Local and global variables:

Short-term and long-term memory

Some variables are only available while a script is running. Others are available to all the scripts in your application. In this tutorial we discuss the differences between the two, and when and how to use each.

Key topics covered in this tutorial

- The difference between global and local variables
- How to use local variables
- How to use global variables
- Naming variables
- Storing global variables permanently using text fields
- Using Search and Replace in the Script Editor

See also: Reference Documentation: Containers, variables, and sources of value

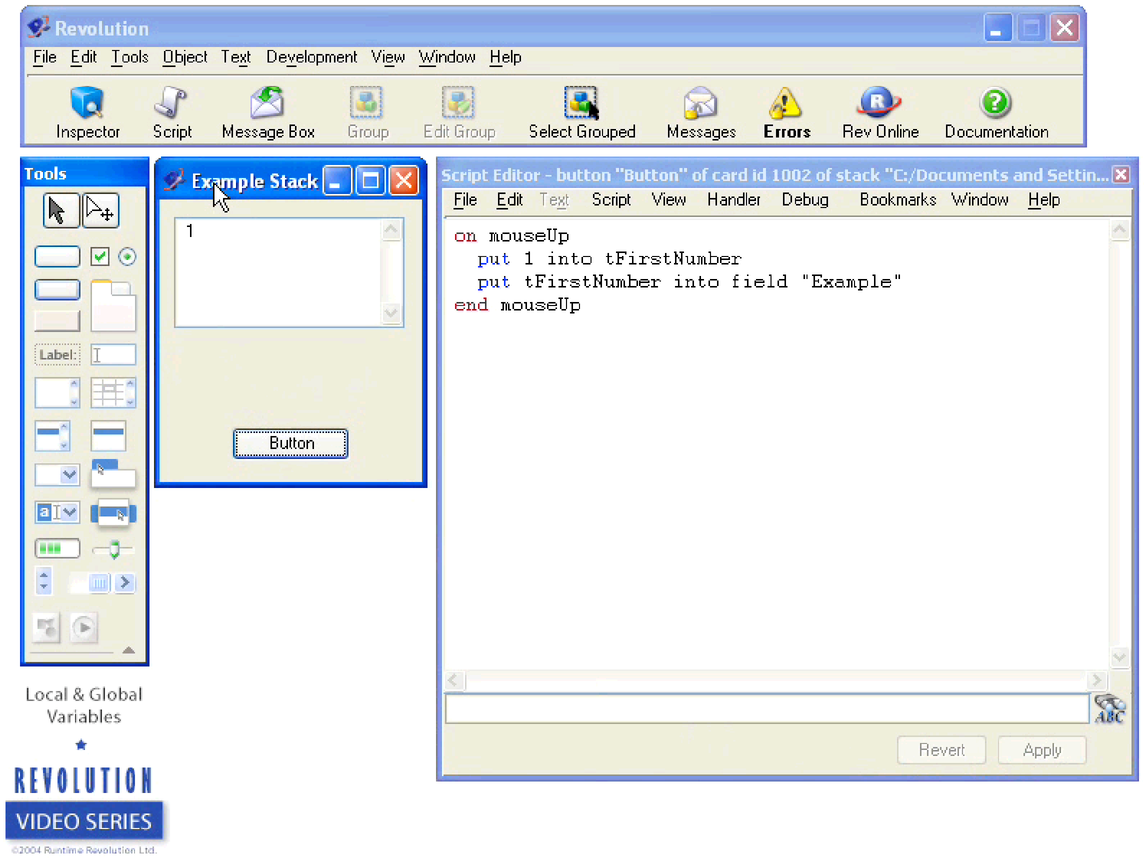
Variables can be accessible to a certain message handler, or to all the message handlers in the script of an object, in these cases they are known as local variables. Variables which are accessible to an entire Revolution application are known as global variables. Most variables you use will be of the first sort – local variables available only to a specific message handler or to all the message handlers in a script.

Let's write a simple script to see the difference between the two types of local variable.

```
put 1 into tFirstNumber

put tFirstNumber into field "Example"
```

This puts '1' into the field (which we previously named 'Example') as we would expect.


