

MX

developer's journal

volume 2 issue 5 www.mxdj.com



THE LEADING MAGAZINE
FOR MACROMEDIA MX
DEVELOPERS & DESIGNERS

extending flash

 **DREAMWEAVER**
Visual Formatting

 **FIREWORKS**
Do It with Style

 **FREEHAND**
The Object Panel

 **COLDFUSION**
Making Headlines

 **DIRECTOR**
Talking to Sprites



\$9.99US \$9.99CAN



0 09281 02978 6 06 >



Visual Formatting
The CSS transition made easier
by zoe gillenwater



Extending Flash
XML2UI and Flash dialog boxes
by guy watson



Do It with Style
A powerful time-saving tool
by charles e. brown

16



28



42



12

Have It Your Way
Custom keyboard shortcuts in Dreamweaver MX 2004
by justin kozuch

38

A New Solution for Flash Remoting
Integrating Flash clients with server-side components
by joe orbman

24

XML for Web Designers
Leveraging Macromedia support
by kevin ruse

7

Why Use CSS?
The benefits offer the best reason
by greg rewis





The Object Panel
The key to functionality in FreeHand MX
 by ron rockwell



Making Headlines
CFMX: A Web services example, Part 2
 by richard gorremans



Talking to Sprites
Keeping tabs on Director
 by james newton

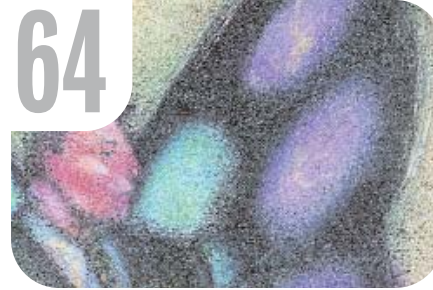
46



54



64



on the cover

60

MOA Tour
Macromedia Open Architecture quirks, clues, shortcuts, and hints
 by tab julius

72

Ending CPU Hogging
Sleep that offers work
 by alex zavatore

With Flash MX 2004's new Extensibility JSAPI, it's now possible to provide cross-platform dialog boxes in your Flash extensions. This article is an in-depth exploration of XML dialog boxes, the XML2UI Engine that parses and displays them, and XML2UI itself, an XML-formatted language.



14

xile
Cartoon
 by louis f. cuffari

74

vanguard
Dream Out Loud
 by nino del padre

Dreamweaver Website Development

MANY needs - ONE solution



MX Kollection for ColdFusion and PHP

Create powerful dynamic lists

- Sort, filter and page result sets
- Perform multiple deletes
- Easily create Master/Detail lists

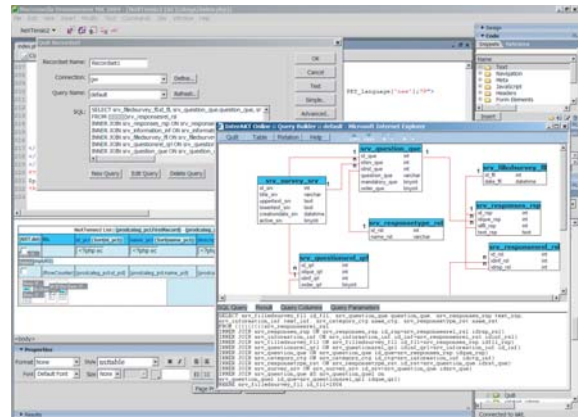
Build unified add/modify forms

- Client side and server side validation
- Insert into two tables
- Create form actions such as send mail or delete related record
- File/Image upload and resize

Create recordsets visually

- Build complex queries across multiple tables quickly

... and many more



The MX Kollection is a **suite of extensions** designed to **change the way you create dynamic web applications** with Dreamweaver MX.

ColdFusion and **PHP** developers will find our product enabling them to visually develop **e-Commerce, CMS, CRM, Backend** and other web solutions with increased efficiency, quality and capability.

Our customers think of it as **the next level in Dreamweaver MX visual software development.**

Download the demo and see the features and benefits of our extensions:

<http://www.interaktonline.com/>

MX Kollection - Professional web tools



Group Publisher Jeremy Geelan
Art Director Louis F. Cuffari

EDITORIAL BOARD
Dreamweaver Editor
Dave McFarland
Flash Editor
Jesse Warden
Fireworks Editor
Kleanthis Economou
FreeHand Editor
Louis F. Cuffari
Ron Rockwell
ColdFusion Editor
Robert Diamond

INTERNATIONAL ADVISORY BOARD
Jens Christian Brynildsen **Norway**,
David Hurrows **UK**, Joshua Davis **USA**,
Jon Gay **USA**, Craig Goodman **USA**,
Phillip Kerman **USA**, Danny Mavromatis **USA**,
Colin Mook **Canada**, Jesse Nieminen **USA**,
Gary Rosenzweig **USA**, John Tidwell **USA**

EDITORIAL
Executive Editors
Gail Schultz, 201 802-3043
gail@sys-con.com
Jamie Matusow, 201 802-3042
jamie@sys-con.com

Editors
Nancy Valentine, 201 802-3044
nancy@sys-con.com
Jean Cassidy, 201 802-3041
jean@sys-con.com
Jennifer Van Winckel, 201 802-3052
jennifer@sys-con.com

Technical Editors
James Newton • Sarge Sargent

To submit a proposal for an article, go to
<http://grids.sys-con.com/proposal>.

Subscriptions
E-mail: subscribe@sys-con.com
U.S. Toll Free: 888 303-5282
International: 201 802-3012
Fax: 201 782-9600
Cover Price U.S. \$5.99
U.S. \$29.99 (12 issues/1 year)
Canada/Mexico: \$49.99/year
International: \$59.99/year
Credit Card, U.S. Banks or Money Orders
Back Issues: \$12/each

Editorial and Advertising Offices
Postmaster: Send all address changes to:
SYS-CON Media
135 Chestnut Ridge Rd.
Montvale, NJ 07645

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ

List Rental Information
Kevin Collopy: 845 731-2684,
kevin.collopy@edithroman.com,
Frank Cipolla: 845 731-3832,
frank.cipolla@epostdirect.com

Promotional Reprints
Kristin Kuhnle, 201 802-3026
kristin@sys-con.com

Copyright © 2004
by SYS-CON Publications, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system, without written permission.

MX Developer's Journal (ISSN#1546-2242) is published monthly (12 times a year) by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

SYS-CON Media and SYS-CON Publications, Inc., reserve the right to revise, republish, and authorize its readers to use the articles submitted for publication. MX and MX-based marks are trademarks or registered trademarks of Macromedia, in the United States and other countries. SYS-CON Publications, Inc., is independent of Macromedia. All brand and product names used on these pages are trade names, service marks or trademarks of their respective companies.

MX
developer's journal

Why Use CSS?

The benefits offer the best reason

by greg rewis

Since the launch of Dreamweaver MX 2004, I've had numerous opportunities to demonstrate its new features and power to both new and existing Dreamweaver users. As with any product demonstration, it doesn't take long before I'm singing the praises of Dreamweaver MX 2004's abilities to design and render CSS, or Cascading Style Sheets.

Recently, however, a novice user asked a question, which, quite honestly, startled me. The user simply asked, "Why should I use CSS?" I realized at that moment that while those of us who work with HTML and CSS on a daily basis are intimately familiar with the benefits, many of you are not.

CSS Beginnings

The governing body of the Web, the W3C, recommended the use of CSS in December 1996 with the ratification of the CSS Level 1 specification, which described attributes for use in HTML pages. These attributes replaced the traditional font tag and other "style" markup such as color and margins. In May 1998, the W3C ratified CSS Level 2, which introduced positioning attributes. These attributes replaced the rampant (and incorrect) usage of the table tag to design the presentation of page elements. The most recent revision to the CSS specifications is CSS 2.1, which refines some attributes and removes others that had only limited, if any, usages in current browsers.

Unfortunately, CSS has been slow in reaching broad adoption. One of the key reasons for this was the browsers, and, in turn, the Web designers building sites for these browsers. At the time of CSS ratification, Netscape Navigator was still the dominant browser, and its support for CSS was basically nonexistent. Microsoft added very limited support with version 3 of its browser, but most Web designers at the time (this author included) were

still coding their pages with Netscape as their reference platform.

Over the years, with each new version, the browser makers have expanded their support of CSS. Today, Internet Explorer 6, Netscape Navigator 7, Mozilla, Opera, and Safari all fully support CSS. That's not to say that our lives as Web designers and developers are without problems. While the above-mentioned browsers support CSS Level 2, they do so with varying degrees of compliance. In some cases, certain attributes still give you cause for frustration. In other words, you still need to follow the age-old mantra of "test, and test often." However, if you stick to core attributes of the CSS specifications, your pages will render correctly.

But why did the W3C create the CSS specification at all? What does it all mean to me as I create HTML-based Web sites and applications? In my opinion, you can divide up the need for CSS and its resulting benefits into three main areas: flexibility, rendering, and accessibility.

Flexibility

I'm sure that almost every Web designer and developer has experienced that moment of panic when, after meticulously laying out a page – complete with numerous nested tables – the client requests a "small" change. This could be something as simple as "can you move that image a little to the left," or as dramatic as "I'm not happy with those headers, can you make their font larger – and while you're at it, how about changing their color too?" If you're dealing with a limited number of pages, you can take a deep breath and spend the better part of an hour making those irritating changes. But when you're dealing with the larger sites, which have become the norm, a simple change is suddenly anything but simple.

What causes the panic in these situations? The markup, which defines the appearance of our pages, is actually part

of the pages themselves. To see an illustration, take any given page from one of your sites and count the number of font and table tags. If only you could remove this markup from the flow, or code, of the actual page – and, even better, if you could externalize it – you could make the changes in a centralized place. Sounds like a job for CSS.

By designing your pages using a single, or even multiple, external style sheets, you can apply those changes to your site by modifying the style sheet and uploading the modified version.

Imagine how difficult it would be to move your site navigation from the left side of the page to the right in a traditional table-based layout. This would take hours of repetitive and very tedious work. If, however, you used the positioning attributes of CSS (more commonly known as CSS-P) to design your pages, a simple change to the “float” attribute in the external style sheet would update the page. There’s also an added benefit: you’ve updated every page that uses that style sheet on the site!

Rendering

Since broadband has become mainstream, many developers have stopped

considering how much time it takes to render a page in a browser. However, for many of you it’s important to remember that your target audiences still surf the Web on dial-up connections. The traditional table-based layout is one of the primary causes of slow-loading pages. This happens because the browser, upon receiving the page from the server, must first examine and “understand” the complex array of nested tables. It must first locate the most deeply nested piece of content and then meticulously work its way back out of the code until it reaches the uppermost container, the body tag. Only after completing this journey can the browser begin rendering the content on the screen.

When you use CSS, the browser begins the rendering process as soon as it receives the content from the server because there is little, if any, actual presentational markup in the page.

There is also a hidden rendering benefit when using external style sheets. In the traditional table-based approach, the browsers must retrieve, analyze, and render each page individually. In other words, the browser is working just as hard at displaying the 30th page in your site as it was when displaying the first page.

If, however, the site uses external style sheets for its presentation, the first page prompts the browser to cache the linked style sheet files that the page uses. This means that all of the subsequent pages in the site using those style sheets will load even faster, since the browser has already cached the style sheets.

The final rendering benefit reminds me of the movie



SYS-CON MEDIA
President & CEO
 Fuat Kircaali, 201 802-3001
 fuat@sys-con.com
Vice President, Business Development
 Grisha Davida, 201 802-3004
 grisha@sys-con.com
Group Publisher
 Jeremy Geelan, 201 802-3040
 jeremy@sys-con.com

ADVERTISING
Senior Vice President, Sales & Marketing
 Carmen Gonzalez, 201 802-3021
 carmen@sys-con.com
Vice President, Sales & Marketing
 Miles Silverman, 201 802-3029
 miles@sys-con.com
Advertising Sales Director
 Robyn Forma, 201 802-3022
 robyn@sys-con.com
Director, Sales & Marketing
 Megan Mussa, 201 802-3023
 megan@sys-con.com
Associate Sales Managers
 Kristin Kuhnle, 201 802-3025
 kristin@sys-con.com
 Beth Jones, 201 802-3028
 beth@sys-con.com

PRODUCTION
Production Consultant
 Jim Morgan, 201 802-3033
 jim@sys-con.com
Lead Designer
 Louis F. Cuffari, 201 802-3035
 louis@sys-con.com
Art Director
 Alex Botero, 201 802-3031
 alex@sys-con.com
Associate Art Director
 Richard Silverberg, 201 802-3036
 richards@sys-con.com
Assistant Art Director
 Tami Beatty, 201 802-3038
 tami@sys-con.com

SYS-CON.COM
Vice President, Information Systems
 Robert Diamond, 201 802-3051
 robert@sys-con.com
Web Designers
 Stephen Kilmurray, 201 802-3053
 stephen@sys-con.com
 Christopher Croce, 201 802-3054
 chris@sys-con.com
Online Editor
 Lin Goetz, 201 802-3045
 lin@sys-con.com

ACCOUNTING
Accounts Receivable
 Charlotte Lopez, 201 802-3062
 charlotte@sys-con.com
Financial Analyst
 Joan LaRose, 201 802-3081
 joan@sys-con.com
Accounts Payable
 Betty White, 201 802-3002
 betty@sys-con.com

EVENTS
President, SYS-CON Events
 Grisha Davida, 201 802-3004
 grisha@sys-con.com
Conference Manager
 Lin Goetz, 201 802-3045
 lin@sys-con.com
National Sales Manager
 Sean Raman 201-802-3069
 raman@sys-con.com

CUSTOMER RELATIONS
Circulation Service Coordinators
 Shelia Dickerson, 201 802-3082
 shelia@sys-con.com
 Edna Earle Russell, 201 802-3081
 edna@sys-con.com
 Linda Lipton, 201 802-3012
 linda@sys-con.com
JDJ Store Manager
 Brundila Staropoli, 201 802-3000
 bruni@sys-con.com

BlueDragon [6.1]

RELEASED!



RUN YOUR CFML:

Within BlueDragon Server,
or as a native .NET or J2EE
web application.

On the platform, web server,
and operating system of
your choice.



Get BlueDragon hosting:

CFDynamics is now offering BlueDragon hosting!
Find out more by visiting: www.cfdynamics.com/bluedragon

As one of the ColdFusion hosting community leaders, CFDynamics is constantly looking for ways to deliver new and cutting edge technologies to the ColdFusion community. We are excited to announce our premiere partnership with New Atlanta by offering BlueDragon hosting. We look forward to seeing how BlueDragon expands the possibilities of ColdFusion. Come join CFDynamics and New Atlanta in expanding the ColdFusion horizon!

CFDynamics

A Division of Konnections Inc.



POWERFUL HOSTING PLANS

- FREE SQL server access
- FREE account setup
- UNLIMITED email accounts
- GENEROUS disk space / data transfer
- 30 day MONEY-BACK GUARANTEE
- GREAT VALUE!

RELIABLE NETWORK

- 99.99% average uptime!
- State-of-the-art data center has complete redundancy in power, HVAC, fire suppression, bandwidth, and security
- 24 / 7 network monitoring

FANTASTIC SUPPORT SERVICES

- Comprehensive online support
- Knowledgeable phone support
- We focus on your individual needs

**SIGN UP FOR A HOSTING
ACCOUNT TODAY!**

Use promo code: **MXDJ04E**
and receive a **FREE T-SHIRT!**



WWW.CFDYNAMICS.COM

866 - CFDYNAMICS

866-233-962-6427

MXDJ Section Editors

Dreamweaver

Dave McFarland

Author of Dreamweaver MX 2004: The Missing Manual, Dave can be relied upon to bring Dreamweaver MX to life for MXDJ readers with clarity, authority, and good humor.



Flash

Jesse Warden

A multimedia engineer and Flash developer, Jesse maintains a Flash blog at www.jessewarden.com and says, referring to the MX product range, that "Things are changing, opportunity is on the frontier, a paradigm shift is occurring for Web design, Web applications, et al."



