Listing all the handlers in a script

This lesson demonstrates how to create a list of all the handlers in a script.

The handlerNames function

The handlerNames function takes two parameters

pScript - the text of the script

pHandlerType - the type of handler to be listed (optional)

This custom function is called with a statement such as one of the following:

```
get handlerNames(the script of button "OK","message")
```

```
put handlerNames(myScript) into allHandlerNames
```

```
if handlerNames(the script of me,it) is "(None)" then go next
```

Locating handlers in a script

The **handlerNames** function works by looking for text that is common to all handlers (or to all handlers of a certain type). There are four kinds of handlers in LiveCode:

- message handlers, which all begin with the word "on", the words "private on", the word "before" or the word "after"
- function handlers, which all begin with the word "function" or the words "private function"
- getProp handlers, which all begin with the word "getProp"
- setProp handlers, which all begin with the word "setProp"

The **handlerNames** function takes advantage of this to locate handlers in a script by finding all the lines that begin with one of the words that mark the start of a handler. For each possible handler type, the function uses the filter (https://livecode.com/resources/api/#livecode_script/filter) command to find only the lines that start with that handler type's characteristic word. For example, to find message handlers, the function gets only the lines in the script that begin with "on", "before" or "after". After using the filter (https://livecode.com/resources/api/#livecode_script/filter) command, the pScript parameter variable contains only the lines that matched the filter we used: in this case, only the starting lines of each handler of the type requested.

```
function handlerNames pScript,pHandlerType
   switch pHandlerType
      case "message"
         ## list the message handlers
         filter pScript matching regex pattern "^(private )?on|^after|^before"
         break
      case "function"
         ## list the function handlers
         filter pScript with regex pattern "^(private )?function"
         break
      case "setProp"
         ## list the setProp handlers
         filter pScript with "setProp*"
         break
      case "getProp"
         ## list the getProp handlers
         filter pScript with "getProp*"
         break
      default
         ## list all the handlers, if no type specified
         filter pScript with "end*"
   end switch
```

You'll notice that you can call the function with just one parameter (pScript). It's possible to do this because the default clause of the switch control structure is executed if pHandlerType is empty (or anything other than "message", "function", "setProp", or "getProp").

Finding the handler names

The handler's name is always part of the script line that starts a handler. In fact, it is the word immediately after the handler type (on, before, after, function, getProp, or setProp). So by taking the second word of each line we've found (or third, if the first word is "private"), we can get just the name of each handler. The repeat loop that does this simply substitutes the appropriate word of each line for the entire line. In this way, we obtain a list of handler names, which is returned to the handler that called the function.

If no handler names were found, that is, if the list is empty, the message "(None found)" is returned. Depending on what this function is intended to do, we might want to return something else. For example, if the return value is going to be used to perform some operation on each handler in turn, we might want to return empty instead.

```
   repeat with x = 1 to the number of lines of pScript
      if word 1 of line x of pScript is "private" then
         ## word 3 of the first line of a private handler is the handler name:
         put word 3 of line x of pScript into line x of pScript
      else
         ## word 2 of the first line of any other handler or the last line of any handler is the handler name:
         put word 2 of line x of pScript into line x of pScript
      end if
   end repeat
   if pScript is empty then
      return "(None found)"
   else
      return pScript
   end if
```

The handlerNames function code

```
function handlerNames pScript,pHandlerType
   switch pHandlerType
      case "message"
         ## list the message handlers
         filter pScript matching regex pattern "^(private )?on.+|^after.+|^before.+"
         break
      case "function"
         ## list the function handlers
         filter pScript with regex pattern "^(private )?function.+"
         break
      case "setProp"
         ## list the setProp handlers
         filter pScript with "setProp*"
         break
      case "getProp"
         ## list the getProp handlers
         filter pScript with "getProp*"
         break
      default
         ## list all the handlers, if no type specified
         filter pScript with "end*"
   end switch

   repeat with x = 1 to the number of lines of pScript
      if word 1 of line x of pScript is "private" then
         ## word 3 of the first line of a private handler is the handler name:
         put word 3 of line x of pScript into line x of pScript
      else
         ## word 2 of the first line of any other handler or the last line of any handler is the handler name:
         put word 2 of line x of pScript into line x of pScript
      end if
   end repeat

   if pScript is empty then
      return "(None found)"
   else
      return pScript
   end if
end handlerNames
```

3 Comments

Fiona Sabba Tuesday Nov 05 2013 at 04:43 AM

I am trying to create a case which will work with numbers, I cannot find any exemplars. Please help.

// Setup the global variables to be used in subroutines
global book_title, book_price, number_books, discount, total_price

on mouseUp
initialise
enter_book_details
//calculate_discount
display_total
end mouseUp

on initialise

```
put "" into book_title
put 0 into book_price
put 0 into number_books
put 0 into total_price
end initialise

on enter_book_details

// Setup the card
put empty into field "output"
set the backgroundColor of this card to 220,220,220
set the textColor of field "output" to 0,0,0

// Prompt the user for the book title
ask "Please enter the title of the book"
put it into book_title

// Prompt the user for the price of the book
ask "Please enter the price of the book"
put it into book_price

// Prompt the user for the number of books
ask "Please enter the number of books you wish to order"
put it into number_books

end enter_book_details

on calculate_discount
// Start a switch statement
switch number_books

case 51 to 80
put 0.1 into discount
put discount into line 3 of field "output"
//set the backgroundColor of this card to 255,0,0
//set the textColor of field "output" to 255,0,0
break

case 11 to 50
put 0.075 into discount
// set the backgroundColor of this card to 0,0,255
//set the textColor of field "output" to 0,0,255
break

case 5 to 10
put 0.05 into discount
//set the backgroundColor of this card to 85,107,47
//set the textColor of field "output" to 85,107,47
break

case 1 to 4
put 0.01 into discount
//set the backgroundColor of this card to 0,0,0
//set the textColor of field "output" to 0,0,0
break
end switch
```

```
// If the user enters a value not on the list above then set the colour to grey and text to black
// Display the following error message
//if field "output" is empty then
//set the backgroundColor of this card to 220,220,220
//set the textColor of field "output" to 0,0,0
//put "There is no discount available" into line 1 of field "output"
//end if

put (book_price - (book_price*discount)) * number_books into total_price
end calculate_discount

on display_total
//this function will set the output of numbers to two decimal places
set numberformat to "00.00"
put book_title into line 1 of field "output"
put " the total price payable is £" &&total_price into line 2 of field "output"
end display_total
```

Mark Wieder Tuesday Nov 05 2013 at 11:41 AM

I think a case statement might not be the most appropriate construct here, and what I would do is code something like

```
if number_books > 50 then
put 0.1 into discount
else if number_books > 10 then
put 0.075 into discount
else if number_books > 4 then
put 0.05 into discount
else
put 0.01 into discount
end if
```

Richard Gaskin Friday Aug 15 2014 at 12:55 PM

With the introduction of the private, before, and after tokens the script here would need revision to account for them.

Fortunately an existing engine function now provides this in a more complete and efficient way: the revAvailableHandlers, e.g.:

```
get revAvailableHandler()
```