Revolution
File Edit Tools Object Text Development View Window Help
Inspector
Script
Message Box
Group
Edit Group
Select Grouped
Messages
Errors
Rev Online
Documentation
Tools
Label:
Example Stack
1
Button
Script Editor - button "Button" of card id 1002 of stack "C:/Documents and Settin...
File Edit Text Script View Handler Debug Bookmarks Window Help
Revert
Apply
Local & Global Variables
REVOLUTION
VIDEO SERIES
©2004 Runtime Revolution Ltd.

```
on mouseUp
  put 1 into tFirstNumber
  put tFirstNumber into field "Example"
end mouseUp
```

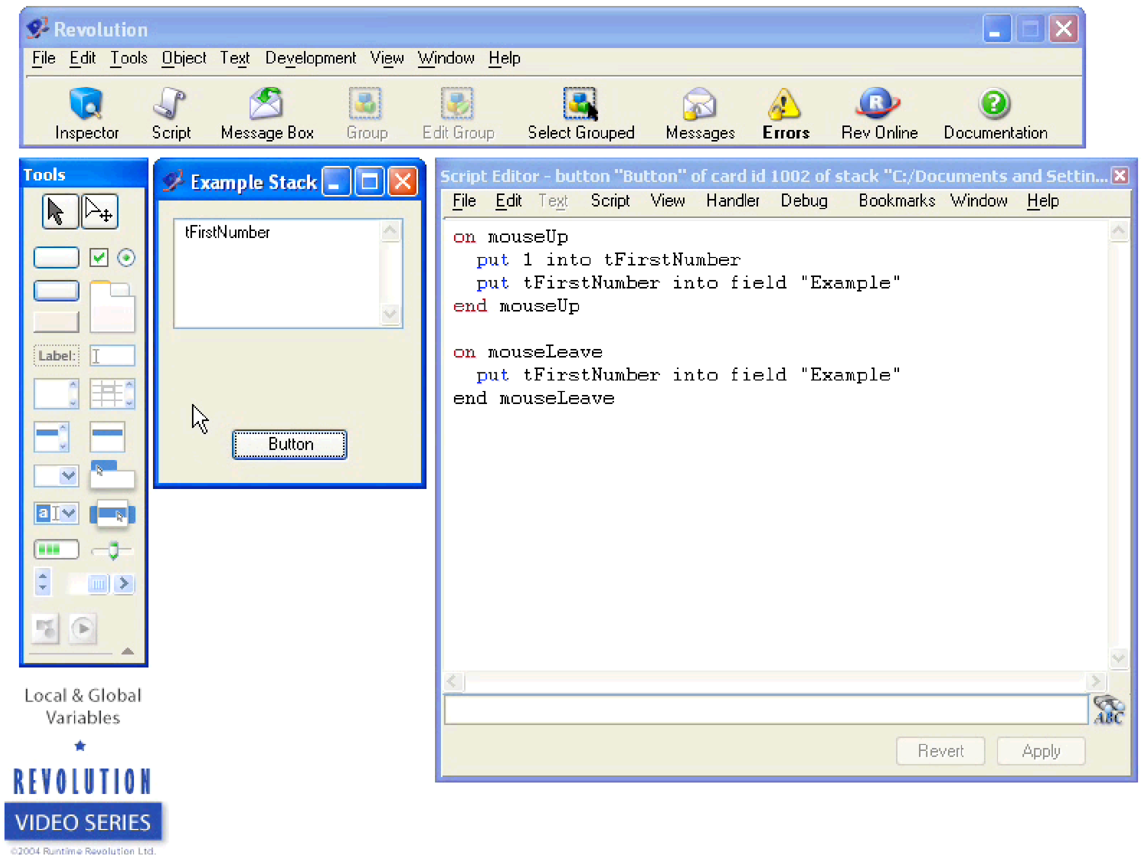
Now let's add a mouseLeave message handler and see if that value is still present.

```
On mouseLeave

put tFirstNumber into field "Example"

end mouseLeave
```