Fireworks

Kleanthis Economou

A Web developer/software engineer since 1995, now specializing in .NET Framework solutions, Kleanthis is a contributing author of various Fireworks publications and is the technical editor of the Fireworks MX Bible. As an extension developer, he contributed two extensions to the latest release of Fireworks.



FreeHand

Louis F. Cuffari

Cofounder and art director of Insomnia Creations (www.insomniacreations.com), Louis has spent most of his life as a studio artist, including mediums from charcoal portraits to oil/acrylic on canvas. In addition to studio art, he has been involved in several motion picture projects in the facility of directing, screenwriting, and art direction. Louis's creative works expand extensively into graphic design, and he has expertise in both Web and print media. He is deputy art director for SYS-CON Media and the designer of MX Developer's Journal.



Ron Rockwell

Illustrator, designer, author, and Team Macromedia member, Ron Rockwell lives and works with his wife, Yvonne, in the Pocono Mountains of Pennsylvania. Ron is MXDJ's FreeHand editor and the author of FreeHand 10 f/x & Design, and coauthor of Studio MX Bible and the Digital Photography Bible. He has Web sites at www.nidus-corp.com and www.brainstormer.org.

ColdFusion

Robert Diamond

Vice president of information systems for SYS-CON Media and editor-in-chief of ColdFusion Developer's Journal, Robert was named one of the "Top thirty magazine industry executives under the age of 30" in Folio magazine's November 2000 issue. He holds a BS degree in information management and technology from the School of Information Studies at Syracuse University. www.robertdiamond.com



Amadeus. In the movie, Mozart asks the emperor what he thought of one of his operas. The emperor responds that it was good, but tedious. When pressed by Mozart, the emperor explains that the problem was simply that there were "too many notes." With Web design, this can also be a problem – the notes are the actual HTML code. The more code there is, the longer it takes the browser to understand and make sense of the page.

When you implement CSS in your designs, you decrease the amount of code the client needs to download. Simply removing all of the font tags from some pages can minimize the amount of code dramatically. If you move to completely CSS-P designs, in many cases you can minimize the amount of code by 50% or more! Less code equals faster-loading pages.

Accessibility

I hear a lot about accessibility these days. Most developers know that they should be thinking about building more accessible sites, but to a large degree only those developers who build sites for government or educational institutions have been forced into actually doing it. When thinking about accessibility, a majority of developers assume this means that they simply need to add things like alt attributes to their images. But there is actually much more to accessibility, and using CSS can make it easier for you to build accessible sites.

One of the primary issues of accessibility – and one where CSS can really make a difference – is in how an assistive technology such as a screen reader "reads" a page. In the traditional table-based world, a screen reader faces an incredible challenge in deciding how to read a page. Think about how confusing it must be for a screen reader as it encounters a deeply nested table – should it read the content, or skip over it? And if it skips over it, how does it get back to the content later?

As you hit a page, you can quickly spot the content that interests you and ignore the navigation or other content at the top of the page. Visually impaired people don't have this luxury. They must

wait for the screen reader to parse through all of the extraneous information between the top of the page and the content they are really interested in.

Of course, there are techniques to make the screen reader skip the navigation, but these usually require adding links to images in your navigation bar or other content. While these techniques work, they can also be confusing, and sighted visitors can see them as well. Using CSS, you can define completely invisible elements on the page – elements that are invisible to other site visitors and your mouse. The screen reader can use these elements to navigate quickly and effectively through the document.

With CSS and its lack of presentational markup, the only thing that a screen reader encounters is actual content. Additionally, as you design using CSS-P, you begin to concentrate on the actual "flow" of content and consider its logical order on the page.

As you've been reading this article you followed the "flow" of information. But in that nested table example, this paragraph could just as easily have been in the upper right-hand corner of the page. In that case, the screen reader would have no way of knowing that it should wait until the end of the article to read it.

Using CSS-P, the browser could display this paragraph in the upper right corner of the page, but its position in the actual text or flow of the document would still be right here where you are seeing it. This makes for a much better, more accessible experience.

• • •

There you have it. I hope I've explained some of the unique benefits of using CSS in your Web endeavors. Obviously, there's a lot to learn. ☺

Greg Rewis is chief Web technologies evangelist at Macromedia. It is Greg's responsibility to be a public spokesperson for the Macromedia Web publishing suite of software and Web application development servers – as well as to represent the company's customers in an advocate's role on the product development teams.
gregwis@macromedia.com

Complete source code and asset management in Dreamweaver MX—now possible with Surround SCM.

Dreamweaver users know a beautiful Web-based product is only skin deep. Underneath, it's a tangle of hundreds or thousands of ever changing source files. Without a good development process and strong tools, bad things happen. Surround SCM can help.

Surround SCM lets you...

Track multiple versions of your source files and easily compare and merge source code changes.

Check out files for exclusive use or work in private workspaces when collaborating on a team.

Automatically notify team members of changes to source files—push changes through your organization.

View complete audit trails of which files changed, why, and by whom.

Associate source code changes with feature requests, defects or change requests (requires additional purchase of TestTrack Pro).

Remotely access your source code repository from Dreamweaver MX.

Surround SCM adds flexible source code and digital asset control, powerful version control, and secure remote file access to Dreamweaver MX. Whether you are a team of one or one hundred, Surround SCM makes it easier to manage your source files, letting you focus on creating beautiful Web-based products.

Features:

Complete source code and digital asset control with private workspaces, automatic merging, role-based security and more.

IDE integration with Dreamweaver MX, JBuilder, Visual Studio, and other leading Web development tools.

Fast and secure remote access to your source files—work from anywhere.

Advanced branching and email notifications put you in complete control of your process.

External application triggers let you integrate Surround SCM into your Web site and product development processes.

Support for comprehensive issue management with TestTrack Pro—link changes to change requests, bug reports, feature requests and more.

Scalable and reliable cross-platform, client/server solution supports Windows, Linux, Solaris, and Mac OS X.

Achieve major improvements in Web and e-business development performance through better tool integration and process automation. Gain complete control over your source code and change process with Surround SCM. Manage defects, development issues, and change requests with award-winning TestTrack Pro. Completely automate product testing with QA Wizard. Streamline your development process with Seapine tools and help your team deliver quality software products on time, every time.

Learn more about
Surround SCM at
www.seapine.com
or call 1-888-683-6456



macromedia
ALLIANCE PARTNER



d

f

fu

fb

fd

db

Have It Your Way

Custom keyboard shortcuts in Dreamweaver MX 2004

by justin kozuch

While dialog boxes and pull-down menus are one way to insert HTML, all that extra mousing around takes time. After the tenth time you choose Insert > Image Objects > Rollover Image, your aching hand will need some relief and you'll be wondering if there's a faster way to get your work done. Fortunately, the customizable nature of Dreamweaver makes most commands just a key tap away.

What Are Shortcuts?

Shortcuts are an easy way to access menu functions without having to use your mouse. They can be used to perform simple tasks such as opening or saving a file, or more complex tasks such as moving or copying files from one location to another.

Why are shortcuts so important? Well, there are a few reasons. First, from an ease-of-use standpoint, you don't need to access a menu item (or more than one

menu item) to use that feature, which makes the keyboard shortcut concept a timesaver. From an accessibility standpoint, computer users who have mobility problems and therefore cannot use a mouse effectively, benefit because less movement is required to use a keyboard shortcut than to use a mouse to perform the same task.

In Macromedia Dreamweaver MX 2004, there are 199 shortcuts that can be used to accomplish almost every task imaginable, from the obvious – like creating a new file (Ctrl+N on the Windows platform, Cmd+N on the Macintosh platform) – to the shortcut used to access the Dreamweaver Help Topics (F1 on both the Windows and Macintosh platforms). However, you will notice over time that not every menu item in Dreamweaver 2004 has a shortcut assigned to it.

In this article, you'll learn how to create your own shortcuts, how to share your finished shortcut set with other Web developers, and where to find useful shortcut resources.

Library/Application Support/Dreamweaver MX 2004/Configuration folder.) I say four because that's the number of default keyboard shortcut sets that Dreamweaver MX 2004 will install.

Once the Keyboard Shortcuts dialog box has opened, you will see something very similar to Image I.

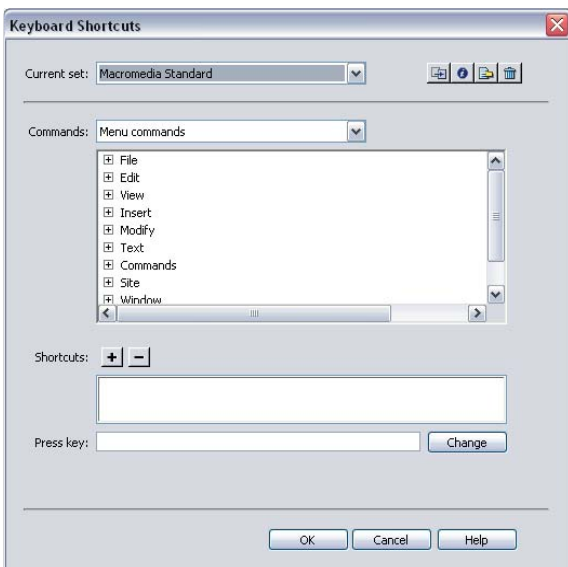
Since Dreamweaver MX 2004 will not allow you to edit the Macromedia Standard keyboard shortcut set through the Keyboard Shortcuts dialog box, the first thing we need to do is create a duplicate shortcut set.

Click on the "Duplicate set" button (see Image II), and type a name for the new shortcut set in the Dialog Box. You can type in any name for the set as long as it is fewer than 27 characters. Use any punctuation marks that you like, such as commas or exclamation marks. Click the OK button and select the name of the new keyboard shortcut set from the Current Set drop-down menu. A dialog box (see Image III) will tell you that you are changing keyboard shortcut sets. Click on the OK button to close the dialog box.

Setting Everything Up

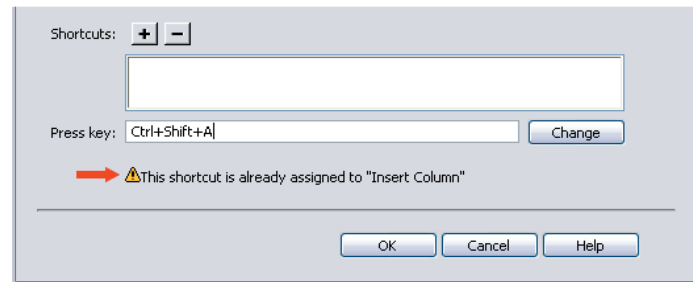
Fire up Dreamweaver MX 2004 and select Edit (Alt + E) > Keyboard Shortcuts (Dreamweaver > Keyboard Shortcuts... for Macs). This will launch the Keyboard Shortcuts dialog box, which might take a minute or two depending on the speed of your computer and other variables. The data being loaded comes from the menus.xml file located in the C:\Program Files\Macromedia\Dreamweaver MX 2004\Configuration\Menus directory and the four XML files located in the C:\Program Files\Macromedia\Dreamweaver MX 2004\Configuration\Menus\Custom Sets directory. (On the Mac these folders are located in

***Tip:** The XML file for your new keyboard shortcut set is not stored in its regular location. Instead, it is stored in the C:\Documents and Settings\username\Application Data\Macromedia\Dreamweaver MX 2004\Configuration\Menus\Custom Sets directory. Be sure to replace username in this directory path with your user name. On a Mac this is stored in your home folder in the Library/ApplicationSupport/Macromedia/Dreamweaver MX 2004/Configuration/Menus/Custom Sets folder.*



Taking a Shortcut

In the “Commands” drop-down menu, you will see six options: Menu commands, Site panel, Code editing, Document editing, Site window, and Snippets (on a Mac there are just four options: Menu commands, Code editing, Document editing, and Snippets). Select “Menu commands” from the drop-down menu, and in the text box below you’ll see a list of all the menus in Dreamweaver MX 2004. Next to the name of the menu, you’ll see a plus sign. This indicates the presence of an expandable list. Go ahead and click the + sign next to the word “File”. What we are going to do is assign a shortcut to the “Save All” menu item. Select the “Save All” item and then click the + sign next to the word “Shortcuts”. Place your cursor in the “Press key” text field and press the following keyboard combination: “Ctrl+Shift+A”.



Tip: Depending on the Current set you have selected, the shortcuts for this set are loaded from the related XML file. For instance, if you have loaded the Macromedia Standard shortcut set, the shortcuts are read from the “Macromedia Standard.xml” file in the C:\Program Files\Macromedia\Dreamweaver MX 2004\Configuration\Menus\Custom Sets directory.

If you open this file in Notepad or SimpleText, you will see something similar to:

```
<SHORTCUTSET name="Macromedia
Standard" type="factory">
  <SHORTCUT ID="DWMMenu_File_New"
keys="Cmd+N" />
```

The first line denotes the name of the Shortcut set (in this case, “Macromedia Standard”) and the type, which in this case is “factory”, which means that this is the default set that Dreamweaver MX 2004 uses.

The second line denotes the Menu ID (which in this case is, “DWMMenu_File_New”), which tells Dreamweaver MX 2004 where this menu item is located. In this case, the menu item is located in the “File” menu and the menu item is called “New”. The keys attribute assigns a keyboard combination value that, when pressed, launches the New File dialog box.

If you enter a keyboard shortcut that is already in use, Dreamweaver MX 2004 will tell you via a warning that shows up underneath the text field (Image IV). Since “Ctrl+Shift+A” is being used to insert a column in a table, we need to choose another one. Place your cursor in the text field and enter “Alt+Ctrl+Shift+S”; then click on the “Change” button. Your new keyboard shortcut will show up in the text box listing all the keyboard shortcuts, and in the detail box below. If you want to delete the shortcut you just created,

simply select the shortcut from the detail box and click on the – sign to delete it.

If you write a lot of your own code, you’re probably familiar with Dreamweaver’s Snippets feature, a tool for adding commonly used code such as database connection strings, or frequently used copyright notices, with the click of a button. MX 2004 now lets you assign keyboard shortcuts to your favorite snippets.

Sharing Your Shortcuts

Now that you’ve mastered the art of creating a shortcut set, you can actually share the set you just created with other people, like a development team or an IT department.

The location of your newly created custom keyboard shortcut set will differ depending on the operating system you are using:

- Windows XP/XP Pro: C:\Documents and Settings\username\Application

“Shortcuts are an easy way to access menu functions without having to use your mouse”

Data\Macromedia\Dreamweaver MX 2004\Configuration\Menus\Custom Sets

- Macintosh OS X: Hard Drive:Users:user-name:Library:Application Support: Applications:Macromedia Dreamweaver MX 2004\Configuration:Custom Sets


The file of your keyboard shortcut set will be identical to the name you just gave your keyboard shortcut set. To share the keyboard shortcut set you have just created, simply navigate to the corresponding folder above and copy the XML file and paste it into the corresponding directory on the machine you would like

tents of sorts for your keyboard shortcuts. Open the Keyboard Shortcuts dialog box using Edit (Alt + E) > Keyboard Shortcuts (Dreamweaver > Keyboard Shortcuts... for Macs), and click on the "Export set as HTML" button (it's the third button from the left). Specify a location to save the HTML file to, enter a filename in the "Filename" text field, and click the OK button. You can print this out, or distribute it with the custom keyboard shortcut set.

Conclusion

Using the Keyboard Shortcuts dialog box is just one of the many ways you can extend the use of Dreamweaver MX 2004.

download the Dreamweaver MX 2004 Quick Reference Guide from www.macromedia.com/support/documentation/en/dreamweaver/index.html

In addition, Danilo Celic of Community MX has written a Dreamweaver MX/MX 2004 extension that will automatically place the extension in the proper place on your hard drive depending on the operating system you are using. You can download it from www.communitymx.com/abstract.cfm?cid=7EF02 

Justin Kozuch is a writer, Web developer, and Team Macromedia member who takes pride in helping other

“...there are 199 shortcuts that can be used to accomplish almost every task imaginable...”

to share the custom keyboard shortcut set with. Once you restart Dreamweaver MX 2004, go to Edit (Alt + E) > Keyboard Shortcuts (Dreamweaver > Keyboard Shortcuts... for Macs), click on the "Current set" drop-down menu, and select the name of the keyboard shortcut set. Dreamweaver MX 2004 will load the custom keyboard shortcut set.

