

# Interactive Design in QuarkXPress

Expressions



# Why Expressions?



The pre-defined actions in QuarkXPress make it easy to create Flash designs without having to program or code.

However, often you need to go a bit further and add something, which the pre-defined actions don't cover. For that Interactive layout offers "Expressions".

# What are Expressions?



Expressions are NOT programming or coding.  
Expressions are also not ActionScripting.

Expressions are formulas.

Think of a program like MS Excel:  
Normal actions are like the Sum Button in Excel.  
Expressions are formulas as SUM(A1:A9) or  
LOOPUP("x";B1:B100;2). ActionScript (coding) would  
be like Macros (Visual Basic) in Excel.

Note: If you need to code (ActionScript), use a  
coding environment like the Adobe Flash package.

# Using Expressions



There are two places where you can use Expressions:

- 1) By using the expression action (Expression > Set)
- 2) By using it everywhere where there is a Calculation sign ( $\Sigma$ ), e.g. in Duration, Coordinates and also CONTROL actions (e.g. IF).

Expressions do not need to be used in scripts, they can also be used by "single actions" (Event tab).

# Using Expressions



To enter an Expression you can either enter text (free-form) or use the Expression Editor.

The Expression Editor (EE) can be invoked by pressing on the  $\Sigma$  symbol.

There is NO difference which method you use, the Expression Editor is like a Wizard helping you to enter correct expressions.

Tip: Even if you don't need the help the EE provides, it offers you more space to enter text.

# Using Expressions



Example (Interactive):

Open Example 1.

It contains seven text boxes that show the days of the week. All boxes are initially hidden.

There's a script that automatically gets executed when page 1 shows. The script will show only the text box displaying the current day.

# Using Expressions



For that, the script uses a function called `Date.GetDay()`, which returns 0 if the computer date is a Sunday. More later.

The script uses seven IF "blocks". The first one looks like this:

```
IF Expression Date.GetDay() == 0  
    Show Object "Sunday"  
END IF
```

Note: The IF statement uses an operator ("`==`") to compare.

# Using Expressions - Operators



To compare elements you need “operators” like ‘equal’, ‘not equal’, ‘small than’ etc.

‘==’ is ‘equal’ (careful, not ‘=’)

‘!=’ is ‘not equal’

‘<’ is ‘smaller’

‘<=’ is ‘smaller or equal’

‘>’ is ‘larger’

‘>=’ is ‘larger or equal’

etc.

So “ $3 > 2$ ” is true and “ $3 < 2$ ” is false.

# Using Expressions



The expressions can be roughly categorized into two areas:

- 1) Functions
- 2) Properties

Both look and behave very similar, the only difference is that functions are predefined (e.g. the computer date or mouse cursor).

Properties work on YOUR objects (e.g. text content or its opacity).

# Using Expressions - Functions



Functions are used in the form of FUNCTIONALAREA.FUNCTION() - separated by a dot.

In our example, Date (a predefined area) is the functional area and the function is called GetDay.

Date.GetDay() gets the computer day of the week as a number, e.g. 0 for Sunday, 1 for Monday etc.

In our example we compare the result of the function (e.g. 3 for Wednesday) with a number. If the condition (3==3) is true, the action (show) after the IF gets executed.

## Using Expressions - Functions



Tip: When you click on the function the Expression Editor gives you a small explanation of what the function or property does.

Also, if you enter wrong expressions (e.g. "Date.GetDays()"), the EE will inform you about it. You can also use the "Validate" button to do so.

Note: Sometimes the EE offers you to create a variable, more on variables later.

# Using Expressions - Properties



Properties come in the form of `OBJECT.PROPERTY()`.

Basically they are the same as functions (GetDate), just they work on the objects you created.

You can use properties of objects to read attributes of objects, e.g. its opacity or position (x/y).

Some properties you can also set yourself to manipulate objects (e.g. to set a different opacity).

# Using Expressions - Properties



Example:

To read the opacity of the object you called "mypicture":

```
mypicture.GetOpacity()
```

To set the opacity of the object you called "mypicture":

```
mypicture.SetOpacity(60)
```

Note the '60' in the second example.

# Using Expressions - Properties



The difference makes sense, right?

If you **read** the opacity of an object (in the first example), all you need is the name of the attribute (property) and you get a result back.

If you **set** the opacity though, you need to specify a value (0 to 100), specifying how opaque the object should become.

Example (Interactive):

Open Example 2 and click on the yellow object.

# Using Expressions - Variables



What now if I want to vary its opacity **over time**?

I could create several actions in one script:

Delay 1 sec

picture.SetOpacity(90)

Delay 1 sec

picture.SetOpacity(80)

Delay 1 sec

picture.SetOpacity(70)

Delay 1 sec

(etc.)

This gets very long (20 actions).

# Using Expressions - Variables



To do this, it is easier using variables.

As the name suggests, variables are used where you need to work with values (or strings) that can vary during runtime or where you do not know the content upon authoring.