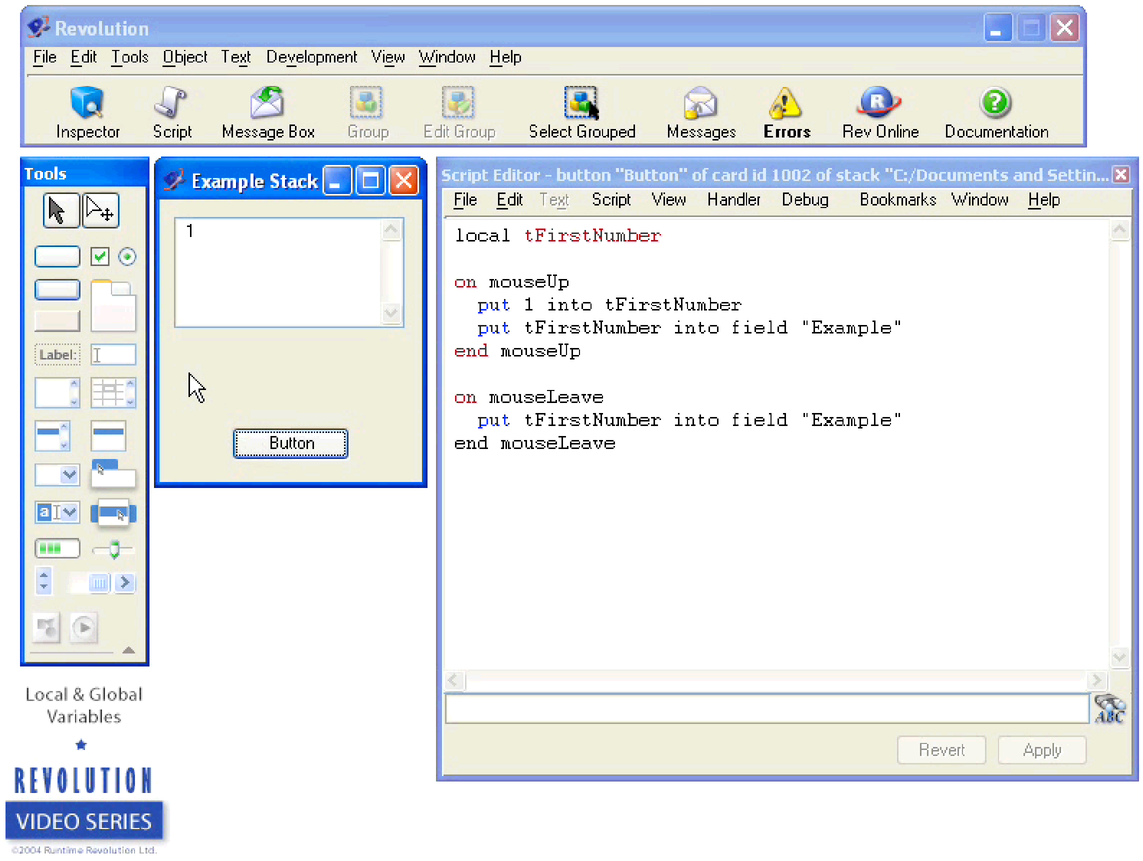
If we try it, we find that we have 1 when we click, but when we move the mouse out Revolution puts 'tFirstNumber' in the field. The value that was in the mouseUp handler is not being carried over to the mouseLeave handler. In fact, the mouseLeave handler doesn't even know that tFirstNumber is a variable at all – nothing has been put into it, so Revolution makes its best guess and interprets it as text. (Note that if you did want this text to be displayed, you would be better to put it in quotes as there is no guarantee that Revolution will be able to understand what you mean otherwise.)


