {
} else {
}
```

```
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
```

```
switch (value) {
case "a":
break;
default:
break;
}
```

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

Operators

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

JavaScript LiveCode Builde

```
// Logical
true && false == false
true || false == true
!false == true
// String
"foo" + "bar" == "foobar"
var strs = ['foo','bar'];
strs.join(" ") == "foo
bar"


"string".startsWith("st");
"string".endsWith("g");


// Chunks
"string".charAt(4) == "n"


var items =
"a,b,c".split(",");
items[2] == "c"


var words = "hi
there".split(" ");
words[0] == "hi"


var lines =
"anb".split("n");
lines[2] == "b"
```

```
var lines = "a,b,c".split("n")
var items =
lines[1].split(",")
items[1].charAt(0) == "a"
```

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


split
"anb" by
"n" into
```

```
tLines
tLines[2]
is "b"
```

```
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.

JavaScript

LiveCode

```
# General
str = 'a' + str;
str = str.slice(1);
str = str.replace("_", "-")
```

Regex

```
var found = /[0-9]/.exec("1");
var num = found[1];
```

```
str.split("n").filter(function(elem)
{
return pattern.exec(elem) != NULL;
});
```

```
//
Genera
put "
before
tVar
delet
char 1
of tVa
repla
"_"
with "
" in
tVar
```

Array Processing

These examples show how array values can be manipulated.

JavaScript

```
# Split / combine
var list = "a,b,c".split(",")
list[1] is "b"
list = list.join(",");
list == "a,b,c"
for (var key in array) {

Do something with array[key];

}

Length

array.length();
```

LiveCode

```
// S
coml
put
into
spli
by "
tVar
"b"
coml
with
tVar
"a,b
// I
repe
each
in t
-- D
some
with
tArr
end


repe
each
tEle
tArr
end

// Leng
the num
element
```

Sorting

These examples show how to sort items and lists.

JavaScript

```
var list = [5, 2, 3, 1, 4]
list.sort();
-> list == [1, 2, 3, 4, 5]
list.reverse();
-> list == [5, 4, 3, 2, 1]
```

```
var data = [[6, 1], [8, 3], [2, 2]];
data.sort(function(a,b) {
return a[2] - b[2]
});
-> data == [[6, 1], [2, 2], [8, 3]]
```

LiveCode Builder

```
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
```

```
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]]
```

Custom Handlers

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

JavaScript

```
function
foo(param)
{
}
//
foo(value)
```

LiveCode Builder

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

JavaScript

LiveCode Bu

```
# Mouse
function handleMouseUp {
}
<button
onmouseup="handleMouseUp" />
function handleMouseDown {
}
<button
onmousedown="handleMouseDown"
/>


function handleMouseMove {
}
<div
onmousemove="handleMouseMove"
/>
```

Keyboard

```
function handleKeyUp {
}
<input onkeyup="handleKeyUp"
/>
```

```
function handleKeyDown {
}
<input onkeydown="handleKeyDown"
/>
```

Offline (Leave a message)

```
// Mouse
handler
OnMouseU
    get
click b
end hand
handler
OnMouseD
get the
click b
end hand


handler
OnMouseM
end hand
```

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