Imagine a variable like a drawer with a label, in which you can put content and look it up again.

The variable needs a **unique** name and you need to specify what it holds (numbers, strings, objects).

## Using Expressions - Variables



In our next example, we'll use a variable called 'opaquer' and it needs to be integer (integers are non-fractional numbers).

To work with it we **first** need to define the variable (name & type). You can do that either in the menu "Edit > Variables" or from with the EE (button "Variables New/Edit").

To work with it, we need to put content (e.g. a value) into a variable. To assign content you need to use the operator '=' (remember the meaning of '=='?).

# Using Expressions - Variables



Example (Interactive):

Open Example 3.

Go to menu "Edit > Variables" and check the variable definition. Variable 'opaquer' is integer.

Clicking on the yellow object will execute a script (only reason for using a script: we will need more than just one action).

This script will reduce the opacity of the yellow object slowly.

# Using Expressions - Variables



The script looks like this:

```
SET EXPRESSION opaquer = 100
WHILE EXPRESSION opaquer > 0
SET EXPRESSION opaquer = opaquer - 10
SET EXPRESSION picture.SetOpacity(opaquer)
DELAY 1 sec (you could also calculate a time)
END WHILE
```

Note: The script uses 6 actions (instead of 20 if we hadn't used a variable). Plus we gain the flexibility to easily change the steps from 10 to e.g. 1.

# Using Expressions - Variables



The first action sets the content of the variable to 100.

The WHILE block afterwards you should understand after having worked through lesson 1. The WHILE loop is being executed as long as the value of 'opaquer' is larger than 0 (zero).

The forth action sets the opacity of the yellow object (by using SetOpacity).

The new expression is the third action.

# Using Expressions - Variables



Expression "opaquer = opaquer - 10"

That's mathematically WRONG!!!????

True. In mathematics of course  $X$  can't be the same as  $X - 10$ .

However '=' is not the mathematical 'equal'. '=' is 'assign' or 'make'.

It reads 'The new value of the variable 'opaquer' will be of what is currently stored in 'opaquer' reduced by 10.

## That's it, you understood Expressions and Variables.

Well, of course that's not all of the possibilities, there are dozens of functions, properties and thousands of ways on how to use it. Use the EE to find what what the functions and properties can do for you.

That makes expressions so powerful and flexible.

# Expressions - More expressions



Other functions that can be helpful are

- 1) To concatenate strings use variables:  
`newtextvar = text1var + text2var`
- 2) Test whether objects intersect:  
`IF Object1.Intersects(Object2)`
- 3) Find out where the mouse is (x/y coordinates):  
`xmouse = Environment.GetXMouse()`
- 4) Create a random number:  
`Number.RandomInt(min,max)`

# Expressions - Some examples



What other things can you do with expressions?

1) Concatenate strings:

`Newtext = text1 + text2`

2) Test whether objects intersect:

`IF Object1.Intersects(Object2)`

3) Find out where the mouse is (x coordinate):

`IF xmouse = Environment.GetXMouse()`

4) Create a random number:

`Number.RandomInt(min,max)`

## Expressions - Some examples



- 5) Get text content of a text box:  
`mytextbox.GetText()`
- 6) Append text stored in variable into text box:  
`mytextbox.Append(mytextvariable)`
- 7) Set an object to position 100/200:  
`myobject.ObjSetLocation(100,200)`
- 8) Slide an object to position 100/200 in 2 secs:  
`myobject.SlideToLocation(100,200,2,1)`

# Expressions - Typical hurdles



There is no typical hurdles, as there expressions can be used in so many ways.

Some things to check if something doesn't work:

- Are all variables of the right type? E.g are you trying to reduce a string by 10?
- Did you confuse '=' and '=='?
- Do you pass the right value to a function? E.g. SetOpacity(n) takes numbers from 0 to 100.
- Are conditions ever true or always true? E.g. are you testing against 'IF x=1'? That's always true.

# Where to go from here?



The best to learn expressions and variables is to look at existing ones. Open the following two:

## 1) Test Your IQ:

[Gives you 5 seconds to answer a simple addition.](#)

The page automatically starts a script that acts as a timer. The timer is using a variable "counter", which is being reduced from 5 to 0. This value is displayed in a text box. When the counter is 0 (after 5 secs) then a 2<sup>nd</sup> script is run to see whether the answer is right. That script reads the correct answer from another text box (to make authoring simpler).

## 2) Scrollbar Vertical:

[Simulates a custom scrollbar and preventing drags outside fixed boundaries.](#)

Again, to make authoring simpler, variables are not initialized by numbers in a scripts but by reading objects coordinates of objects on the page. While the (red) scrollbar is being dragged a script is running that determines the mouse coordinates and sets the red scrollbar to these (but only if it is within boundaries). When the user releases the mouse button the 2<sup>nd</sup> script is ended.



Need more help?

Go to [forums.quark.com](https://forums.quark.com)