Revolution
File Edit Tools Object Text Development View Window Help
Inspector
Script
Message Box
Group
Edit Group
Select Grouped
Messages
Errors
Rev Online
Documentation
Tools
Example Stack
tFirstNumber
Button
Script Editor - button "Button" of card id 1002 of stack "C:/Documents and Settin...
File Edit Text Script View Handler Debug Bookmarks Window Help
Revert
Apply
Local & Global Variables
REVOLUTION
VIDEO SERIES
©2004 Runtime Revolution Ltd.

```
on mouseUp
  put 1 into tFirstNumber
  put tFirstNumber into field "Example"
end mouseUp

on mouseLeave
  put tFirstNumber into field "Example"
end mouseLeave
```

So suppose you want this variable, currently local to only the mouseUp handler to be available in the mouseLeave handler too? You need to tell Revolution to make this variable available to all the message handlers in the script. You do this by adding 'local tFirstNumber' to the top of the script. If we click we still place 1 in the variable, and when we move the mouse out of the button we still have 1 in the variable.


Revolution
File Edit Tools Object Text Development View Window Help
Inspector
Script
Message Box
Group
Edit Group
Select Grouped
Messages
Errors
Rev Online
Documentation
Tools
Label:
Example Stack
1
Button
Script Editor - button "Button" of card id 1002 of stack "C:/Documents and Settin...
File Edit Text Script View Handler Debug Bookmarks Window Help
Revert
Apply
Local & Global Variables
REVOLUTION
VIDEO SERIES
©2004 Runtime Revolution Ltd.

```
local tFirstNumber

on mouseUp
  put 1 into tFirstNumber
  put tFirstNumber into field "Example"
end mouseUp

on mouseLeave
  put tFirstNumber into field "Example"
end mouseLeave
```

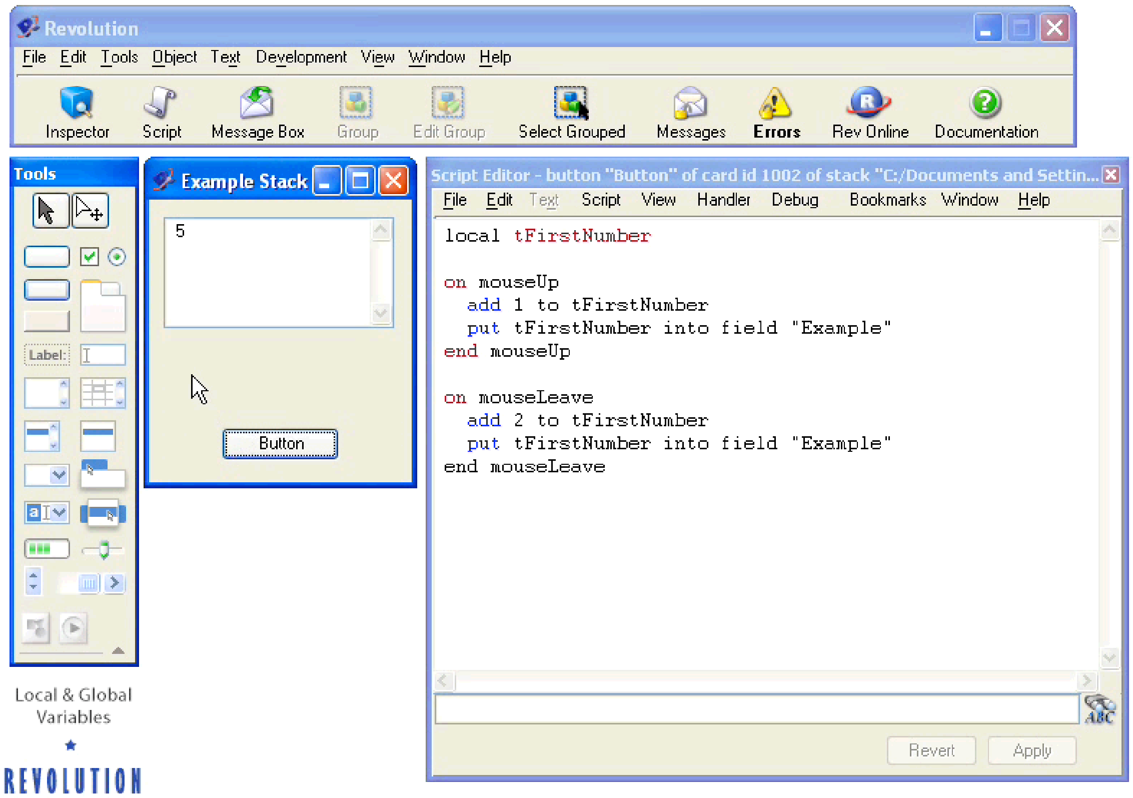
That local variable will continue to be available to all the other handlers in the script, getting erased only when we edit the script or close Revolution. Let's edit our script to try this out. Instead of putting 1 into our variable 'tFirstNumber', let's add 1 to tFirstNumber in the mouseUp handler and add 2 in the mouseLeave handler.

```
add 1 to tFirstNumber
```