You can also create a table of con-

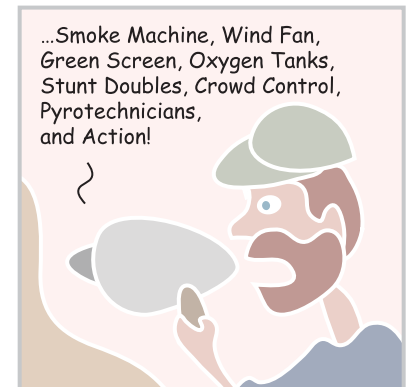
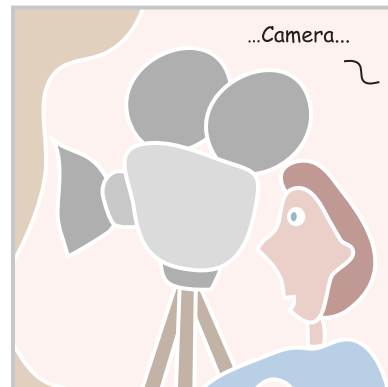
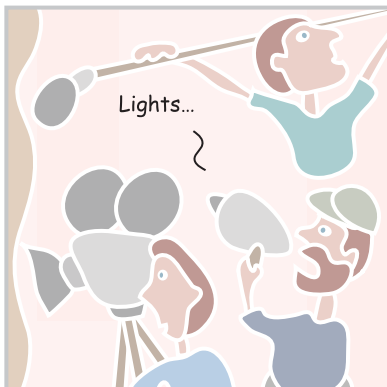
No longer will you have to search through the menus looking for that elusive menu item. By assigning a shortcut to one or more menu items that you use often, you are cutting down on your development time and improving your workflow.

If you would like to download a list of the keyboard shortcut sets that come with Dreamweaver MX 2004, you can

Dreamweaver users. His work is published weekly on CommunityMX.com, the home of the MX Community. He's also the founder of Dreaming in TO (www.dreaminginto.com), a Macromedia Dreamweaver User Group located in Toronto, ON. A dynamic "junkie", Justin's passion lies in PHP/MySQL, organic design, and breadcrumb navigation. justin@dreaminginto.com

xile

written & illustrated by louis f. cuffari 





It's everybody's PDF™

Finally, a software company that offers affordable yet flexible PDF solutions to meet every customer's needs. Using activePDF™ to automate the PDF creation process eliminates the need for end-user intervention so your employees can concentrate on what they do best.

Licensed per server, activePDF solutions include a COM interface for easy integration with ColdFusion. Dynamically populate PDF forms with information from a database, convert ColdFusion web pages to PDF on the fly, dynamically print reports to PDF using CF and much more. Users can also merge, stitch, stamp, secure and linearize PDF, all at a fraction of the cost of comparable solutions. Download your free trial version today!



www.activePDF.com



Visual ▶▶▶ Formatting



by zoe gillenwater

Support for cascading style sheets, or CSS, has been present in Dreamweaver for many years; you may have taken advantage of it as just another software feature without really knowing how to utilize it fully, efficiently, and correctly. This article will introduce you to some general guidelines to follow while setting up and working with CSS-based Web pages so you can achieve more consistent rendering cross-browser. It assumes you know what CSS is, have some idea of its syntax and rules, and are eager to take advantage of its benefits, but is aimed at the Web designer who has not yet taken the leap to using it as their primary layout and visual formatting method.



Choosing a Doctype and Rendering Mode

To assure that your CSS is rendered in a reliable way, you'll need to include a document type declaration, or doctype, on your pages. A doctype tells the browser which version of (X)HTML you are using and usually appears on the first line of your document. Here's an example of the default doctype Dreamweaver MX 2004 uses when you create a new blank HTML page:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd"
>
```

This doctype is complete because it contains both the public identifier (the part that states it is HTML 4.01 Transitional) and the system identifier (the URL to the document type definition, or DTD). Doctypes can also be written without the system identifier and still be valid, but you'll usually want to work with complete doctypes because of the impact this has on the way your page appears in browsers.

Your doctype determines whether your page displays in a browser's "quirks rendering mode" or in its "standards rendering mode." In quirks, a browser will try to handle sloppy authoring and will emulate the quirks and bugs of browsers of the mid- to late '90s. In standards, the browser will try to follow the World Wide Web Consortium's (W3C) current recommendations, even if the results are unexpected.

In general, the following doctypes will cause your page to be rendered in quirks mode:

- No doctype
- HTML 3.2 or earlier
- HTML 4.0 Transitional or Frameset
- HTML 4.01 Transitional or Frameset without URL

The following doctypes will cause your page to be rendered in standards mode:

- All strict doctypes
- All XHTML doctypes
- HTML 4.01 Transitional and Frameset with URL

There are a couple of exceptions, so you may want to check the major browsers' sites to verify the rendering mode for your chosen doctype.

One caveat: complete HTML 4.01 Transitional and Frameset doctypes, as well as any XHTML Transitional and Frameset doctypes, actually put Mozilla into its proprietary "almost standards mode." Almost standards is the same as standards except in the way that images in table cells and divs are rendered – in standards, images sit on the baseline of a box (where text would sit), while in almost standards they fill the entire table cell or div. Almost standards is a good choice for old-fashioned table layouts of sliced images.

Also note that an XML prologue (or anything, for that matter, even an empty comment) preceding an XHTML doctype will throw IE6 and Opera 7.0x into quirks mode. Dreamweaver MX inserted this prologue in XHTML files, but MX 2004 does not. Since it can be omitted without harm, it's recommended you just remove it if you want to stay in standards.

How do you know which doctype to choose? XHTML is not a new standard intended to replace HTML; basically it is a more precise version of HTML that is intended to make it easier to bridge over to XML in the future. If you are creating a static site, you probably don't need to worry about XHTML and can use HTML 4.01 without feeling a tad bit guilty.

What about the difference between Strict and Transitional? Strict gives you the cleanest code, the most forward compatibility, and the best separation between content and presentation because it does not allow many of the presentational tags and attributes that Transitional does. You don't need these deprecated tags any more – you can use CSS to accomplish most of the things they were used for – but if you're working with a client who insists you use some of the old tricks for the old browsers, or if you're just revising existing pages, it may be better to stick with Transitional for now. Here are a few of the elements and attributes you can't use in Strict:

- Center tag (use CSS instead)
- Font tag (use CSS instead)
- Target attribute on links (opening new windows is considered a usability no-no)

- bgcolor attribute (use CSS instead)
- Link, alink, and vlink attributes (use CSS instead)

No matter which version of (X)HTML you choose, I recommend picking a doctype that keeps you in standards mode. Standards mode usually results in a higher degree of cross-browser consistency and gets you accustomed to current rendering models so you can get in the habit of coding to the standards instead of old browser quirks. In addition, many CSS hacks depend on you using standards mode. For instance, many of the hacks designed to work around the incorrect box model in Windows Internet Explorer (IE) 5.x do not target IE6, because IE6 gets the box model right when it is in standards. However, if your page was in quirks, IE6 would have a box model problem, and since the hacks do not target IE6, it would not be fixed.

Dreamweaver automatically inserts a complete HTML 4.01 Transitional doctype on each new page, so you don't have to worry about rendering in standards. But what if you want to use HTML Strict, not Transitional, by default? Luckily, you can easily edit the default Dreamweaver HTML template. Locate the default template in the Dreamweaver Configuration folder: Configuration\Document Types\NewDocuments\Default.html. The configuration folder is located in C:\Program Files\Macromedia\Dreamweaver MX 2004\ (Win) or Library/Application Support/Macromedia/Dreamweaver MX 2004\ (Mac). Be sure to make a backup of the file (just in case). Open it in Dreamweaver and view the code (see Image I). Change the doctype to:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"
>
```

Once you save the page, all new pages will include the doctype you entered. You may want to edit some of the other default pages as well, such as the Dreamweaver Template document (Default.dwt), to include the doctype that you want.

If you choose to use XHTML to build your pages, it's easy: from the New Document dialog box, select HTML and

check the "Make document XHTML compliant" box at the lower right before clicking Create (see Image II). This will create a page with a complete XHTML Transitional doctype already inserted (see Image III). MX 2004 leaves out the XML prologue by default, so your pages should render in standards mode cross-browser. Unfortunately, there is no XHTML template for you to edit like there is for HTML, so if you would rather use XHTML Strict instead of Transitional, you're going to have to make that change by hand on each page (or use Find and Replace).

Setting Standard Rules

There are some rules that you will use over and over again in your CSS files for every site you work on. Just as you can modify the HTML template Dreamweaver uses, you can also modify its CSS template to include your standard rules. Open Default.css from the New-Documents folder (full path above) and edit away! Any changes you make and save will show up every time you create a blank CSS document from within Dreamweaver.

Here are some suggestions for rules you may want to add to your default style sheet:

```
body {
margin: 0;
padding: 0;
}
```

If you want to get rid of the default page margin on your page, remember that you need to zero out both margin and padding, since Opera uses padding rather than margin to add the extra space.

```
body {
font-face: Arial, Helvetica, sans-serif;
}
```

The specific fonts listed above are just examples, but the idea is to include your font declaration on the body. This way you can define a default font for text so you don't needlessly repeat it across several tags such as div, p, and other text-related tags that can simply inherit font properties from the body.

image I

```
Default.html
Code Split Design Title: Untitled Document
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
2 "http://www.w3.org/TR/html4/loose.dtd">
3 <html>
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset="
6 <title>Untitled Document</title>
7 </head>
8
9 <body>
10 </body>
11 </html>
```

image II

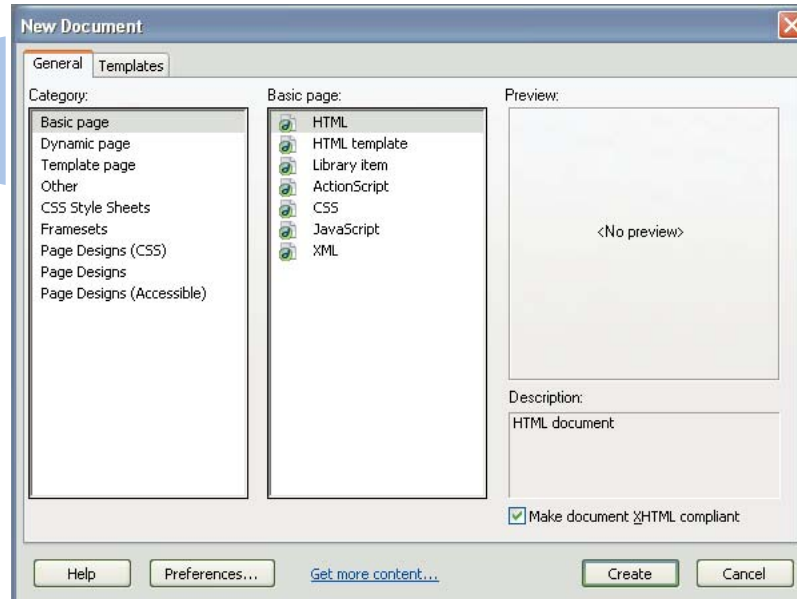


image III

```
Untitled-2
Code Split Design Title: Untitled Document
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
5 <title>Untitled Document</title>
6 </head>
7
8 <body>
9 </body>
10 </html>
11
```

image IV



```
body {
font-size: 100.01%;
}
```

If you plan on using a relative meas-

urement for text – such as ems or percentages – you'll need to use this little trick to avoid bugs in IE and Opera. IE has a bug that displays em measurements smaller than 1.0 as microscopic if the user

has his or her text size set to “Smaller” or “Smallest.” Setting a percentage font size on the body fixes this bug as long as you’re in standard mode.

If you want to set the base size to 100% to match users’ default settings (an accessibility plus), keep in mind that Opera has its own bug that computes 100% to be one pixel smaller than it should be. This means that text can become microscopic in Opera as well, since all subsequent font sizes are based off a base font that is too small. Using a value other than 100%, even 100.01%, fixes this bug in Opera.

```
table {
  font-size: 1.0em;
}
```

This rule fixes a bug in Windows IE5.x that prevents the font size from inheriting into tables as it should. The value of 1.0 em tells the table to take its font size from the surrounding content, which is exactly what it’s supposed to do anyway, so the rule doesn’t cause any damage in standards-compliant browsers either.

```
div.clear {
```

```
clear: both;
height: 0;
margin: 0;
line-height: 0;
font-size: 1px;
}
```

If you use floats to lay out divs, you are going to need a clearing element at some point to hold things together. When you float an object, you remove it from the flow of the document, so it does not affect other block elements, only inline content. This means that the float’s parent element will not expand to contain the float, as you might expect. To force the parent to contain the float, insert a clearing element within the parent but after the float. The parent will have to expand down to hold the cleared element, containing the float in the process.

Although a simple `<br style=“clear: both”>` will do the trick, you often want your clearing element to take up no space on screen, as if it wasn’t even there. The rules for `div.clear` above do a good job of eliminating the space that the div would normally take up, providing an easy way to contain your floats seamless-

ly. In your HTML, just write `<div class=“clear”> </div>`.

Hiding CSS from Old Browsers

To save your sanity – and your client’s cash – it’s best to focus on usability for all browsers, rather than pixel-perfect cross-browser visual consistency. At some point, Ford decided to stop providing support for its Model Ts, and at some point it no longer makes sense for us to spend exorbitant amounts of time optimizing visual appearance in an outdated browser with 1% market share. For most sites, trying to make things look perfect in version 4 browsers simply doesn’t provide good ROI.

Notice that I said “visual appearance” – you of course want to make sure your site is still useable regardless of the user’s browser. However, complicated CSS can actually hamper your site’s usability in older browsers that can’t handle it. Fortunately, you can hide CSS from these browsers to assure that their users will be able to access your site content without complicated CSS markup, turning things into a mess.

The most common and simplest way

The screenshot shows a Netscape browser window displaying the W3C Markup Validation Service. The browser's title bar reads "The W3C Markup Validation Service - Netscape". The address bar contains "http://validator.w3.org/". The main content area features the W3C logo and the heading "MarkUp Validation Service". Below this, a welcome message states: "Welcome to the W3C MarkUp Validation Service; a free service that checks documents like HTML and XHTML for conformance to W3C Recommendations and other standards." The "Validate Files" section includes an "Address:" input field with a "Validate URI..." button and a "Local File:" input field with a "Browse..." button and a "Validate File" button. A "Recent Updates" section follows, mentioning an "Extended Interface" and an "Extended File Upload Interface". A sidebar on the right lists various tools and resources, including "Home Page", "Documentation", "Source Code", "What's New", "Accesskeys", "Feedback", "About...", "Favelets", "Site Valet", "WDG Validator", "CSS Validator", "Link Checker", "HTML Tidy", "Tidy Online", "XHTML 1.1", "XHTML 1.0", and "HTML 4.01". The browser's status bar at the bottom shows "Done".



For the greatest hits
of the 70's, 80's and 90's
call your web host's
tech support.

For answers call us at 1-866-EDGEWEB
3 3 4 3 9 3 2

When calling your web host for support you want answers, not an annoying song stuck in your head from spending all day on hold. At EdgeWebHosting.net, we'll answer your call in two rings or less. There's no annoying on-hold music, no recorded messages or confusing menu merry-go-rounds. And when you call, one of our qualified experts will have the answers you're looking for. Not that you'll need to call us often since our self-healing servers virtually eliminate the potential for problems and automatically resolve most CF, IIS and ASP problems in 60 seconds or less with no human interaction. Plus, our multi-user support system allows you to track support requests for each of your engineers individually, lookup server availability, receive a copy of all errors on your site in real time, and even monitor intrusion attempts on your site in real time. **For a new kind of easy listening, talk to EdgeWebHosting.net**

By the Numbers:

- 2 Rings or less, live support
- 100% Guarantee
- 99.998% Uptime
- 150 MBPS Fiber Connectivity
- 24 x 7 Emergency support
- 24 Hour free backup



EDGE
WEB HOSTING

What are you WAITING for?

