Custom Commands

Commands allow you to group a piece of code together and then use that code whenever you need it. If you are familiar with macros or procedures LiveCode commands are similar.

Custom commands were briefly introduced in Simple Messages, but let's revisit them here.

A command is an instruction to LiveCode to do something. There are built in commands like **put** and **set** (which you have come across before!) but you can also create custom commands. Custom commands can have multiple lines and you give them a name you can use to "call" them from anywhere in your code. Calling a command tells LiveCode to execute all the code within that command.

If you want to make changes to your command you can, and the new code is executed whenever the command is called. This makes corrections and changes much easier to do.

Parameters

Parameters are an extra piece of information that is passed to a custom command when it is called. Parameters allow the command to use the information provided to it; the parameter can be different each time the command is called.

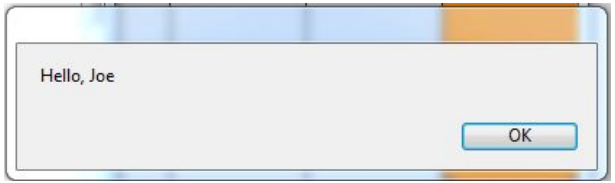
For example, you can define a custom command **displayGreeting**:

```
command displayGreeting pName
  answer "Hello," && pName
end displayGreeting
```

This command can be called when a button is clicked, and we can pass “Joe” as a parameter:

```
on mouseUp
  displayGreeting "Joe"
end mouseUp
```

This will result in the dialog box on the right popping up!



The operatorPressed Command

Recall how the LiveCode message path works. If you have forgotten then go back to the slides in the Simple Messages App to jog your memory!

If we call a command, then LiveCode will first look for that command in the script of the object that called it. If the command isn't there, it will then look for it in the object's owner. And so on.

We are going to define a custom command in the card script, which we will call from the **divide**, **multiply**, **plus** and **minus** buttons.

Open the Card Script (using the Object menu) and add the command on the right.

```
command operatorPressed pOperator

  set the showBorder to button "divide" to false
  set the showBorder to button "multiply" to false
  set the showBorder to button "minus" to false
  set the showBorder to button "plus" to false

  set the showBorder of button pOperator to true

end operatorPressed
```

Update the Operator Button Code

All we need to do now is update the code in the script of buttons **divide**, **multiply**, **plus** and **minus** to the code shown on the right.

When LiveCode executes the code on the right:

1. It calls the command **operatorPressed**, with the name of the button that was clicked as a parameter
2. It looks for the command in the script of the button that was clicked
3. When it cannot find that command, it then looks to the script of the owner of the button, the card script, for the command
4. It finds the command in the card script and it executes the code in the command

We have now removed the need for an extra 16 lines of repeated code by simply adding 6 lines of code to a custom command.

```
on mouseUp
  operatorPressed the short name of me
end mouseUp
```

LiveCode Key Concept

name - This property specifies the name of an object.

set the [long | abbreviated | short] name of object to string
get the name

Use an object's **name** property to refer to the object.

The short name of an object is simply the name that the object has been assigned. When you set an object's **name** property or type it into the object's property inspector, the short name is what you provide. The abbreviated name of an object is the object's type followed by the object's short name in quotes. If you don't specify a modifier for the **name** property, the abbreviated name is what you get. The long name of an object includes information about its owner (and about the owner of that owner and so on).

For example, if you have a button "clickMe" on card 1 of a stack then:

- the long name is **button "clickMe" of card id 1002 of stack "Untitled 1"**
- the abbreviated name is **button "clickMe"**
- the short name is **clickMe**