and

```
add 2 to tFirstNumber
```

We'll apply the script – note that doing this empties any local variables in the script, so tFirstNumber is now made empty. If we click the button we get 1. If you try to add a number to an empty variable, Revolution guesses that you want to add a number to zero, saving you some work. Let's move the mouse out of the button and we have 3, then 5, and so on.


Revolution
File Edit Tools Object Text Development View Window Help
Inspector
Script
Message Box
Group
Edit Group
Select Grouped
Messages
Errors
Rev Online
Documentation
Tools
Label:
Example Stack
5
Button
Script Editor - button "Button" of card id 1002 of stack "C:/Documents and Settin...
File Edit Text Script View Handler Debug Bookmarks Window Help
Revert
Apply

```
local tFirstNumber

on mouseUp
  add 1 to tFirstNumber
  put tFirstNumber into field "Example"
end mouseUp

on mouseLeave
  add 2 to tFirstNumber
  put tFirstNumber into field "Example"
end mouseLeave
```

Local & Global Variables

REVOLUTION VIDEO SERIES

Let's have another look at the script. When we started, we made the tFirstNumber variable a local variable to just the mouseUp handler. We named the variable with a 't' at the start to designate it to ourselves as 'the' or 'temporary'. However, we then changed its scope – we changed the variable so that the other message handlers in the script of the button would have access to it.

It's a good idea when we make that change to name the variable a little differently. Revolution doesn't mind what you name a variable: its basic requirements are only that it doesn't conflict with a word that it already uses to mean something else, that it's a single word without a space, and that it starts with a letter and doesn't contain any operator symbols such as multiply or bracket. However, when you come back to read your script in a few months, and it's grown longer than the few lines we have here, you may have a hard time remembering just what that variable is for. We therefore recommend you always name your variable something descriptive. You can name your variable:

```
my_very_long_variable_name_which_is_really_memorable
```

but that has the downside that you have to type it every time you want to use it. We compromise by writing the variable with a lower case first letter 't' to remind us it is temporary, followed by one or two words, each word with an upper case letter, for readability. However, in this case, the variable is available to all the handlers in this script, not just temporary to one handler. We can use the letter 'l' instead of the letter 't' to remind ourselves the variable is a script-wide local variable. Getting into good habits now will make it easier to make changes to your script later, so let's use find and replace to make that change now. We'll click on 'Find and Replace' (the icon with 'ABC' and a pair of binoculars) in the script editor to bring up find and replace. We'll type in tFirstNumber and replace all instances of it with lFirstNumber.

So, now that we've looked at the two types of local variable: what if you want to make a variable available to your entire application, not just to the button object? You need to use a global variable. Normally we label global variables with a 'g' at the start to remind us what they are. You also have to declare a global variable in any message handler that you want to have access to it, or if you want an entire script to have access to it, at the top of the script. Let's try adding a global variable to the mouseUp handler:

```
global gSecondNumber
```