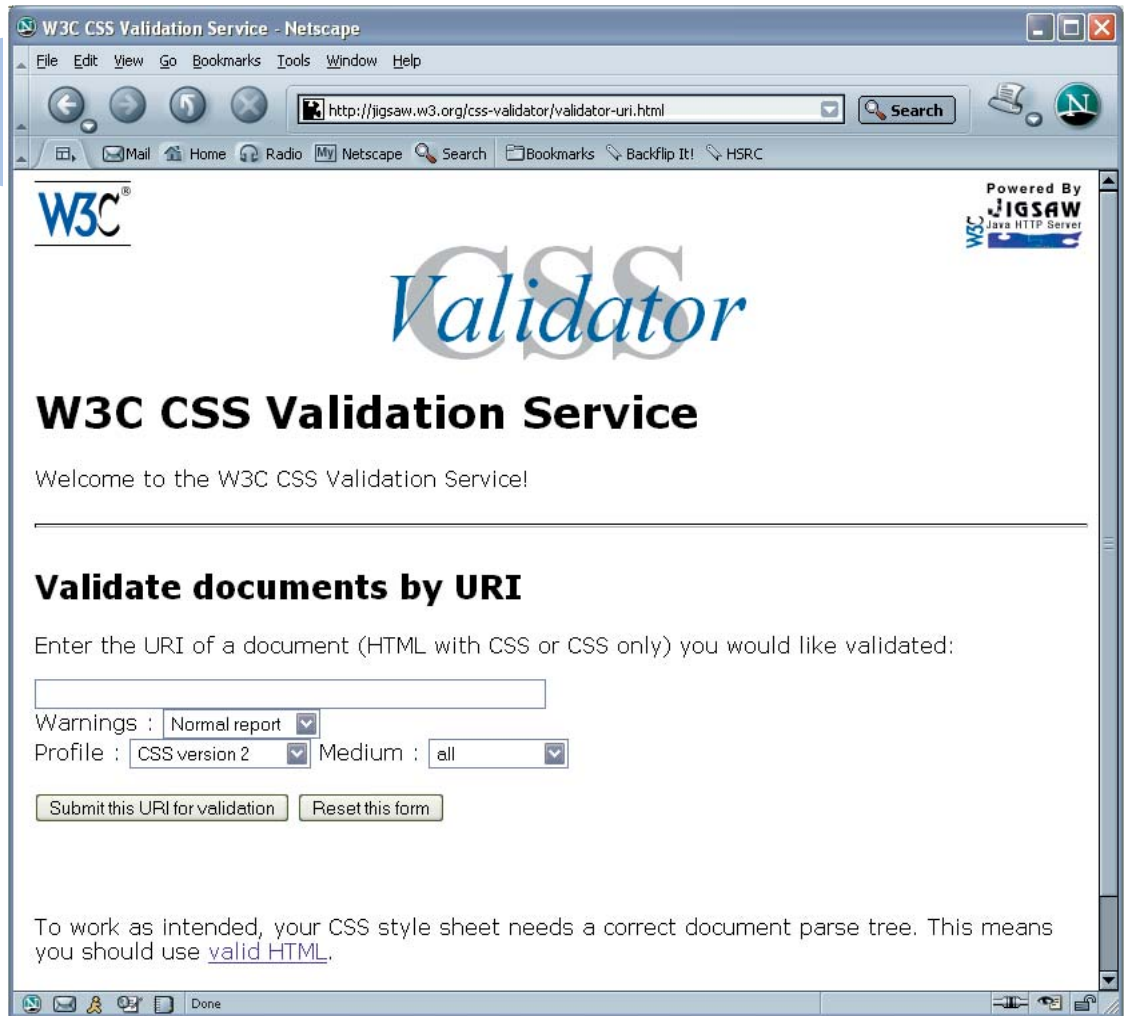
www.edgewebhosting.net

Shared Hosting ¥ Managed Dedicated Servers ¥ Semi-Private Servers
ColdFusion ¥ SQL Server ¥ .NET ¥ Self-Healing Servers ¥ Value Priced

Win
a year
of free
hosting*



*On Shared Hosting or the equivalent value
See <http://edgewebhosting.net/cfdj> for details



to do this is by including your external style sheet using `@import` instead of the link tag. From the Link External Style Sheet dialog box, browse to your CSS file, and select Add As Import (see Image IV). Dreamweaver will insert the following code into the head of your document:

```
<style type="text/css">
<!--
@import url("styles.css");
-->
</style>
```

Netscape Navigator (NN) 4.x will not be able to see style sheets called this way and will be presented with plain text pages. However, IE4, another old browser with poor CSS support, *will* see these styles. Get rid of the "url" part if you want to hide it from IE4 as well as NN4.x:

```
<style type="text/css">
<!--
```

```
@import "styles.css";
-->
</style>
```

If you want to give these old browsers at least limited styling that they can handle, use the link tag to include a basic style sheet and `@import` to include a more advanced style sheet for modern browsers:

```
<link href="styles-basic.css"
rel="stylesheet" type="text/css">
<style type="text/css">
<!--
@import "styles-advanced.css";
-->
</style>
```

The version 4 browsers will ignore the imported style sheet and render the page using the values in the linked one, allowing you to more fully utilize the capabilities of CSS for those browsers that can handle it. If you do use this method,

though, remember that modern browsers will read both style sheets, so if you create specific rules in the basic style sheet that you don't want applied in the advanced one, make sure you specifically set them back to the correct values in the imported sheet.

Using Print Style Sheets

In addition to separate style sheets for older and newer browsers, CSS allows you to set up a style sheet to format the printed version of your Web page. Before CSS, providing a text-only or printer-friendly version of your page often meant creating a separate copy of the content designed for a printer, meaning more work for you and your server. With CSS, you can use a single HTML file and specify how the printed page should appear, including which items on your page should or should not be printed, simply by linking to another style sheet with a media type of "print":



```
<link rel="stylesheet" type="text/css"
href="sheet.css" media="print">
```

Keep in mind that if you have not specified a media type for your other style sheets, they will default to `media="all"` and apply to both screen and print. This means that you will need to place your print style sheet after the other style sheets in the head of your document to make sure its rules take precedence in the cascade on the printed page. To avoid this issue, you can set your other style sheets to `media="screen"`, but having the overlap is actually often desirable – it keeps you from having to include rules you want both on screen and in print in two separate places, so that all your print style sheet has to contain is rules that are specifically changed for print.

Note that NN4.x only recognizes style sheets with a media value of `"screen"`, or no media value at all, so your print style sheet is not going to work in that browser.

Also note that there was a bug in the original release of Dreamweaver MX 2004 that would render the print styles in the design view if you used `@import` with the media specification in the style tag, rather than in the `@import` line, to include the print style sheet. This bug was fixed in the 7.0.1 patch released on March 11, 2004, so be sure to download it from Macromedia's site.

Some things to include in your print style sheet:

- Suppress printing of elements that are only needed on screen, such as navigational elements, by using `"display: none"`.
- Set colors to black and white or other colors that are easier to read on the printed page. Keep in mind that most browsers do not print background colors by default, so if you have white text on a dark background, you may want to change the color of your text to

black in your print style sheet.

- You can further improve readability on the printed page by changing to a serif font and increasing line-height.
- Remember that not everyone will be printing to a color printer, so make sure any information you conveyed in color (e.g., "completed items are marked in red") is conveyed in an additional manner, such as a font style change or with the addition of a border.
- Remove fixed widths so that text can flow to accommodate whatever size paper the user is printing to. Convert other units to measurements that make more sense for print (such as changing pixels to ems or inches).
- Although you may be tempted to change your font sizes to points, leaving them as ems improves your page's accessibility and works just as well in print.
- You may want to remove floats from items and let them flow normally down the page for more reliable printing cross-browser. Older Gecko-based browsers had a bug that cut off floats if they could not fit on the current page, instead of letting them flow to subsequent pages. Although this bug has been fixed, visitors using NN6.x, for example, will still have a problem.

When redefining rules in your print style sheet, make sure you use the exact same name that was used in your general style sheet. If you refer to `"div#navbar"` in your general style sheet, don't refer to it as just `"#navbar"` in the print style sheet, even if those selectors mean the same thing. Sometimes things will fail to be successfully overridden without this naming consistency.

Validating Your Documents

As you build your pages, remember to validate both the (X)HTML and CSS to make sure you are writing things correctly. A validator is a utility that checks

whether a document's syntax conforms to the official definition of its doctype. Although validation is not an end to itself, it is a useful means of ensuring your pages are free from simple errors such as typos or disallowed values, making it a valuable first step in figuring out why things aren't working as you intended. Valid code is more likely to render properly cross-browser and is more forward compatible.

The W3C, which created the (X)HTML and CSS specifications, offers online validators at <http://validator.w3.org/> for (X)HTML (see Image V) and <http://jigsaw.w3.org/css-validator/> for CSS (see Image VI).

You can also validate your page within Dreamweaver by going to File > Check Page > Validate Markup (or Validate as XML if it's an XHTML page). The Validation tab of the Results panel either displays a "No errors or warnings" message or lists the syntax errors it found (see Image VII). Double click on any of these errors to highlight it in the code.

Moving Forward

Even with valid code, you'll still run into inconsistencies in various browsers due to the host of bugs that still exist in their rendering engines. However, following the guidelines above will help you avoid many of the cross-browser headaches Web development inevitably brings up, letting you more easily transition to using CSS throughout your sites as the visual formatting method. ☁

Zoe Gillenwater, a Web designer at the University of North Carolina at Chapel Hill, has a passion for standards-based development. She also keeps busy with graphic design and multimedia projects. Zoe is an active participant in the css-discuss community and is one of those who believes CSS-based layout is ready for prime time.
zoe@pixelsurge.com

XML for Web Designers

Leveraging Macromedia support

by kevin ruse

no doubt, you have heard about XML. XML is everywhere. For Web designers, that can add to the confusion. If something is everywhere, it's nowhere. If only you heard "XML is the new HTML," then maybe you could wrap your mind around it as a markup language. But chances are, you've heard much more than that about XML.

You may have heard one or more of the following:

- XHTML is based on XML.
- XML is a way of storing information.
- XML is part of Web services.
- XML is an object in Flash.
- XML is part of Flash remoting.
- Dreamweaver supports XML.

XML is all of this and more. Thus the confusion for the average Web designer. What is XML? What does it mean to me? Where does it fit in my workflow? How can I leverage Macromedia's support of XML? The goal of this article is to answer these fundamental questions. Even if you never create an XML file, chances are you will be working with an XML file sometime in the very near future.

What Is XML?

Very simply, XML is the eXtensible Markup Language, a World Wide Web Consortium Recommendation as of February 1998. The first important aspect of your understanding of XML is recognizing that it is a meta-language. A meta-language is simply a master language used to create your own unique language, thus there are no predefined tags. For example the meta-language SGML was used by Tim Berners-Lee to create the Hypertext Markup Language (HTML). XML has already been used to create languages such as the Wireless Markup Language (WML) and the Voice Markup

Language (VML). As the author of an XML file you create the tag names. When you receive or author an XML file with custom tag names, you are working with, or have created, an XML application. This fact alone is a significant one. It makes XML fundamentally different from HTML. Code I is an HTML snippet and Code II is a complete XML file.

The HTML file contains information about how the data (content) should be displayed. For example, a table should be rendered with a header row containing the words "Recipe," "Ingredients," and "Amount," followed by a row with corresponding columns that read: "Meatloaf," "Ground Beef," and "1 Pound." There is no information in the markup that explains what the data is. The XML file contains no presentation or display information, but only information about what the data in the tags mean. Thus the nature of the XML tag is to be both human-readable and ultimately machine-readable. XML is meant to be a self-describing document.

The XML file contains no presentation information, so we can reuse it for many purposes. This XML file can be made for use in a Web page, a PDA, a cellphone, a database, a proprietary application, etc.

Currently XML is being used to store the content for Web sites and will therefore land in the hands of the Web designer at some point. Think of it as the new ASCII text file for the Web and more. Web designers may even be asked to translate the current content of their Web sites to XML, so that the data may be repurposed.

Dreamweaver and XML Editing

Dreamweaver MX 2004 can be used to author XML files. The procedure is very straightforward. From the File menu, choose New. Select the "General" tab at

the top of the "New Document" dialog box. Choose "Basic Page" from the Category section on the left and "XML" from the "Basic Page" category on the right, then click the "Create" button.

Dreamweaver creates a blank page in Code View that begins with what is known as the XML declaration:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
```

This line of code is found at the top of the XML document. It is known as a processing instruction, and it simply announces the page as being an XML file written in version 1.0 of the language as defined by the W3C. Because XML is meant to support international coding practices it is written in Unicode as opposed to ASCII code. The default encoding type used in Dreamweaver is ISO-8859-1, which represents a character set similar to ASCII known as the "Latin Alphabet No. 1." However, the Latin Alphabet No. 1 (commonly known as ISO Latin 1) characters also contain characters and letters commonly used in writing the languages of Western Europe and other cultures. ISO-8859 is a family of character sets that extend ASCII, such as ISO-8859-2, which represents the characters of central/eastern Europe. The discussion of character encoding is beyond the scope of this article – the main point is that should you find the encoding type of your XML application unfit for a particular purpose, you may change it in Dreamweaver. The most commonly deployed encoding type is UTF-8, which is a subset of ISO-10646. The ISO-10646 is an international standard superset of widely used national and international characters. You may want to adjust your preferences in Dreamweaver so that the XML declaration writes the encoding

type known as Unicode-8. To do this, choose Edit # Preferences from the main menu. Select the "New Document" category on the right. In the default encoding field (unless the default encoding has been changed, it should read the default encoding type: Western European.) select "Unicode(UTF-8)" and click "OK." All newly created XML files will now begin with the following XML declaration:

```
<?xml version="1.0" encoding="utf-8" ?>
```

Well-Formed XML

XML files must follow certain rules in order to be considered well formed, and all XML files must be well formed. These rules ensure that the XML file is structured. Here again, we see a small but very significant difference between XML and HTML. For example, browsers that parse HTML can understand the following:

```
<UL>
  <LI>Ham
  <LI>Eggs
  <LI>Milk</li>
</OL>
```

as well as:

```
<H1>My heading goes here</H2>
```

In the above code snippets we find a tag closed with an , an tag in uppercase closed with an in lowercase, and an <H1> closed with an <H2>. XML requires far more structure, thus, the snippets above would not be considered well-formed.

All XML files must have a root element. This is the element that contains all the other elements. For example, the root element of an HTML 1.0 Web page is <HTML> because that is the tag that opens and closes the document. Other rules include the following: all opening tags must have closing tags and they must match precisely and that includes case-sensitivity. Thus an opening <Recipe> tag must be closed and cannot be closed with </recipe> because XML is case sensitive (our opening tag included a capital R in recipe and the closing tag is all lowercase). The rules for writing well-formed XML include:

1. All XML Files must contain a root element.

2. All start tags must have end tags.
3. All start tags and end tags must match.
4. When implementing rule two, remember XML is case-sensitive.
5. All tags must be properly nested.
6. All attributes' values must be in quotes.

Dreamweaver will check your file for well-formedness and will indicate your errors using the "Results" panel. Select the "Validation" tab at the top of the "Results" panel. Press the Validate button (the green, right-facing triangle on the left side of the panel) and choose "Validate Current Document." If your XML file is well-formed, the status bar in the "Results" panel will read: "Complete." If your XML breaks a well-formedness rule, the Results panel will display the line with the error followed by a description of the error (see Image I).

Valid XML

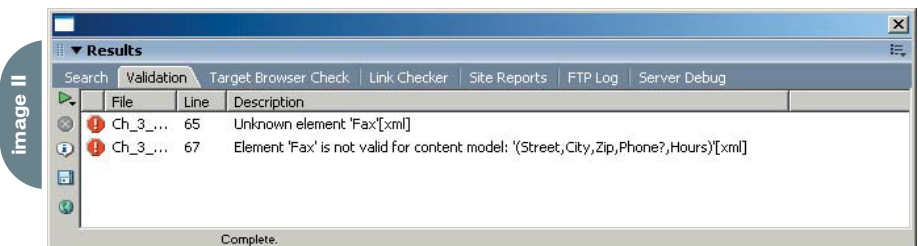
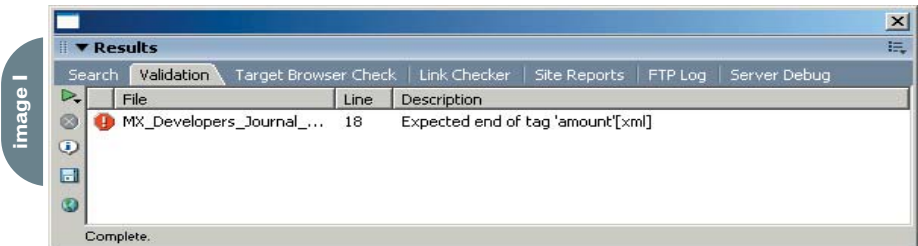
The minimum requirement when working with XML files is that they be well formed. This enforcement is what makes the repurposing of XML files possible because all systems can expect to receive the incoming XML file in a structured, predictable format.

In addition to structure, some systems will require that specific tags be used in a specific order and that the tags contain specific information. Remember that XML is a meta-language, which programmers can use to create their own markup languages. One such language derived from XML is the Wireless Markup Language used in PDAs and cellphone browsers. This is another XML language that

Dreamweaver MX 2004 supports and is capable of writing. Being an XML application, the Wireless Markup Language has specific predefined tags that must be used according to a structure defined by the authors of the language. This structure is enforced with another digital file that can be written in several languages, although the most common are the DTD and the schema. Both of these documents work the same way. They declare the elements (or tag names), how often they can be used and in what order, as well as where in the document they are to be used. DTD's and schemas also declare the other components of an XML file such as attributes, processing instructions and entities. If your XML file does not break any of the rules set forth in the DTD or schema, your file is considered valid. To use a DTD you attach the file to your XML file with the following line of code:

