Email from Paul Spencer to the Fusion-Dev mailing-list
about the current status of Jx and it's Future

For those of you wondering about the changes in the Jx API, I thought I would give a bit more detail on that subject in a separate thread (this one) because we really haven't talked about Jx much before (and I'd like more people to be interested in it!). This is quite a long email, but hopefully informative.

First, a bit of an introduction to Jx for those that are unfamiliar with it.

Jx is a toolkit for building web applications. It bears some similarity to ExtJS, GWT, YUI and other such toolkits, but takes a somewhat different approach.

First, the similarities ...

Jx, just like the other toolkits, comprises a set of building blocks that assist a web developer in creating appealing web-based applications. Typical building blocks are buttons, menus, tabs, dialogs, panels and layout controls. You pick and choose which components you need and plug them together with your content and custom application logic. The programming APIs are even somewhat similar once you are familiar with each toolkit. The toolkits also provide you with a consistent visual style for all the components that you are using, something that really makes an application more pleasant and easy to use for your end user.

Jx provides the following components:

Buttons

Button - a simple clickable button
Flyout Button - exposes a panel when the button is clicked
Color Button - exposes a color picker when the button is clicked
Multi Button - a palette of buttons, one of which is in the toolbar - it can be changed by choosing a new button from a flyout, similar to a photoshop concept
Combo Button - an editable selection list
Button Set - not a visual widget, but groups toggle buttons together so that only one can be active at a time

Tabs

Tab - a button that shows a content area when activated and hides it when deactivated.
Tab Box - a combined tab bar and content area that you can put tabs into
Tab Set - manages a set of tabs so that only one is active at a time

Menus

Menu Item - an entry in a menu
Sub Menu - a menu that goes in another menu
Menu - a menu that can contain menu items and sub menus, can be put in a toolbar
Context Menu - a menu that can be shown anywhere, usually triggered by a right mouse click

Trees

Tree Item - an entry in a tree
Tree Folder - a folder in a tree, it contains folders and items
Tree - a root organizer for a tree

Panels

Panel - a basic layout element with a title bar optional toolbars and a content area

Panel Set - manages a set of Panels in a vertical area such that the panels can be resized and consume the vertical space of the container (similar to the Outlook bar or an accordion)

Dialog - a floating panel that looks like a panel but can be moved and resized

Grids

Grid - a tabular control that has fixed headers for rows and columns and scrollable content.

Layout Controls

Splitter - a control that splits an HTML element horizontally or vertically and allows the user to resize the split areas.

Layout - a control that manages the size of elements based on a set of rules

Other toolkits provide similar building blocks, often they provide much more than Jx does.

Now the differences ...

When we first started working on Jx, there really was no other alternative and the other well known toolkits didn't exist. At the core of Jx is the guiding principle that we should leverage the browser's layout engine and innate capabilities as much as possible. A corollary to this principle was to use semantically correct HTML structures whenever possible.

Having a thorough working knowledge of how HTML and CSS work in various browsers has allowed us to stick reasonably closely to this principle. We don't use unnecessary HTML elements and we don't use CSS hacks (anymore). This is not to say that we have adhered perfectly to this, but as we discover new techniques to do away with code or minimize the HTML structure we try to include them as soon as possible.

As a practical example, the ExtJS toolkit (not to pick on them, but they are popular and will provide a good example) provides a basic button object that is visually and functionally very similar to the Jx button object.

The generated HTML from a simple button with just an image in it:

<table cellspacing="0" cellpadding="0" border="0" class="x-btn-wrap x- btn x-btn-icon" id="ext-comp-1034" style="width: auto;"><tbody><tr><td class="x-btn-left"> </td><td class="x-btn-center"><button type="button" class="x-btn-text" id="ext- gen52" style="background-image: url(list-items.gif);"> </button>< /td><td class="x-btn-right"> </td></tr></tbody></table>

Note that the structure of the button is created using a table and there are 8 separate elements involved in the button structure

The equivalent Jx button:

</ div>

The structure of the button contains 4 elements and yet provides the same visual style and functionality.

