July 2008

In the Trenches: Scripting

AppleScripting Tables in QuarkXPress

By Ben Waldie

The release of QuarkXPress 7 has brought with it a number of enhancements that provide AppleScript developers with greater control over tables. In this month's column, we will explore a number of ways to use AppleScript to interact with tables in QuarkXPress 7.

If you open QuarkXPress' AppleScript dictionary in Script Editor (located in */Applications/AppleScript/*), you will find that the table-specific AppleScript terminology has been separated out into its own suite, appropriately named *Table Suite*. See figure 1.

Figure 1. Quark's Table Suite of AppleScript Terminology

Here, you will find the various classes, i.e. objects in QuarkXPress, that represent table elements, including tables, cells, rows, columns, and gridlines. If you click around, you will find that many of the classes in the *Table Suite* possess properties, or attributes, which are accessible, and in many cases, modifiable, from AppleScript.

To follow along with the examples that will be discussed in this month's column, you will need to launch QuarkXPress and create a new document. Please note that these examples are intended for use with QuarkXPress 7, and while some of the code may work in older versions of Quark, some will not.

Creating a Table

In AppleScript, a table is referred to as a `table box`, and you will find that the procedure for creating a table box via AppleScript is very similar to that of creating a text or picture box. To do so, you will use the `make` command. For example, the following code will create a new table on the first page of the frontmost QuarkXPress document.

```
tell application "QuarkXPress"
     tell page 1 of document 1
          make new table box at beginning
     end tell
end tell
--> table box 1 of page 1 of document "Project2" of
application "QuarkXPress"
```

While this alone may be useful to some degree, it becomes even more useful with the addition of the `make` command's `with properties` parameter. By utilizing this parameter, you can specify values for various attributes of the table, to be applied as the table is created. For example, the following code will create a 3-column by 3-row table of a specified size and position. See figure 2.

```
tell application "QuarkXPress"
     tell page 1 of document 1
          make new table box at beginning with
properties {bounds:{"1 in", "1 in", "3 in", "4 in"},
column count:3, row count:3}
     end tell
end tell
--> table box 1 of page 1 of document "Project2" of
application "QuarkXPress"
```

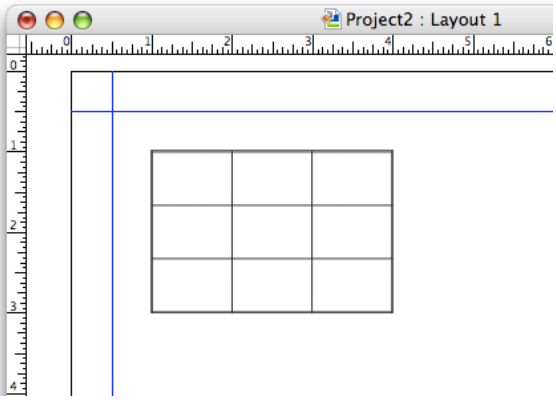

Figure 2. A Newly Created Table

In this example, the size and position of the table are specified using the `bounds` property of a table. The value specified, `{"1 in", "1 in", "3 in", "4 in"}`, is a list of position indicators representing the `{top of the table, left of the table, bottom of the table, right of the table}`.

Working with Gridlines

In QuarkXPress 7 (not so in older versions) gridlines are AppleScriptable via the `horizontal gridline` and `vertical gridline` classes. Attributes of gridlines, including width, shade, and opacity, are now modifiable via AppleScript.

To target a specific gridline, do so by its index, or position within the table. For example, the following code adjusts the width of the third horizontal gridline of a table.

```
tell application "QuarkXPress"
     tell table box 1 of page 1 of document 1
          set width of horizontal gridline 3 to "2 pt"
     end tell
end tell
```

Attributes of multiple gridlines can be modified at once with the use of AppleScript's `every` element reference, as demonstrated by the following code, which will set the color and width of every horizontal gridline of a table at once. See figure 3.

```
tell application "QuarkXPress"
     tell table box 1 of page 1 of document 1
          set color of every horizontal gridline to
"Magenta"
          set width of every horizontal gridline to "2
pt"
     end tell
end tell
```

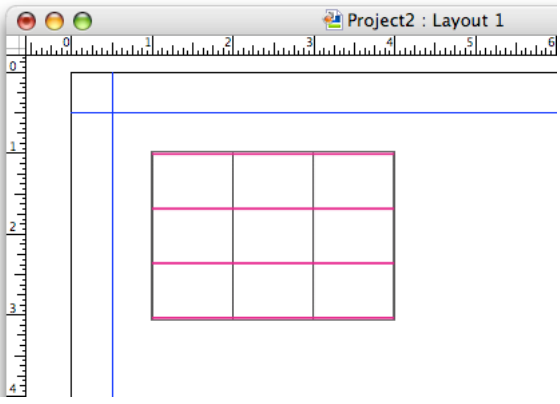

Figure 3. Colored Gridlines in a Table

Working with Rows and Columns

Rows and columns are referenced using the `table row` and `table column` classes. Like gridlines, rows and columns can be targeted by their index, or by using AppleScript's `every` element reference. Modifiable attributes of rows and columns include height (for rows), width (for columns), and auto fit. The following example code demonstrates how to adjust the height of the first row, and the width of the second column, within a table. See figure 4.

```
tell application "QuarkXPress"
     tell table box 1 of page 1 of document 1
          set height of table row 1 to 3
          set width of table column 2 to 3
     end tell
end tell
```

