Element Movement Contracts

Element movement contracts allow you to dynamically move Drumbeat SmartElements on a web page. All movements are linear (they move at a constant velocity – a straight line without changing speed). This stuff basically falls into the category of eye-candy.

Limitations

These contracts only work in IE4.0 and above and Communicator 4.0 and above.

If you use the SmartSpacer SmartElement you *may* have problems. Let's start with the blanket statement:

> *"All element movement contracts are incompatible with pages that contain the SmartSpacer SmartElement".*

Now that that's out of the way you can generally use the element movement contracts with SmartSpacers as long as all the element movement occurs above all SmartSpacers.

The problem has to do with how browsers compute coordinates. Essentially each chunk of browser screen real estate (client area) has a separate coordinate system, starting at the origin of (0,0). So, if you try to move elements across SmartSpacers or even below a single SmartSpacer the coordinate computations get a little screwy. Then again, you may like what it does. There really is no harm done by applying these contracts in these cases, you just probably won't get what you want.

Finally these contracts do not work with form elements (edit box, buttons, list boxes, etc.) in Communicator 4.0. Netscape Communicator support for moving form elements is a little weak.

Parameters

These contracts share a number of parameters, which are described here.

Speed

This is the speed at which the elements move. There are three choices: Slow, Medium, and Fast, which are admittedly subjective values. See the implementation notes below to customize these speeds to suit your own tastes.

Location

This parameter is either labeled "From" or "To" depending on the specific contract. This is the off-screen position and there are eight choices. The off-screen position is computed on the client at run time. Note that in IE4.0 any off-screen position below or to the right causes scroll bars to appear in the browser. You can actually scroll the supposedly off-screen element into view.

This parameter is a bit harder to customize. My best advice here is to "go to the source, Luke." The code that computes the off-screen position is contained in the support script getOffscreenPos. You can modify that routine to do your bidding – as far as off-screen positions are concerned.

Left and Top

These are in absolute pixel coordinates.

Implementation and Customization Notes

The speed of the element movement contracts is in terms of the number of seconds to complete the movement. The support script DurationArray contains the values associated with each of the available speed values in the speed Parameter. The support script looks like this:

```
DurationToSeconds = new Array(3);
DurationToSeconds["Slow"]   = 6;
DurationToSeconds["Medium"] = 3;
DurationToSeconds["Fast"]   = 1;
```

To customize the values just change the numeric value associated with the string. For example to make "Slow" twice as slow, change the value of 6 to a 12 – meaning 12 seconds to complete the movement.

The edge transition contracts use this array also, so changing speeds here will affect the reveal / conceal speed of the edge transition contracts.