```
<!DOCTYPE Recipe SYSTEM "recipe.dtd">
```

where !DOCTYPE indicates the use of a DTD with this XML file and "Recipe" indicates the root element of the file. The keyword "System" indicates that the DTD file is proprietary in use and can be found in the local network as opposed to "Public," which would indicate that the DTD is for public use and can be found on the Internet. Our recipe.dtd might be an XML application unique to our company and thus a SYSTEM DTD, whereas the Wireless Markup Language is a widely used XML application and the DTD is public and can be found on the Internet.



If you were using Dreamweaver to write an XML application, you could point to the DTD or schema file (by way of the aforementioned line of code) and have Dreamweaver check to see if you are writing the code correctly. Dreamweaver uses the “Results” panel to indicate validity errors (see Image II).

Dreamweaver MX 2004 supports the opening and viewing of XML files as well as the creation of XML files. In addition it provides the means to check our XML files to ensure they are both well-formed and valid.

Using XML in Dreamweaver

Dreamweaver MX 2004 provides several uses for XML. You can import XML into predefined templates. You can export the data from templates to XML and you can access XML data through ColdFusion. XML is also widely used in Flash MX 2004.

XML and Dreamweaver Templates

Macromedia lets you import XML files into Dreamweaver MX templates (.dwt files). In addition, you can also export data from a Dreamweaver MX template to the XML format. The process is very straightforward once you understand the foundations that must be in place.

The first step is to indicate the name and location of the .dwt file that you wish to import your XML into. This is done through the “template” attribute, which points to the .dwt file. The template attribute is placed in the root element of your XML file as follows:

```
<recipe template="Example1.dwt">
```

To import an XML file into your Dreamweaver template, you must name your template's editable regions with names that match the tags in the XML

file. Thus, the recipe example would require creating editable regions with the names: recipe, name, ingredient, and amount. There is one problem with the structure of this XML file – Dreamweaver does not understand XPath, which is a language that identifies locations of elements within an XML file. Thus, Dreamweaver will understand the location of the root element and the children of the root element only. Dreamweaver cannot access nested tags such as the following snippet:

```
<ingredients>
  <ingredient>Ground Beef
    <amount measurement =
      "pounds">1</amount>
</ingredient>
```

Dreamweaver cannot locate the nested <amount> tag or the nested <ingredient> tag because they are not children of the root element. Therefore to import XML into Dreamweaver your file structure must match Code III.

In most cases your XML files will not reflect this structure and you will need to transform the XML to match. XSLT is a stylesheet language that can transform XML files into any other type of file and can be used to transform the original XML file in Code II into the XML file in Code III.

If your Web site is complete and uses templates you can also export a templates data to an XML file. Dreamweaver will structure the XML file as in Code III.

Using XML in ColdFusion

Dreamweaver MX 2004 can create dynamic sites using server-side technologies such as ASP, JSP, and ColdFusion to connect to data sources such as Microsoft Access, Oracle, or any ODBC-compliant data. ColdFusion can also access data in an XML file, including the elements, attributes, and the data they contain. You must work in Code View in order to write the ColdFusion Markup that loads, reads, and parses the XML file. Following that code you simply output the XML to the browser using the ColdFusion <cfoutput> tag.

Loading, Reading, and Parsing XML with ColdFusion

The code to load an XML file (including the first line comment):

code I

```
<table>
  <tr>
    <th>Recipe</th>
    <th>Ingredients</th>
    <th>Amount</th>
  </tr>
  <tr>
    <td>Meatloaf</td>
    <td>Ground Beef</td>
    <td>1 Pound</td>
```

code II

```
<?xml version="1.0">
<recipe>
  <name>Meatloaf </name>
  <ingredients>
    <ingredient>Ground Beef
      <amount measurement = "pounds">1</amount>
    </ingredient>
    <ingredient>Onion
      <amount measurement = "quantity">2</amount>
    </ingredient>
  </ingredients>
</recipe>
```

code III

```
<?xml version="1.0">
<recipe>
  <name>Meatloaf </name>
  <ingredient1>Ground Beef</ingredient1>
  <amount1 measurement = "pounds">1</amount1>
  <ingredient2>Onion</ingredient2>
  <amount2 measurement = "quantity">2</amount2>
</recipe>
```

```
<!-- Load the XML File -->
<CFSET MyXmlFile =
ExpandPath("FileName.xml")>
```

Simply use the <CFSET> tag to set a variable named MyXmlFile which is set using the ExpandPath() method to return the full path to the XML file supplied as the argument ("FileName.xml").

The code to read an XML file (including the first line comment):

```
<!-- Read the XML File -->
<CFFILE ACTION="READ"
FILE="#MyXmlFile#"
VARIABLE="MyXmlCode">
```

The <CFFILE> tag manages interactions with the server and includes the attributes: ACTION set to read a text file on the server; FILE set to the name of the file to read, which in this case is the file indicated in the variable MyXmlFile referenced as a dynamic value by the # symbols surrounding the variable name; and VARIABLE set to the variable name of the XML file.

The code to parse an XML file (including the first line comment):

```
<!-- Parse the XML File -->
<CFSET MyXml = XmlParse(MyXmlCode)>
```

Again, a variable is set, this time using the XmlParse() method to parse the XML file referenced as MyXmlCode from the previous <CFFILE> tag.

Displaying XML in the Browser with ColdFusion

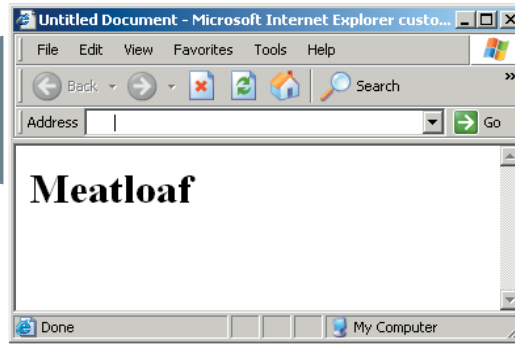
Using the XML file in Code II, we can display the XML in the Web browser as follows:

```
<cfoutput>
  <H1>
    #MyXml.recipe.name.XmlText#
  </H1>
</cfoutput>
```

This code would result in the output shown in Image III.

The ColdFusion Markup Language includes numerous tags and methods for accessing and manipulating XML data for processing and display in the Web browser.

Image III



Macromedia and XML

As you can see, Dreamweaver, while not a full-fledged XML editor, can be used to create, edit, and validate your XML files. After you have created your XML applications you can then access them through ColdFusion as well as Macromedia Flash MX 2004, which includes many new features for manipulating XML including the XMLConnector for accessing XML data from the server. ☁

Kevin Ruse is a technical trainer for Fortune 500 companies throughout the United States. He is the author of XML for Web Designers Using Macromedia Studio MX 2004, published by Charles River Media, Inc. kevin@kevinruse.com

Satisfied with your current Hosting provider?
The grass really is greener on the ServerSide.



ColdFusion Hosting is our specialty. Find out why ColdFusion Developers nationwide choose ServerSide for high quality, high availability web hosting and support.

The grass is greener at www.serverside.net

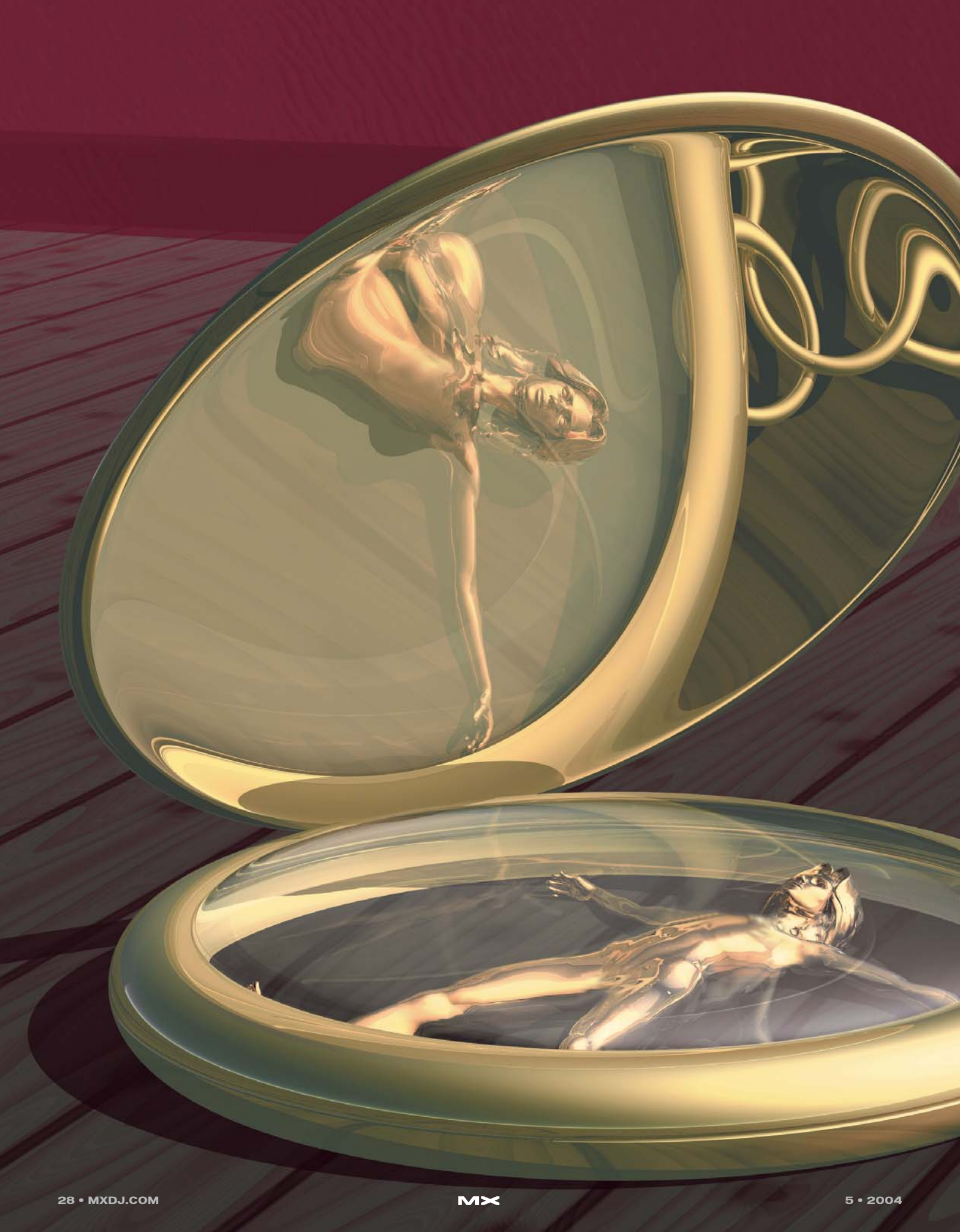
macromedia
COLDFUSION MX

macromedia
 ALLIANCE PARTNER

Mention code #MXD04 : and we'll waive the set-up fee


serversideTM
 Dynamic Web Development. Superior Managed Hosting.

(888) 682.2544
hosting@serverside.net
www.serverside.net



extending flash

Part 1 of this article (Vol. 1, issue 3) covered a general introduction to the Extensibility layer and discussed the fundamental DOM and the relationship between different parts of a flash document and their associated objects. It introduced the history panel and showed you how to build your own flash panels. In this installment we take an in-depth look at XMLUI and flash dialog boxes.

by guy watson

introducing jsapi, part 2



Dialog Boxes - XUL

Some of the Timeline Effects that come pre-installed with Flash MX 2004 allow the user to modify numerous settings to change the outcome of the effect. This is achieved using a dialog box that appears when the Timeline Effect is selected. The dialog box contains a Flash movie control that allows the user to modify the settings of the effect.

The fine engineers at Macromedia have included the ability to provide cross-platform dialog boxes in your Flash extensions. This comes in the form of an XML2UI Engine, which parses and displays dialog boxes, defined using an XML-formatted language called XML2UI. XML2UI is a subset of XUL (www.mozilla.org/projects/xul/xul.html). This article assumes prior knowledge or experience with XML.

Displaying a Dialog Box

Depending on what type of Flash extension you're creating, the XML2UI document is defined in different ways. I discuss the use of XML2UI making the assumption that you are implementing dialog boxes into your commands.

When creating a Flash command that requires some kind of user interaction to configure settings, you define your GUI in a separate XML document with an .xml file extension. This file contains the dialog box definition. XML2UI documents are parsed and displayed by Flash MX 2004, so we have to be able to tell the interpreter to display a dialog box as needed.

In JSFL scripts a method is available that will read the source of a dialog box and then display it; this is called "xmlPanel" and is a method of the document object. As document objects repre-

sent Flash documents open in the authoring environment, it's possible to open an XML2UI dialog box only when one or more documents are open in the Flash Authoring Environment. Below is an example of the "xmlPanel" method:

```
flash.getDocumentDOM().xmlPanel("file://C:/myGUI.xml");
```

When this code is executed, the interpreter reads the dialog box definition from the document specified and renders the controls defined in our XML2UI document. The standard operating system interface controls are used, which means that when your XML2UI document is rendered on a Macintosh, the interface controls will look different from when it's rendered on a Windows machine.

Notice the "getDocumentDOM" method of the "flash" object. This method returns the document object for the currently active Flash document.

When the "xmlPanel" method is called, your JSFL script will pause execution until the dialog box is closed.

Dialog Box Definition

You define the interface of your dialog box by writing a structured XML document that includes special XML nodes that Flash will interpret and display. Each XML node represents a particular part of your interface and has attributes that allow you to customize various settings related to that particular element. You can include various nodes in your XML2UI document that represent common interface controls such as textboxes, radio buttons, and checkboxes. When Flash parses your XML2UI document, all the nodes that it understands are interpreted into a visual interface. If you include nodes that Flash doesn't understand or support in your XML2UI document, Flash will just ignore them, as long as these nodes haven't been misspelled. All node names and attribute names must be lowercase. All XML2UI documents begin and end with a "dialog" node:

```
<dialog>...</dialog>
```

Inside the "dialog" node you define the layout of your dialog box and the various interface controls you want to display within it.

The "dialog" node has various attributes that you can specify. The "buttons" attribute allows you to specify a combination of system buttons to display in your dialog box. You can include three system buttons: accept, cancel, and help. Define the system buttons you want using a comma-separated list; each item in the list is the name of a system button.