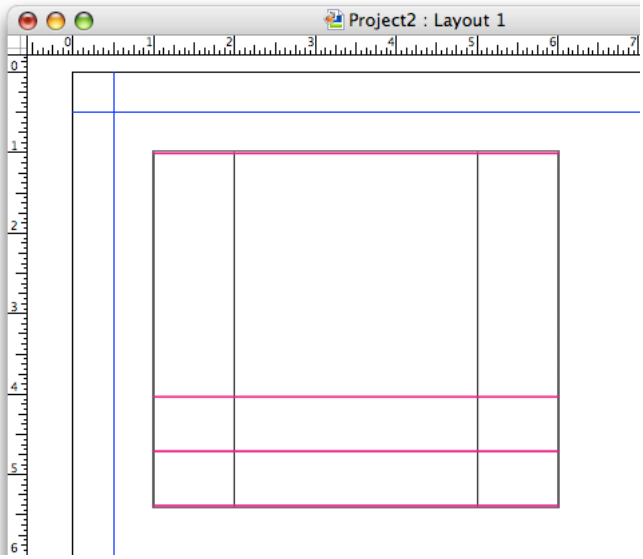

Figure 4. A Heightened Table Row

Working with Cells

Cells are accessible within table rows and columns by their index, or by using AppleScript's `every` element reference. Since table cells may contain different types of content, there are multiple classes that may be used to represent them, including the `graphic cell`, `picture cell`, and `text cell` classes. For example, the following code generates a reference to the first text cell in the second column of a table.

```
tell application "QuarkXPress"
```

```
        tell table box 1 of page 1 of document 1
            text cell 1 of table column 2
        end tell
    end tell
    --> text cell 1 of table column 2 of table box 1 of
    page 1 of document "Project1" of application
    "QuarkXPress"
```

In this case, if the first cell of the second column contained picture content, and the second cell contained text content, then the result of the above code would be a reference to the second cell in the column.

An all-encompassing `generic cell` class may be used to generically refer to a cell regardless of its content type. For example, the following code generates a reference to the first cell within the second column of a table, regardless of whether that cell contains picture or text content.

```
tell application "QuarkXPress"
    tell table box 1 of page 1 of document 1
        generic cell 1 of table column 2
    end tell
end tell
--> generic cell 1 of table column 2 of table box 1 of
page 1 of document "Project1" of application
"QuarkXPress"
```

Placing Text in a Table Cell

Using AppleScript, it is possible to add text content to a cell in a table, provided that the cell is a text cell. Since you may be unsure of the type of a cell, it makes good practice to check it prior to attempting to interact with the cell. To check the content of a cell, you can access its `cell type` property. This property is modifiable, and can be changed to the desired type during script execution, if necessary.

The following example code demonstrates this process. This code will check the `cell type` property of the first cell of the second row of the specified table, and change it to `text content`, if it is not already. The code will then set the text content of the cell to the string `"Some Text"`.

```
tell application "QuarkXPress"
    tell table box 1 of page 1 of document 1
        if cell type of generic cell 1 of table row
    2 is not equal to text cell type then set cell type of
    generic cell 1 of table row 2 to text cell type
        set text of text cell 1 of table row 2 to
    "Some Text"
    end tell
```

```
end tell
```

Placing a Picture in a Table Cell

Likewise, when placing a picture in a cell, it is a good idea to check the type of the cell using the `cell type` property. The following example code demonstrates the process of prompting the user to choose a picture file, ensuring that a specified cell is a picture cell, and then placing the specified image into the cell.

```
set theImage to choose file with prompt "Please select
an image:" without invisibles
tell application "QuarkXPress"
     tell generic cell 1 of table row 1 of table box 1
of page 1 of document 1
          if cell type is not equal to picture cell
type then set cell type to picture cell type
          set image 1 to theImage
     end tell
end tell
```