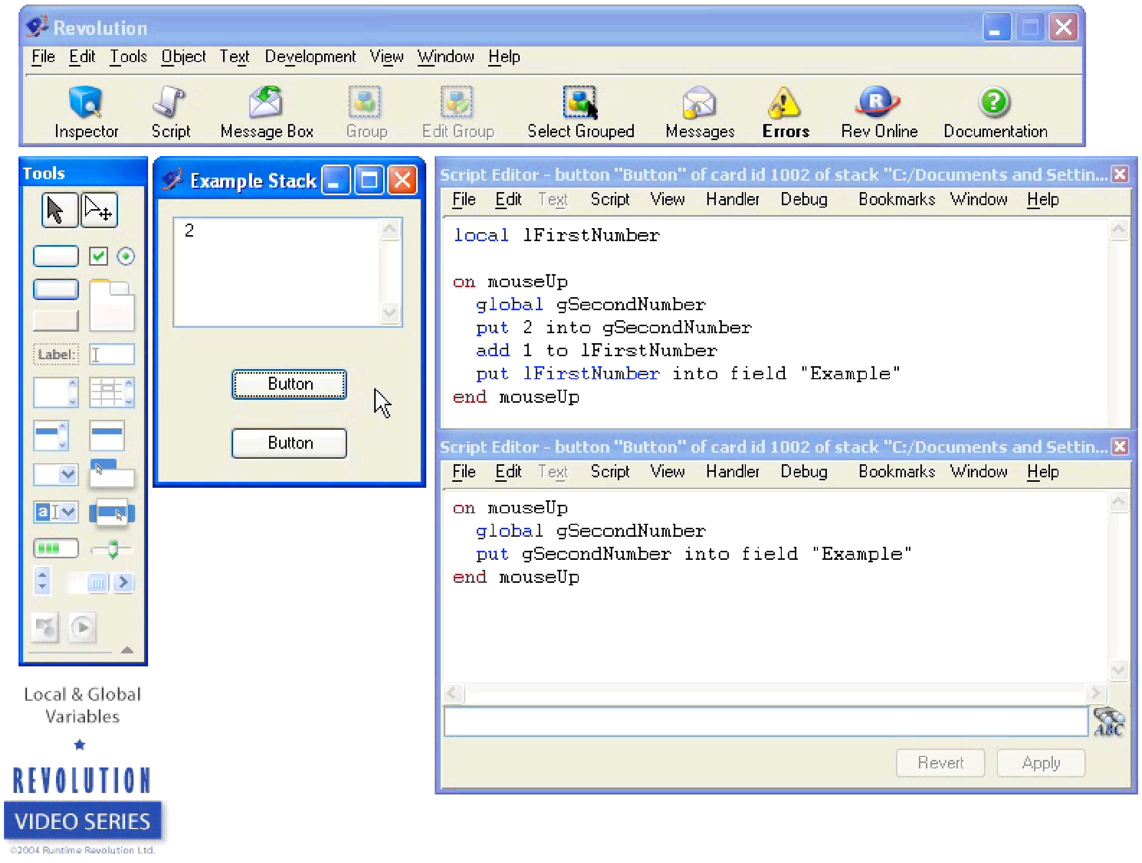
Now let's put something into that variable:

```
put 2 into gSecondNumber
```

Before we try it we need some way of telling us whether it worked. To do this let's create another button, declare the same global variable, and put the contents of that variable into the field.

```
global gSecondNumber

put gSecondNumber into field "Example"
```


Revolution
File Edit Tools Object Text Development View Window Help
Inspector
Script
Message Box
Group
Edit Group
Select Grouped
Messages
Errors
Rev Online
Documentation
Tools
Example Stack
2
Button
Button
Script Editor - button "Button" of card id 1002 of stack "C:/Documents and Settin...
File Edit Text Script View Handler Debug Bookmarks Window Help
Revert
Apply
Local & Global Variables
REVOLUTION
VIDEO SERIES
©2004 Runtime Revolution Ltd.

```
local lFirstNumber

on mouseUp
  global gSecondNumber
  put 2 into gSecondNumber
  add 1 to lFirstNumber
  put lFirstNumber into field "Example"
end mouseUp
```

```
on mouseUp
  global gSecondNumber
  put gSecondNumber into field "Example"
end mouseUp
```

We recommend you only use global variables when you need to, as these use up memory the entire time your application is running. Also, having extra variables available when you don't need them could make it harder to track down any problems in your scripts.