```
<dialog buttons="accept,help">...</dialog>
```

This markup will display an empty dialog box with no title that contains two system buttons, "OK" and "Help". Flash lays out system buttons automatically. On both Windows and Macintosh, the buttons are laid out on the bottom row of the dialog box in the standard order for that platform.

If you exclude the three system buttons from your dialog box by not specifying a "buttons" attribute, then it may be troublesome for the user to exit your dialog box; the Escape key will close the dialog box just as if the user had pressed the Cancel button.

To specify a title for your dialog box, add a title attribute to your "dialog" node, the value of which will be the text that Flash uses as the title for your dialog box.

Here is the markup for a dialog box with no title attribute specified (see Image I):

```
<dialog buttons="accept">
<label value="Enter your name" />
<textbox />
</dialog>
```

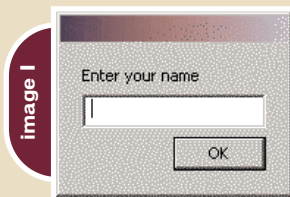
Here is the markup for a dialog box with a title attribute (see Image II):

```
<dialog title="Enter your name" buttons="accept">
<textbox />
</dialog>
```

If your title is long make sure your dialog box is wide enough to display it, otherwise the end of your title will be replaced with "...".

Interface Controls

You can include various nodes in your XML2UI document that represent com-



mon interface controls. The interface controls that Flash will recognize and display are:

- Listbox
- Radio button
- Checkbox
- Textbox
- Color picker
- Dropdown list
- Slider bar
- Label
- Flash movie

Each tag that represents one of those interface controls may require child nodes and/or attributes to define the various characteristics of the element.

Textbox

The “textbox” node represents an interface control that displays a textbox into which a user can type. Various attributes allow you to define the settings of this particular element. For example, you can specify the maximum number of characters that the user can enter into a textbox. You can also specify whether the textbox can contain more than one line of text, and you can define the default text that is displayed in the textbox:

```
<textbox maxlength="50"
multiline="true" value="Your Address"
/>
```

This markup will display a multiline textbox into which the user cannot type more than 50 characters; the default text displayed in it will be “Your Address”.

Label

The “label” node should be used regularly in your dialog boxes to clarify the purpose of a particular control. The “label” node has two important attributes; the first is the “value” attribute, which defines the text that will be shown for the label:

```
<label value="The label text" />
```

The “control” attribute allows you to associate a label with a particular interface control:

```
<label value="Enter your name" control="firstName" />
<textbox id="firstName" />
```

In the above markup, I have associated a label with a textbox by giving the “textbox” node an “id” attribute, and by setting this “id” as the value of the “control” attribute for my label.

Color Picker

It’s possible to show a chip of color in your interface that when clicked will expand to display a grid of colors from which the user can select a desired color. Upon selecting the desired color, the color of the chip is updated.

To use a color picker in your dialog box, specify a “colorchip” node. This node has one important attribute, which you can use to define the default hexadecimal color displayed in the chip. The attribute is named “color” and is used as follows:

```
<dialog title="Color Picker" buttons="accept">
<colorchip color="#FF0000" />
</dialog>
```

The above markup will display a color picker, set to red (#FF0000) until the user changes it (see Image III).

Button

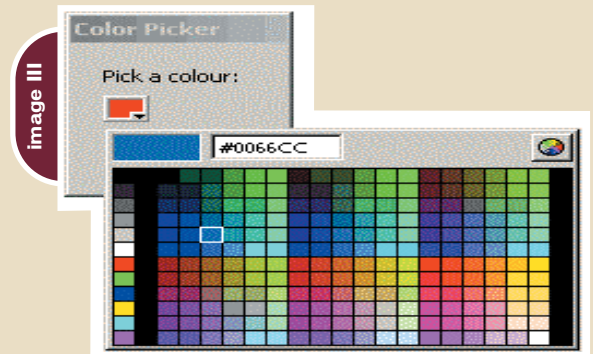
In addition to the ability to include the three system buttons in your dialog boxes (accept, cancel, help), it’s also possible to define your own buttons. To define a button in your dialog box definition use the “button” node. The label that is displayed on your button is defined using the “label” attribute of the “button” node:

```
<dialog title="Click the button">
<button label="Click me" />
</dialog>
```

Checkbox

Checkboxes are interface controls that have two possible states, selected and unselected (on/off, 0/1, true/false, yes/no). Checkboxes are generally used to allow the user to choose one or more options from a list of available options. Individual checkboxes can have labels; to specify a label for a checkbox you define the “label” attribute; the label is displayed on the right of the checkbox.

Code I is a sample usage of checkbox-



es; this markup is rendered as shown in Image IV.

Slider Bar

Slider bars are used to present an option that requires the user to choose an integer value from a specified range, such as a percentage from 0–100. To implement a slider bar into your dialog box, use the “popupslider” node. This node has two required attributes: “minvalue”, which is the minimum possible value in the range, and “maxvalue”, which is the maximum possible value in the range:

```
<dialog title="Slider bar example"
buttons="accept">
<popupslider minvalue="0" maxvalue="100" />
</dialog>
```

Radio Buttons

Radio buttons are similar to checkboxes when used individually, as they have two states, selected and unselected (on/off, 1/0, true/false, yes/no). However, in Flash, radio buttons cannot function individually; they have to be a part of a radio button group. The difference between a group of checkboxes and a group of radio buttons is that only one radio button in a group can be selected; selecting another radio button in the group deselects the currently selected option in the group. In a group of checkboxes it is possible to select all of the checkboxes.

To implement a radio button in your interface, you must first define a “radiogroup” node, which will contain the individual radio button nodes that are a part of the group.

```
<radiogroup>...</radiogroup>
```



image IV

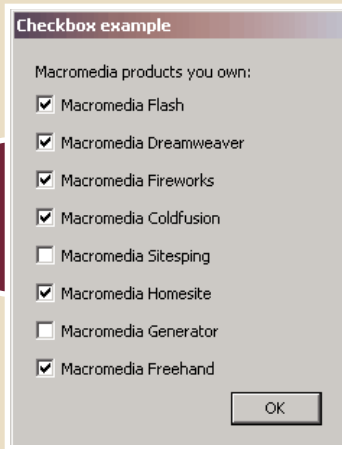
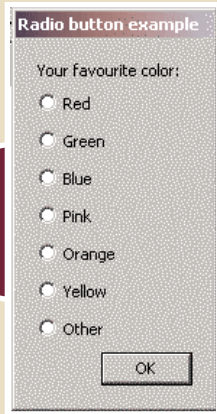


image V



Then you can define one or more individual radio buttons group members by adding a "radio" node as a child of the "radiogroup" node.

```
<radiogroup>
<radio />
<radio />
</radiogroup>
```

Radio buttons generally have a label associated with them to display the option they are choosing. To define a label for a radio button you must specify the "label" attribute (see Code II); this markup is rendered as shown in Image V.

Listbox

Listboxes present a selection of options to a user, allowing only one option to be selected. To implement a listbox control into your dialog box you need to define a "listbox node". The "listbox" node contains individual "listitem" nodes, which represent each individual item in the listbox. Each listitem can be assigned a label and a value. The value of the "label" attribute is displayed as the label for the option in the listbox, and the

"value" attribute is used to define the value of the listbox as a group when the item is selected (see Code III); this markup is rendered as shown in Image VI.

We have four options, and rather than the user having to scroll to see the options that aren't in view, we can define how many items to display in the viewable area at any one time using the "rows" attribute of the "listbox" node. To view all the items in the listbox without having to scroll, set the value of the "rows" attribute to be the total number of list items in our listbox plus one (see Code IV). Now the dialog box looks like Image VII.

Dropdown List

Dropdown lists show multiple options from which a user can choose one. Dropdown lists and listboxes are similar in function, but a listbox generally takes up more space and a listbox's options are generally not displayed all at once – thus the user has to scroll through the options using a scrollbar in the control. To implement a dropdown list into your dialog box use the "menulist" node, which is a container for individual "menuitem" nodes (see Code V).

To stick to the XUL guidelines you are supposed to place another container inside the "menulist" node. I assume that the next version of Flash will have more support for XUL elements and thus recommend that you adhere to these standards if you want your extensions to work in the future.

The standard states that you should place all "menuitem" nodes inside a "menupop" node; our markup now looks like Code VI. Whether you include it or not, the outcome is the same (see Image VIII).

By default, the dropdown list is empty until an option is selected, but it's possible to specify which menuitem is selected by including a "selected" attribute for the "menuitem" node that you want to be selected. The value can be "true" or "false", where true is selected and false is not selected. Not including the selected attribute is the same as specifying a "false" value (see Code VII).

It is good practice to include the "selected" attribute for each "menuitem" node in your dropdown list (see Code VIII).

Flash Movie

In a Web browser, it's also possible to build dialog boxes using the numerous form tags and input element tags. In the early days it wasn't possible for the state of a form, or the various input elements, to change based upon user input. Dynamic forms became possible with the introduction of DHTML into Web browsers. With XML2UI, it's also not possible to change a form once it has been rendered. Macromedia decided to allow developers to include Flash movies inside a dialog box, thus creating a host of new possibilities (e.g., loading dynamic data into your dialog box from a remote location).

Use a Flash movie control to create dynamic dialog boxes that can change state after they have been rendered. All of the input controls in XML2UI and many more are available in Flash MX 2004 as Flash components. The Flash movie for your dialog box can contain anything; you may just want to have a pretty animation piece in your dialog box, or you may want a little control over the look and feel of your dialog box. You can also have a Flash movie and any of XML2UI's other input controls in the same dialog box.

If you're having trouble writing an XML2UI document, all you need to know

image VI

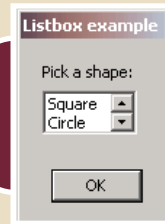


image VII

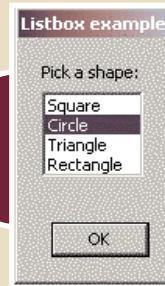
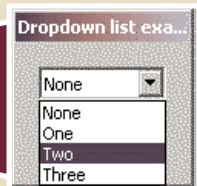


image VIII



One damn good Cold Fusion hosting firm!

Webcore Technologies specializes in ColdFusion, ASP and SQL hosting. We offer these solutions in both dedicated server and shared virtual environments.

Webcore can design and implement clustered or load-balanced solutions, managed firewalls, VPN's, multi-site failover, and more. In addition, we also offer corporate Internet access, web site design, and technology consulting services.

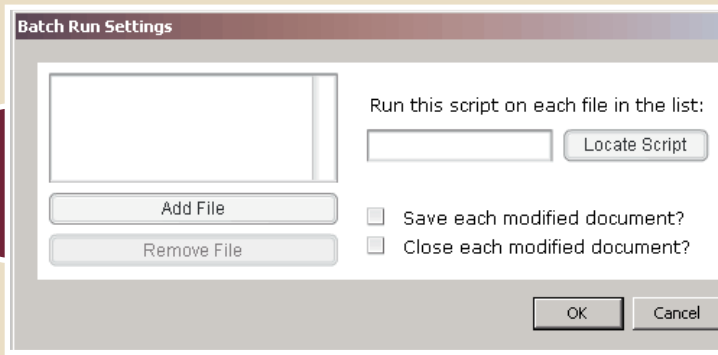
Our in-house tier 1 level data center enables us to provide the most reliable hosting in the industry. Our Microsoft and Cisco certified team ensures that all support issues are resolved promptly and professionally. Unlike other hosting companies that answer with a phone attendant, you will receive a live person when you call Webcore. No phone attendant, no frustrations, just world class customer service and support the first time you call.



Webcore Technologies, Inc.
877.WCT.HOST
www.webcoretech.com



image IX



is the basic XML to display a Flash movie; then you can build all your dialog boxes in Flash. It's possible for Flash movies displayed in a dialog box to execute JSFL code using the MMExecute function.

I made my own Flash command that opens a dialog box containing a Flash movie. The Flash movie allows the user to browse for files and add them to a list. When the dialog box is closed, a JSFL script is executed for each file that was added to the list, thus enabling batch-scripting by Flash developers.

Image IX shows what my dialog box looks like; I built it using Flash MX 2004 and Flash components. When the "Add File" push button is pressed I execute some JSFL using the MMExecute ActionScript function, which opens a "Select File" dialog box. When the user selects a file, the fileURI is returned to ActionScript via the MMExecute function, and then I add it to the list.

To implement a Flash movie in your dialog box, use the "flash" node, which is not a standard element of XUL:

```
<dialog title="Batch Run Settings"
buttons="accept,cancel" >
  <flash width="475" src="Batch
Run.swf" height="150" id="settings"
/>
</dialog>
```

You must include an "src" attribute for all "flash" nodes so that when your dialog box is rendered the interpreter knows where to look for the Flash movie. You must also specify the width and height you want the Flash movie to be displayed at; otherwise it will be displayed at 10 pixels wide by 10 pixels high. The width and height attributes should match those that you would use if you were to display the Flash movie in an HTML page.

Dialog Box Layout

A dialog box will not function correctly if it isn't laid out properly. In XML2UI there are two ways of laying out your dialog boxes: (1) a grid system with rows and columns, or (2) hboxes and vboxes, which automatically lay out their contents horizontally and vertically, respectively (the better option in my opinion).

Grid Layout

To specify a grid to lay out your content, use the "grid" node, which is used to group together a collection of columns and rows. The columns and rows are simply containers and cannot be seen. The columns and/or the rows you specify in your grid are used either as white space or to position interface controls. The "grid" node should contain a "columns" node, a "rows" node, or both:

```
<grid>
  <columns>...</columns>
  <rows>...</rows>
</grid>
```

Inside the "columns" node you define one or more empty "column nodes", and each one adds a new column to your virtual layout grid. Remember that columns run from left to right. It's recommended that you don't place any nodes within your "column" nodes; if you do, then each node contained within a "column" node is placed inside each successive row in the grid. The column with the most child nodes determines the number of rows in each column.

Inside the "rows" node, you define one or more "row" nodes; each one adds a new row to your virtual layout grid. Remember that rows run from top to bottom. Each node contained within a "row" node is placed according to its sequential position; for example, the first child node

of a "row" node will be placed in the first column of the grid, and the second child node in the second column of the grid.

To align the contents of a row, you must specify an "align" attribute; its value can be "start", "center", "end", or "baseline".

Code IX is a sample grid that contains two columns, and each column has two rows. This markup is rendered as shown in Image X.

Note: It is good practice to always define your columns first (with a list of empty columns as its children) and then to define your rows, each row containing the interface controls. Also make sure that you define enough empty columns for the amount of interface controls in each row.

Box Layout

Boxes allow you to divide a dialog box into a series of boxes. Interface controls inside a box arrange themselves horizontally or vertically. By combining a series of boxes you can lay out your dialog box in no time, leaving the hard work to the renderer as it decides where the interface controls will be placed. There are two types of boxes: (1) vertical boxes, which lay out their contents one on top of the other, and (2) horizontal boxes, which lay out their contents one next to the other. To lay out your content vertically, use the "vbox" node, which is simply a container node into which you place the nodes of interface controls you want to be arranged vertically (see Code X). This markup produces the dialog box shown in Image XI.

As you can see, the interface controls are stacked on top of each other, from top to bottom, which looks neat, but isn't perfect. However, arranging the interface controls horizontally looks even worse.

To lay out your content horizontally, use the "hbox" node, which is simply a container node into which you place the nodes of interface controls you want to be arranged horizontally (see Code XI). This markup produces the dialog box shown in Image XII.

You can place multiple "vbox" nodes inside an "hbox" node and vice versa, allowing you to vertically arrange multiple groups of horizontally arranged elements (see Code XII). This produces the dialog box shown in Image XIII, which is perfectly arranged.

Which Settings Did the User Choose?

The purpose of displaying a dialog box is to allow the user to change various settings, which will determine how your Flash command will work. Excellent! But how do we determine what choices the user made in the dialog box? The “xmlPanel” method returns an object containing one or more properties, depending upon how many were defined in the XML2UI document.

You define a property by specifying an “id” attribute for any interface control node; when the dialog box is closed, Flash MX 2004 populates that particular property name with the value that the user chose. All properties of your dialog box and their associated values are grouped together in the object returned from the “xmlPanel” method call. For example, if I specify an “id” attribute as “username” for a textbox node, the object returned from the “xmlPanel” method will contain a property named “username” whose value will be what the user entered into the textbox:

```
<dialog buttons="accept,cancel">
<textbox id="username" />
</dialog>
```

There is one property that will always be defined in the object returned by an “xmlPanel” method call; the property is named “dismiss”, and it can have one of two possible string values. If the dialog box was closed using the OK button, then the value will be “accept”; however, if the dialog box was closed using the Cancel button, then the value will be “cancel”. In addition, if a dialog box is closed using the Cancel button, then only the property “dismiss” will be returned (see Code XIII).

It’s easy to get a return value from an XML2UI dialog box that contains any of the basic controls – the color picker, the textbox, or the checkbox – as there is only one option. However, for controls such as the radio button group, list box, and dropdown list, it’s a different story. There are numerous options.

In a group of radio buttons only one radio button can be selected, and thus there is only one value to be returned. However, each radio button in the group can have a separate value. To specify a

value for a radio button, use the “value” attribute of the radio node; you must also specify an “id” attribute for the “radio-group” that contains the radio buttons, such that when the user makes a choice, the value can be returned to JSFL (see Code XIV). When the dialog box is displayed, the user is presented with a list of options (see Image XIV).

When the dialog box is closed using the OK button, the value that was assigned to the selected radio button is returned as the value for the radio group. If I were to select “Yellow” and then close the dialog box, an object would be returned to JSFL containing two properties, (1) the default “dismiss” property, which will have a value of “accept”, and (2) another property called “color” whose value will be “y”, as that is the value we assigned to the “Yellow” radio button in the XML2UI document for the dialog box.

The listbox and dropdown list work in exactly the same way; you need to specify an “id” attribute for the “container” node that contains the individual items. Then you assign a “value” attribute to each individual item (see Code XV).

In Code XV, the “listbox” node is the container node; thus, we’ve given it an “id” attribute. Its children, the “listitem” nodes, are the individual items, and each one has its own “value” attribute.

As always, there’s an exception to the rule. It’s possible for Flash to pass multiple values back to JSFL in the return object, but it doesn’t work in the same way as the other interface controls, as each of the other interface controls will only return one value. To solve this problem Macromedia has created a new ActionScript object, which is available to Flash movies running inside of dialog boxes. This ActionScript object is conveniently named “XMLUI” and it has four methods:

- XMLUI.accept()
- XMLUI.cancel()
- XMLUI.get()
- XMLUI.set()

The first two are simple; they do the same thing that occurs when a user clicks on the Accept button and the Cancel button. Remember that the Accept button is displayed as “OK”.

If you want to customize the look and feel of the OK and Cancel buttons by

implementing them into your Flash movie, use these methods as opposed to using the default system buttons.

The “XMLUI.get” method will return the value of a “property” node defined in the XML2UI document that contains the Flash Movie:

```
XMLUI.get ("propertyName" );
```

The return value will always be a string.

The “XMLUI.set” method will change the value of a property node defined in the XML2UI document that contains the Flash movie:

```
XMLUI.set ("propertyName", "value" )
```

This method will only accept a string as the value for the “propertyName” argument, and it will only accept a string as the value for the second argument (the “value” argument), thus any ActionScript arrays should be joined using the array.join() method prior to using this method.

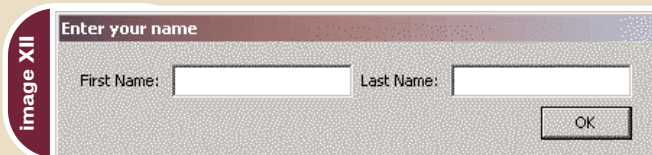
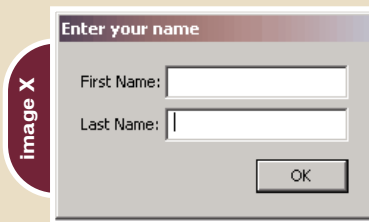




image XIV

I mentioned that the XMLUI.set method could only change the value of a property, which means that the actual property has to be defined in the XML2UI document as well, which makes it impossible to pass any old property back to JSFL from a Flash movie control.

To define a property that you want to be returned back to JSFL from a dialog box, use the XML2UI "property" node.

```
<dialog title="Batch Run Settings"
buttons="accept,cancel" >
<flash width="475" src="Batch
```

```
Run.swf" height="150" id="settings"
/>
<property id="files" />
</dialog>
```

The "property" node has two possible attributes. The first, the "id" attribute, is required and defines the name of the property whose value will be included in the return object. The second, the "value" attribute, allows you to define a default value for the property in case that particular one is not set by the Flash movie control.

Using ActionScript in a Flash movie that is in that same dialog box, I can set the value of the files property using:

```
theFiles=["tester.jpg", "tester2.jpg"];
XMLUI.set("file", theFile.join(", "));
```

Then the JSFL script that opened the dialog box will return an object containing two properties when this dialog box is closed. The first is the default "dismiss" property and the second is the "file" property, which will contain a string, concatenated with a comma (,) so I can then split

it back into an array using JSFL.

Resources

- XULPlanet: www.xulplanet.com
- Flash Extensibility: www.flashextensibility.com
- ExtendFlash: www.flashguru.co.uk/mailman/listinfo/extendflash_flashguru.co.uk

Guy Watson (aka FlashGuru) is a well-recognized figure in the Flash community, supporting the community with tutorials and source files, moderating the large Flash community forums, and running his own Flash resource Web site - FlashGuru's MX 101. Guy was one of the two developers that created the award-winning zoom interface for Relevare and now works for Endemol UK, the creative force behind reality television, producing programs such as "Big Brother" and "The Salon". Guy spends most of his time developing Flash games and applications for high-profile clients such as Channel 5 Television, Ladbrookes, and UK Style. guy@flashguru.co.uk

code I

```
<dialog title="Checkbox example" buttons="accept">
<label value="Macromedia products you own:" />
<checkbox label="Macromedia Flash" />
<checkbox label="Macromedia Dreamweaver" />
<checkbox label="Macromedia Fireworks" />
<checkbox label="Macromedia Coldfusion" />
<checkbox label="Macromedia Sitesping" />
<checkbox label="Macromedia Homesite" />
<checkbox label="Macromedia Generator" />
<checkbox label="Macromedia Freehand" />
</dialog>
```

code II

```
<dialog title="Radio button example" buttons="accept">
<label value="Your favourite color:" />
<radiogroup>
<radio label="Red" />
<radio label="Green" />
<radio label="Blue" />
<radio label="Pink" />
<radio label="Orange" />
<radio label="Yellow" />
<radio label="Other" />
</radiogroup>
</dialog>
```

code III

```
<dialog title="Listbox example" buttons="accept">
<label value="Pick a shape:" />
<listbox>
<listitem label="Square" />
```

```
<listitem label="Circle" />
<listitem label="Triangle" />
<listitem label="Rectangle" />
</listbox>
</dialog>
```

code IV

```
<dialog title="Listbox example" buttons="accept">
<label value="Pick a shape:" />
<listbox rows="5">
<listitem label="Square" />
<listitem label="Circle" />
<listitem label="Triangle" />
<listitem label="Rectangle" />
</listbox>
</dialog>
```

code V

```
<dialog title="Dropdown list example" buttons="accept">
<menulist>
<menuitem label="None" />
<menuitem label="One" />
<menuitem label="Two" />
<menuitem label="Three" />
</menulist>
</dialog>
```

code VI

```
<dialog title="Dropdown list example" buttons="accept">
<menulist>
<menupop>
<menuitem label="None" />
```

code VII

```

<menuItem label="One" />
<menuItem label="Two" />
<menuItem label="Three" />
</menuop>
</menulist>
</dialog>

```

code XI

```

<dialog buttons="accept" title="Enter your name">
<hbox>
<label value="First Name:" control="fName" />
<textbox id="fName" />
<label value="Last Name:" control="sName" />
<textbox id="sName" />
</hbox>
</dialog>

```

code VIII

```

<dialog title="Dropdown list example" buttons="accept">
<menulist>
<menuop>
<menuItem label="None" selected="true" />
<menuItem label="One" selected="false" />
<menuItem label="Two" />
<menuItem label="Three" />
</menuop>
</menulist>
</dialog>

```

code XII

```

<dialog buttons="accept" title="Enter your name">
<vbox>
<hbox>
<label value="First Name:" control="fName" />
<textbox id="fName" />
</hbox>
<hbox>
<label value="Last Name:" control="sName" />
<textbox id="sName" />
</hbox>
</vbox>
</dialog>

```

code IX

```

<dialog title="Dropdown list example" buttons="accept">
<menulist>
<menuop>
<menuItem label="None" selected="true" />
<menuItem label="One" selected="false" />
<menuItem label="Two" selected="false" />
<menuItem label="Three" selected="false" />
</menuop>
</menulist>
</dialog>

```

code XIII

```

settings=flash.getDocumentDOM().xmlPanel('file:///C:/myGU
I.xml');
if(settings.dismiss == "accept")
{
username=settings.username
}
else
{
//cancelled the dialog box, use default settings or do
nothing
}

```

code X

```

<dialog buttons="accept" title="Enter your name">
<grid>
<columns>
<column />
<column />
</columns>
<rows>
<row>
<label value="First Name:" control="fName" />
<textbox id="fName" />
</row>
<row>
<label value="Last Name:" control="sName" /><
<textbox id="sName" />
</row>
</rows>
</grid>
</dialog>

```

code XIV

```

<dialog title="Radio button example" buttons="accept">
<label value="Your favourite color:" />
<radiogroup id="color">
<radio label="Red" value="r" />
<radio label="Green" value="g" />
<radio label="Blue" value="b" />
<radio label="Pink" value="p" />
<radio label="Orange" value="o" />
<radio label="Yellow" value="y" />
<radio label="Other" value="ot" />
</radiogroup>
</dialog>

```

code XV

```

<dialog buttons="accept" title="Enter your name">
<vbox>
<label value="First Name:" control="fName" />
<textbox id="fName" />
<label value="Last Name:" control="sName" />
<textbox id="sName" />
</vbox>
</dialog>

```

```

<dialog title="Listbox example" buttons="accept">
<label value="Pick a shape:" />
<listbox id="shape">
<listitem label="Square" value="square" />
<listitem label="Circle" value="circle" />
<listitem label="Triangle" value="triangle" />
<listitem label="Rectangle" value="rectangle" />
</listbox>
</dialog>

```



image XIV

I mentioned that the XMLUI.set method could only change the value of a property, which means that the actual property has to be defined in the XML2UI document as well, which makes it impossible to pass any old property back to JSFL from a Flash movie control.

To define a property that you want to be returned back to JSFL from a dialog box, use the XML2UI "property" node.

```
<dialog title="Batch Run Settings"
buttons="accept,cancel" >
<flash width="475" src="Batch
```

```
Run.swf" height="150" id="settings"
/>
<property id="files" />
</dialog>
```

The "property" node has two possible attributes. The first, the "id" attribute, is required and defines the name of the property whose value will be included in the return object. The second, the "value" attribute, allows you to define a default value for the property in case that particular one is not set by the Flash movie control.

Using ActionScript in a Flash movie that is in that same dialog box, I can set the value of the files property using:

```
theFiles=["tester.jpg", "tester2.jpg"];
XMLUI.set("file", theFile.join(", "));
```

Then the JSFL script that opened the dialog box will return an object containing two properties when this dialog box is closed. The first is the default "dismiss" property and the second is the "file" property, which will contain a string, concatenated with a comma (,) so I can then split

it back into an array using JSFL.

Resources

- XULPlanet: www.xulplanet.com
- Flash Extensibility: www.flashextensibility.com
- ExtendFlash: www.flashguru.co.uk/mailman/listinfo/extendflash_flashguru.co.uk

Guy Watson (aka FlashGuru) is a well-recognized figure in the Flash community, supporting the community with tutorials and source files, moderating the large Flash community forums, and running his own Flash resource Web site - FlashGuru's MX 101. Guy was one of the two developers that created the award-winning zoom interface for Relevare and now works for Endemol UK, the creative force behind reality television, producing programs such as "Big Brother" and "The Salon". Guy spends most of his time developing Flash games and applications for high-profile clients such as Channel 5 Television, Ladbrookes, and UK Style. guy@flashguru.co.uk

code I

```
<dialog title="Checkbox example" buttons="accept">
<label value="Macromedia products you own:" />
<checkbox label="Macromedia Flash" />
<checkbox label="Macromedia Dreamweaver" />
<checkbox label="Macromedia Fireworks" />
<checkbox label="Macromedia Coldfusion" />
<checkbox label="Macromedia Sitesping" />
<checkbox label="Macromedia Homesite" />
<checkbox label="Macromedia Generator" />
<checkbox label="Macromedia Freehand" />
</dialog>
```

code II

```
<dialog title="Radio button example" buttons="accept">
<label value="Your favourite color:" />
<radiogroup>
<radio label="Red" />
<radio label="Green" />
<radio label="Blue" />
<radio label="Pink" />
<radio label="Orange" />
<radio label="Yellow" />
<radio label="Other" />
</radiogroup>
</dialog>
```

code III

```
<dialog title="Listbox example" buttons="accept">
<label value="Pick a shape:" />
<listbox>
<listitem label="Square" />
```

```
<listitem label="Circle" />
<listitem label="Triangle" />
<listitem label="Rectangle" />
</listbox>
</dialog>
```

code IV

```
<dialog title="Listbox example" buttons="accept">
<label value="Pick a shape:" />
<listbox rows="5">
<listitem label="Square" />
<listitem label="Circle" />
<listitem label="Triangle" />
<listitem label="Rectangle" />
</listbox>
</dialog>
```

code V

```
<dialog title="Dropdown list example" buttons="accept">
<menulist>
<menuitem label="None" />
<menuitem label="One" />
<menuitem label="Two" />
<menuitem label="Three" />
</menulist>
</dialog>
```

code VI

```
<dialog title="Dropdown list example" buttons="accept">
<menulist>
<menupop>
<menuitem label="None" />
```

code VII

```

<menuItem label="One" />
<menuItem label="Two" />
<menuItem label="Three" />
</menupop>
</menulist>
</dialog>

```

code XI

```

<dialog buttons="accept" title="Enter your name">
<hbox>
<label value="First Name:" control="fName" />
<textbox id="fName" />
<label value="Last Name:" control="sName" />
<textbox id="sName" />
</hbox>
</dialog>

```

code VIII

```

<dialog title="Dropdown list example" buttons="accept">
<menulist>
<menupop>
<menuItem label="None" selected="true" />
<menuItem label="One" selected="false" />
<menuItem label="Two" />
<menuItem label="Three" />
</menupop>
</menulist>
</dialog>

```

code XII

```

<dialog buttons="accept" title="Enter your name">
<vbox>
<hbox>
<label value="First Name:" control="fName" />
<textbox id="fName" />
</hbox>
<hbox>
<label value="Last Name:" control="sName" />
<textbox id="sName" />
</hbox>
</vbox>
</dialog>

```

code IX

```

<dialog title="Dropdown list example" buttons="accept">
<menulist>
<menupop>
<menuItem label="None" selected="true" />
<menuItem label="One" selected="false" />
<menuItem label="Two" selected="false" />
<menuItem label="Three" selected="false" />
</menupop>
</menulist>
</dialog>

```

code XIII

```

settings=flash.getDocumentDOM().xmlPanel('file:///C:/myGU
I.xml');
if(settings.dismiss == "accept")
{
username=settings.username
}
else
{
//cancelled the dialog box, use default settings or do
nothing
}

```

code X

```

<dialog buttons="accept" title="Enter your name">
<grid>
<columns>
<column />
<column />
</columns>
<rows>
<row>
<label value="First Name:" control="fName" />
<textbox id="fName" />
</row>
<row>
<label value="Last Name:" control="sName" /><
<textbox id="sName" />
</row>
</rows>
</grid>
</dialog>

```

code XIV