In addition to the basic structure, the ExtJS button uses javascript to trap the mouse over and out events and apply a class to the table to give it a hover effect. Jx uses pure CSS by using the a:hover selector to achieve the identical result, except the javascript interpreter doesn't have to fire up and run code.

While this may not seem like a big difference, it does make a difference when you have lots of buttons and other objects using half or less of the HTML structure and a great many less event handlers.

All of this isn't to say that ExtJS is a bad toolkit - far from it, they have many advantages over Jx including fantastic looking styles and a much bigger set of great components (the grid control is extremely good!). We don't have those in Jx. But it is useful to point out why Jx (still) exists - believe me, we have discussed dropping Jx several times in favour of some of these other toolkits.

So, now a bit more about what has changed.

The fundamental change that I wanted to make was to change the underlying javascript library that Jx uses from Prototype and Scriptaculous to MooTools.

We started using Prototype because I really liked the way it was built - there were some alternatives at the time but this one seemed to be the simplest to use and understand. Scriptaculous is a separate library built on Prototype to provide visual effects.

As the state of the art in javascript libraries improved, I became less satisfied with Prototype and especially Scriptaculous. They didn't seem to be progressing at the same rate as other libraries, and were starting to feel quite heavy. Jx has a fair amount of code and we were becoming sensitive to its size. There were a couple of alternatives, primarily jQuery and MooTools. In the end, I decided to go with MooTools because it was very similar to Prototype, incorporated some excellent visual effects, and was substantially smaller.

The mechanics of switching from Prototype to MooTools was fairly simple due to a similarity in style (MooTools was written by folks that had used Prototype), and we were able to take advantage of some new functions in MooTools to eliminate quite a bit of code (mostly around class/object structure, HTML element creation and event management).

In addition to switching over to MooTools, we decided to have a thorough review of the current API, object structure and corresponding CSS. This resulted in several weeks of tweaking and reworking various objects, CSS, images etc until we came up with the current state. The result is that the code is a lot cleaner, the API is more consistent (there is still more that could be done here), and we use more techniques for layout that avoid the need for javascript assistance.

Buttons are a basic building block of most applications. In the previous version of Jx, there were several things that looked like buttons but were actually implemented with different code because we didn't see

the core similarities the first time around. These have all been changed to use the basic button class and extend it with new behaviours, eliminating a lot of code and unnecessary complexity in designing CSS for button-like things. In acknowledgement of this change, things that are button-like are now in a slightly different name space. For instance, where you would have created a new Jx.Tab in the previous version, now you create a new Jx.Button.Tab.

One of the immediate advantages that we derived from this was that all button-like things now take exactly the same options for labels, images, tooltips etc. Whereas before, we didn't have a way of including images on Tabs, now Tabs are essentially a button with a different surrounding chrome and the image support comes naturally through the button code. So now all these button-like controls all work exactly the same way.

Tabs, Menu Items and all the button types now use the basic button structure and code through inheritance.

We've done a similar thing with Dialogs and Panels. In the previous version, these were two separate code bases. Now, panel contains the basic layout code and dialog just uses it with some different chrome and some new behaviour for being able to move and resize it. Again, a big reduction in code and now they both behave the same way.

The other optimization here was replacing the TabBox with a Panel, not as substantial an improvement but still providing more consistency.

We also worked quite a bit on looking that the javascript API exposed for each component so that working with each would be similar. We've largely settled on the technique of using an options object as a way of passing arguments to object constructors. This technique is used by many other toolkits too.

Thanks to those that managed to read this far! We have already started to incorporate this new version of Jx in Fusion as I indicated in another email. But Jx is really its own entity and deserves some of its own limelight. In the coming weeks, we will be putting the finishing touches on the new version including some documentation and examples, and launching it on google code. I'll be sure to make an announcement here when that happens.

In the mean time, if you have any questions about Jx, please feel free to ask on this list. It is still a bit of a work in progress and we have lots of little things left to do.

The work on Jx is the effort of several people at DM Solutions Group, primarily myself and Fred Warnock.

Cheers

Paul

Paul Spencer
Chief Technology Officer
DM Solutions Group Inc
http://www.dmsolutions.ca/