Pulling Things Together

Now, let's pull together the topics that we have discussed throughout this month's column. Suppose I am preparing a QuarkXPress document that will contain a set of sunset photos, along with their titles. The example AppleScript code below can help to make my workflow more efficient.

```
-- Prompt the user to choose a sunset image
set theImage to choose file with prompt "Please select
a sunset image:" without invisibles

-- Prompt the user to enter a title for the image
set theTextToAdd to text returned of (display dialog
"Please enter a title for the specified image:"
default answer "A Spectacular Sunset")

tell application "QuarkXPress"
     tell page 1 of document 1

          -- Build the table
          set theTable to make new table box at
beginning with properties {bounds:{"1 in", "1.75 in",
"2 in", "6.25 in"}, column count:1, row count:2}
          tell theTable

               -- Adjust the height of the first row
               set height of table row 1 to "3 in"
```

```
                -- Hide the gridlines
                set color of every horizontal gridline
to "None"
                set color of every vertical gridline to
"None"

                -- Change the cell type of the first
cell in row 1
                tell generic cell 1 of table row 1
                     if cell type is not equal to
picture cell type then set cell type to picture cell
type

                     -- Place the image
                     set image 1 to theImage

                     -- Fit the image
                     tell image 1
                          set bounds to proportional
fit
                          set bounds to centered
                     end tell
                end tell

                -- Change the cell type of the first
cell in row 2
                if cell type of generic cell 1 of table
row 2 is not equal to text cell type then set cell
type of generic cell 1 of table row 2 to text cell
type

                -- Place the text
                set text of text cell 1 of table row 2
to theTextToAdd

                -- Justify and style the text
                tell text cell 1 of table row 2
                     set vertical justification to
centered
                     tell paragraph 1
                          set justification to centered
                          set font to "Skia"
                          set size to 28
                     end tell
                end tell
           end tell
      end tell
end tell
```

When run, this code will first prompt the user to choose a sunset image file, and then to specify a title for the chosen image. The script will then create a one-column two-row table in the frontmost opened QuarkXPress document. It will then place and fit the photo into the first cell of the first row, and then place and style the specified photo title into the first cell of the second row. The result is a table containing the specified sunset photo and title, formatted according to my needs. See figure 5.

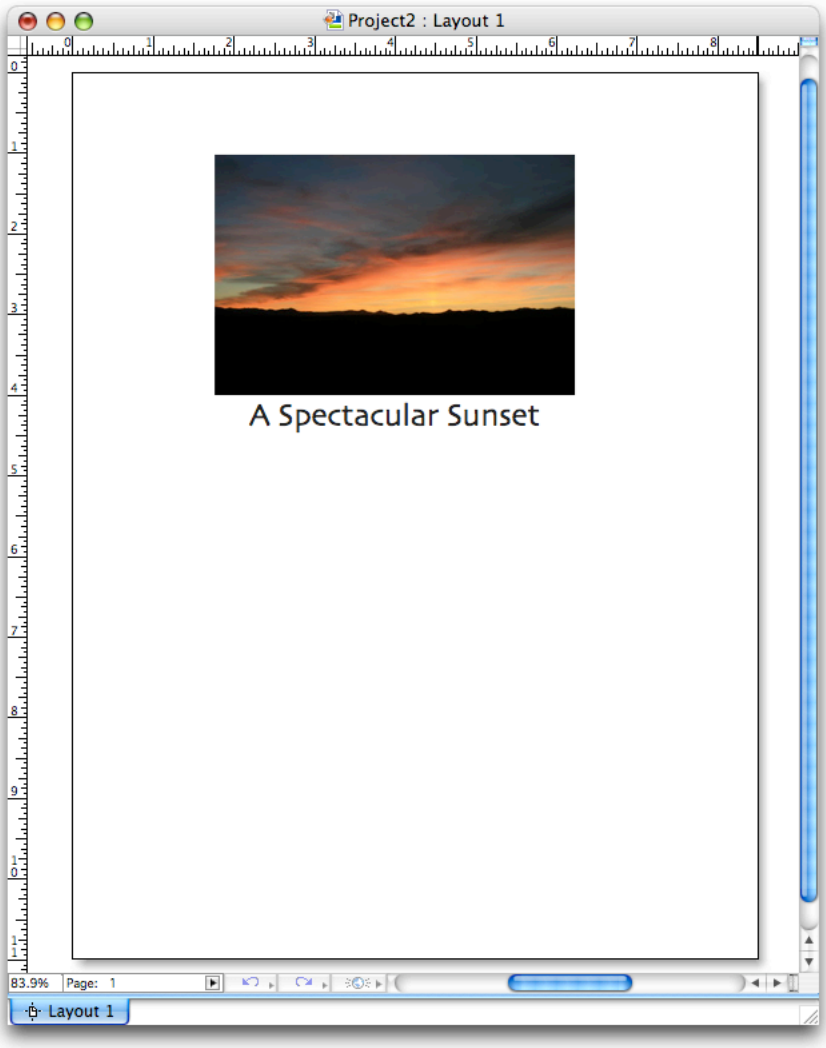

Figure 5. Example of a Completed Table

In Conclusion

If your QuarkXPress documents incorporate tables, then I encourage you to take the topics that we have discussed in this month's column, and begin to utilize them to make your own workflow more efficient.

To learn more about automating table interaction in QuarkXPress via AppleScript, spend some time becoming acquainted with the terminology in the *Table Suite* in Quark's AppleScript dictionary. For even more information, be sure to take a look at Quark's *Guide to Apple Events Scripting*, which is installed alongside QuarkXPress in the *Documents > Apple Events Scripting* folder.

See you in the trenches.