Finally, let's look at what you can do if your application needs to remember a variable after it has closed. An example might be a variable that holds the number of times the application has been used. Normally, any variable will be forgotten when you close the application. To get round this problem, you can put a variable into a text field for safekeeping. This text field can be hidden from the user, but will keep its contents even when you close the application. Here's a script to do just that. First, we'll open the stack script:

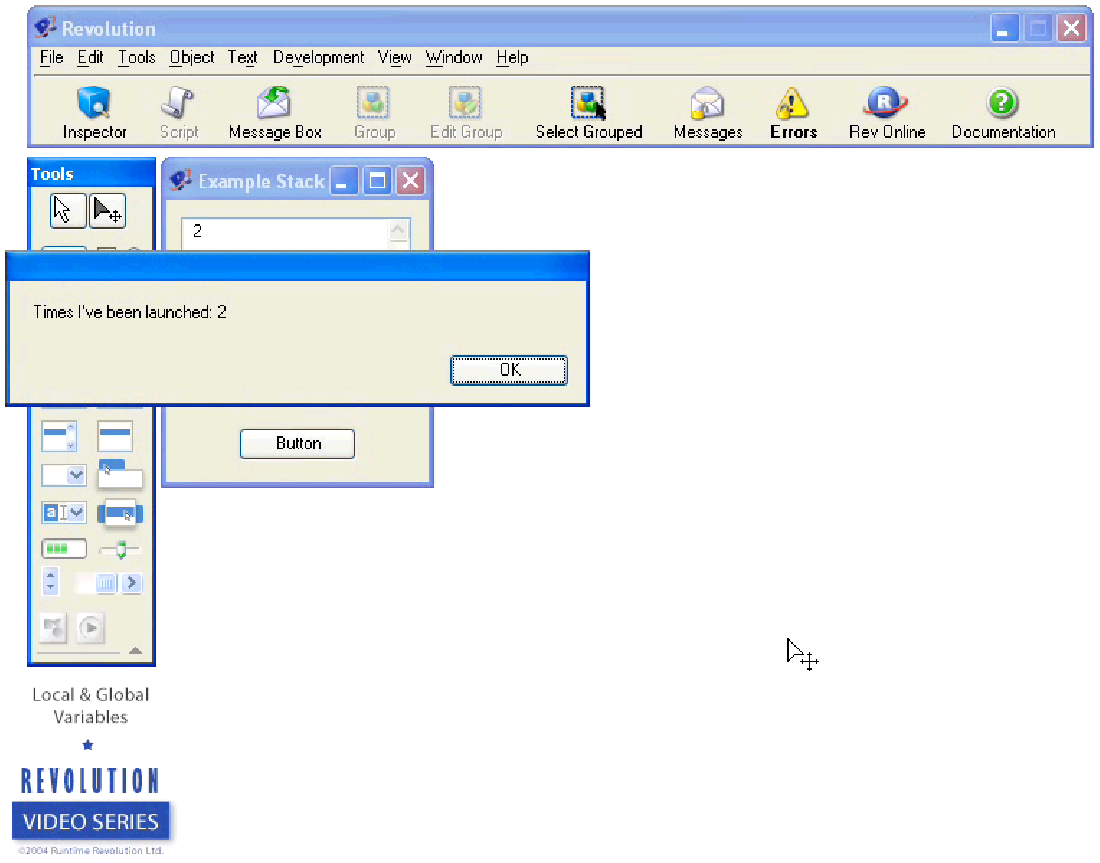
```
on openstack

add 1 to field "Launch Counter"

answer "Times I've been launched:" && field "Launch Counter"

end openstack.
```

Each time this stack is opened, Revolution sends an 'openstack' message to the stack which will trigger this script. The script then adds one to the number in the field and displays the result in a dialog. Now all we have to do is create the field, name it 'Launch Counter' and then switch to run mode to enter the initial value: 1. Finally, let's hide this field by unchecking the 'visible' checkbox in the properties palette, so the user can't see it. Now, if we save and close the stack, then reopen it the first thing that happens is that a dialog is displayed saying 'Times I've been launched: 2'.



Times I've been launched: 2