```

<dialog title="Radio button example" buttons="accept">
<label value="Your favourite color:" />
<radiogroup id="color">
<radio label="Red" value="r" />
<radio label="Green" value="g" />
<radio label="Blue" value="b" />
<radio label="Pink" value="p" />
<radio label="Orange" value="o" />
<radio label="Yellow" value="y" />
<radio label="Other" value="ot" />
</radiogroup>
</dialog>

```

code XV

```

<dialog buttons="accept" title="Enter your name">
<vbox>
<label value="First Name:" control="fName" />
<textbox id="fName" />
<label value="Last Name:" control="sName" />
<textbox id="sName" />
</vbox>
</dialog>

```

```

<dialog title="Listbox example" buttons="accept">
<label value="Pick a shape:" />
<listbox id="shape">
<listitem label="Square" value="square" />
<listitem label="Circle" value="circle" />
<listitem label="Triangle" value="triangle" />
<listitem label="Rectangle" value="rectangle" />
</listbox>
</dialog>

```



A New Solution for Flash Remoting

Integrating Flash clients with server-side components

by joe orbman

If you do Flash Remoting in .NET – read on. We at *MXDJ* began to hear about FlashORB and went straight to the source for an inside view of this alternative to Macromedia's Flash Remoting.

Everyone is talking about rich Internet applications. Some are experimenting; others are building real applications. Flash MX downloads are exceeding all expectations, and there is a general consensus that the Internet has evolved enough to take the user experience to the next level. With all the advancements on the client side, however, there has not been enough innovation on the server side. Macromedia has provided the initial set of the server-side technologies with Flash Remoting, but has not updated them since the initial release. The technology is still intrusive, noncohesive, and expensive. Many of these factors have influenced the creation of several alternatives to Flash Remoting. One of them, FlashORB, is reviewed in this article.

FlashORB is essentially a Flash Messaging server, consisting of three major subsystems: Flash Remoting, Web Services Gateway, and XML Socket Server. Available in two editions (Java and .NET), the software provides a new way to integrate Flash clients with the server-side components (Java/.NET objects, EJBs, Web services, business applications). Architecturally the product is positioned between the client Flash UI and the server-side application; as a result, FlashORB addresses the needs of both UI and server component developers. For example, facilities within FlashORB like Call Tracing in the Management Console and the ActionScript code generator aid the UI team, while custom serializers, activation modes, and object factories help server-side developers. FlashORB nicely separates the responsibilities of the UI and the server-side development teams.

A Few Words About Flash Remoting

If you are new to the concept of Flash Remoting, it is worthwhile to get a brief overview of the technology before delving into the core of FlashORB. The primary purpose of Flash Remoting is to enable Flash clients to perform invocations on server-side services (Java/.NET objects, Web services, etc.). The technology consists of two primary parts: Flash Remoting Components and Flash Remoting Server. The Remoting components are a set of ActionScript programs that enable connectivity and messaging between Flash clients and the server side. The Flash Remoting Server is responsible for receiving client requests, dispatching invocations on the specified components, and serializing responses in the appropriate format. The format utilized in Flash Remoting is called AMF (Action Message Format); it is a binary format but is streamed over HTTP/HTTPS to ease firewall traversal. Code I and Code II demonstrate the server-side class and the client-side ActionScript invoking a method of the class.

The operation flow is very simple.

1. Flash client creates a connection to a server running at "http://remoting server url" (line 3).
2. It obtains a reference to a service identified by the full type (class) name. (line 4).
3. The client invokes a method from the server class, as if it is present locally (line 5).
4. Flash Remoting Server receives client request and performs method invocation on the designated type(class). The result from the method invocation is sent to the client.
5. The response received from the server is dispatched into a function named as `methodName_Result`, where `methodName` is the invoked method (line 7).

Better Flash Remoting with .NET

Now let's take a look at FlashORB as a remoting solution focusing on the main features differentiating it from the competing alternatives available in the marketplace. Each feature will be demonstrated with an example to clarify its use with actual code. For the simplicity and clarity, all of the examples use FlashORB.NET – the edition of the product for the Microsoft.NET environment; all the features discussed are also available in the Java edition.

Type Adaptation

One of the major product differentiators is the type adaptation system. Driven by the design principle of nonintrusive integration, the type adaptation system allows automatic plug-and-play of FlashORB into user applications. To understand how type adaptation works consider the example in Code III and Code IV.

The Flash side sends an untyped object to the server (line 10). When FlashORB receives such an object it tries to convert it into the formal argument type of the invoked method. All the fields from the untyped object are matched against the fields in the formal argument type. This process of converting data received from Flash clients into formal argument types is called "type adaptation." Although the example shown is rather primitive, FlashORB can easily convert even the most complicated data structures. It also works well with the built-in collection and utility classes. For example, an instance of ActionScript Array can be adapted to System.Collections.ArrayList or System.Collections.Stack or any other data structure where data is organized in a linear fashion.

Security

The lack of any security is the Achilles heel

of most Flash Remoting servers. A particularly egregious example in Code V demonstrates the severity of this omission. Since all public classes and their corresponding public methods are automatically exposed for Flash Remoting invocations, the code can shut down any Java VM or IIS application.

As you can see in Code V, without security restrictions in place it takes just a few lines of code to bring down the entire server. Therefore, securing Flash Remoting applications is of critical importance. Using FlashORB's configuration file or the management console, developers or administrators can restrict or grant access to the code identified by the package/namespace name or full class name. There are four types of restrictions that can be grouped together: restriction by role name, IP address, subnet address, or hostname. For example, to prevent the code above from working, FlashORB can be configured to reject access to java.lang.* for any requests arriving from clients whose IP addresses match the *.*.*.* pattern.

Remote References

The concept of remote references is not new in distributed computing.

(line 3) interface and providing a method that returns an instance of itself (line 9). The ActionScript code shown in Code VII demonstrates how to obtain a remote reference to an instance of the RemoteReferenceClass class and invoke its methods.

A Flash Remoting service invokes a method on a server-side object (line 5). Since the method on the server side returns an instance of Flashorb.IRemote,

Management Console

The FlashORB management console is a Flash Remoting application designed to simplify the tasks of managing and configuring the product. Currently the console provides the following capabilities: display and/or configuration of the product runtime information, FlashORB security, logging categories, logging policies, handler chains, and object factories. Also, the console exposes the Call

image 1

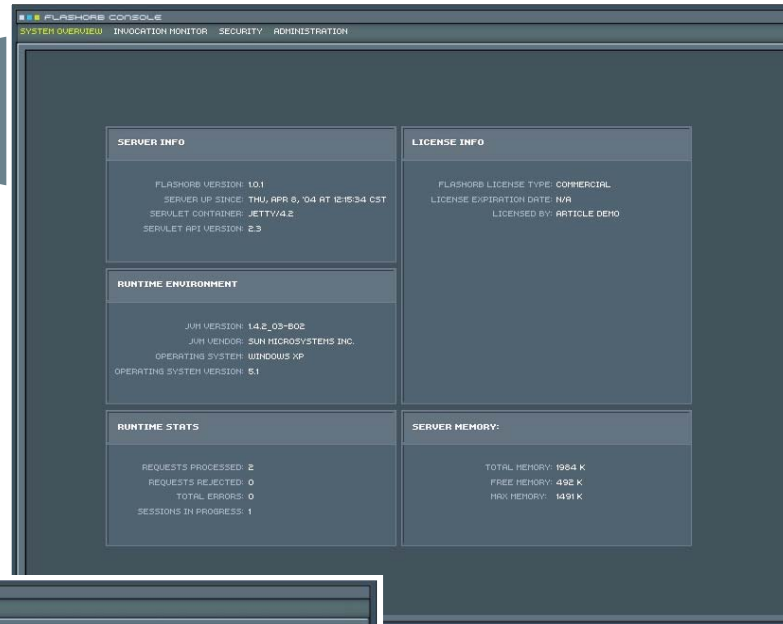
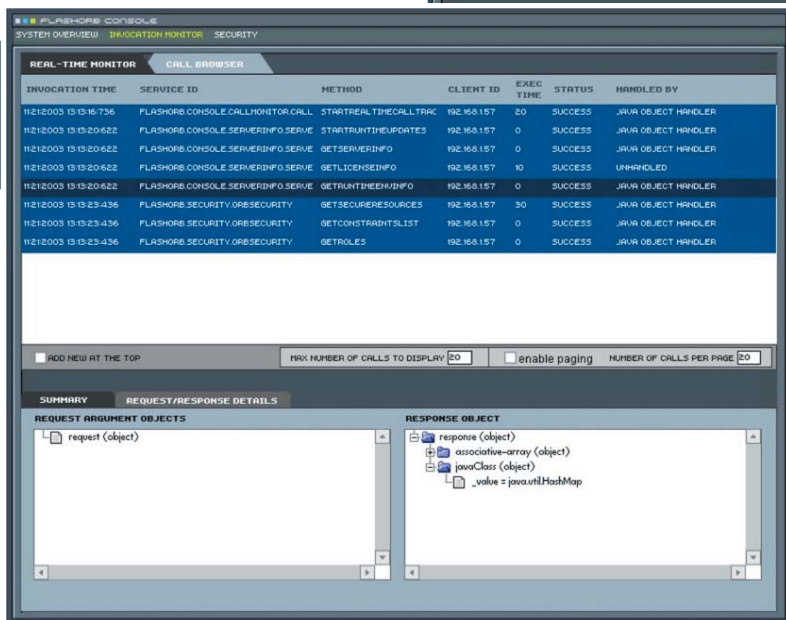


image 2



However, it is a novel in the area of Flash Remoting. FlashORB is the only server supporting remote references. If you're wondering what they are and how it works consider the example in Code VI.

Remote references are client-side proxies to server-side objects. On the server side a remote reference is created by developing a class that implements the Flashorb.IRemote

FlashORB serializes it as a remote reference. The client side receives and caches the reference (lines 7–9) and then performs a remote method invocation via the remote reference object (line 11).

This feature brings the UI and the server sides closer together and provides an extra level of sophistication for the client/server applications.

Trace feature described in greater detail later in the article. Image 1 is a screenshot of one of the modules in the management console.

XML Socket Server

FlashORB was designed to provide a rich framework for building interactive rich Internet applications. The requirements of the rich-client applications go far beyond Flash Remoting and demand support for various message exchange paradigms: point-to-point, broadcast, unicast, server event push, etc. FlashORB includes an XML Socket Server to address these needs. The socket server is very lightweight and is optimized for the maximum performance. As the name suggests, the primary usage of this feature is processing XML documents; it can, however, also process binary messages. It has a rich set of easy-to-use APIs to facilitate building interactive messaging applications with the features like chat, message broadcast, event notification, and asynchronous server-side updates. For example, FlashORB Management console uses XML Socket Server to receive a continuous stream of

With more than 15 years of software engineering practice and 8 years of distributed computing experience, Joe Orban plays a key role in the day-to-day operations of Midnight Coders. Joe is responsible for product architecture, strategic positioning, and developer communication. orban@flashorb.com



server "runtime stats" without resorting to a sluggish polling architecture.

Call Tracing

Call tracing is a concept slightly similar to what Flash developers know as the "NetConnection Debugger." If you are not familiar with this concept, NetDebug is a debugging facility that exposes detailed information about Flash Remoting calls. The biggest disadvantage of this, however, is that NetDebug works only in the development environment for calls that occur during the current execution of the application. It is impossible to inspect Remoting calls from previous application runs. Unlike its Macromedia counterpart, FlashORB call tracing is a server-side feature. As a result, it works regardless of whether the Remoting application was executed in a development

environment, a standalone Flash player or in a browser. When the feature is enabled, the server efficiently persists the data about Flash Remoting calls. This allows the Call Trace subsystem to provide data about the invocations that occurred during the current session, or any previous session. Additionally a facility is provided to apply search filters onto the data in order to highlight specific invocations. For example, you can create a filter to search for all invocations where invocation time is greater than 500ms and method name contains verb "set."

The FlashORB management console provides a graphical Call Tracing module to make this information easily accessible to developers. The module allows observing invocations in real time as well as browsing and searching the call trace store. Each invocation displayed in the module can be

inspected to get the details about method argument values and the return value. Image II is a screenshot showing the Call Tracing module from the console.

Conclusion

As application developers seek new ways to differentiate their software and to improve the end-user experience, the client/server development frameworks must be able to provide a new, rich set of features. Flash Remoting is a great client/server integration approach, but it must rapidly evolve to simplify the integration and shorten the gap between UI front ends and server-side components. FlashORB is an example of a quickly evolving, feature-rich development framework. A free evaluation copy is available at www.flashorb.com; try it out! ☺

code I

```
namespace Flashorb.Examples
{
    public class SampleClass
    {
        public int EchoInteger( int i )
        {
            return i;
        }
    }
}
```

code II

```
1 #include "NetServices.as"
2
3 conx = NetServices.createGatewayConnection
  ( "http://remoting server url" );
4 service = conx.getService( "Flashorb.Examples.SampleClass", this
  );
5 service.EchoInteger( 5 );
6
7 function EchoInteger_Result( result )
8 {
9     trace( "received back " + result );
10 }
```

code III

```
namespace Flashorb.Examples
{
    public class HRDepartment
    {
        public Employee AddEmployee( Employee emp )
        {
            // some business logic here
            return emp;
        }
    }

    public class Employee
    {
        String Name;
        String Title;
        long Salary;
    }
}
```

code IV

```
1 #include "NetServices.as"
2
3 conx = NetServices.createGatewayConnection( "http://remoting serv-
  er url" );
4 service = conx.getService( "Flashorb.Examples.HRDepartment", this
  );
5
6 var employee = new Object();
7 employee.Name = "Joe Orbman";
8 employee.Title = "Chief Architect";
9 employee.Salary = 200000;
10 service.AddEmployee( employee );
11
12 function AddEmployee_Result( result )
13 {
14     trace( "received back employee " + result.Name );
15 }
```

code V

```
Shutting down unsecure Java VM
#include "NetServices.as"
conx = NetServices.createGatewayConnection( "http://remoting server
url" );
service = conx.getService( "java.lang.System", this );
service.Exit( 1 );
```

```
Shutting down unsecure .NET application
#include "NetServices.as"
conx = NetServices.createGatewayConnection( "http://remoting server
url" );
service = conx.getService( "System.Environment", this );
service.Exit( 1 );
```

code VI

```
1 namespace Flashorb.Examples
2 {
3     public class RemoteReferenceClass : Flashorb.IRemote
4     {
5         private string data;
6
7         public void IRemote getRemoteReference()
8         {
9             return this;
10        }
11
12        public void setData( string data )
13        {
14            this.data = data;
15        }
16
17        public string getData()
18        {
19            return data;
20        }
21    }
22 }
```

code VII

```
1 #include "NetServices.as"
2
3 conx = NetServices.createGatewayConnection( "http://remoting serv-
  er url" );
4 service =
  conx.getService( "Flashorb.Examples.RemoteReferenceClass", this );
5 service.getRemoteReference();
6
7 function getRemoteReference_Result( result )
8 {
9     this.remoteReference = result;
10    // calling remote method
11    this.remoteReference.setData( "Remote References are cool" );
12 }
13
14 function setData_Result( result )
15 {
16    this.reference.getData();
17 }
18
19 function getData_Result( result )
20 {
21    // should print "Remote References are cool"
22    trace( result );
23 }
```



HostMySite.com

Built for ColdFusion Pros

by ColdFusion Pros

plans from

\$8.95 / mo.

FREE Domain Name*

FREE Setup

FREE 2 Months

- 24 / 7 / 365 Phone Support
- 99.9+% Uptime
- Macromedia Alliance Partner
- "Full Control" Panel
- CFMX 6.1 or CF 5.0
- SQL Server 2000 or 7.0
- Custom Tags Welcome

Visit www.HostMySite.com/mxdj for:

2 Months Free

and FREE Setup on Any Hosting Plan*



**"When it comes to ColdFusion hosting,
HostMySite.com rules them all!"**

James Kennedy
mbateam.com

*offer applies to any annual shared hosting plan

call
today!

877•248•HOST
(4678)



Do It with
Style





by charles e. brown

The value of any piece of software is in how much time and effort it can save you. What good is the software if it is not going to increase productivity and efficiency? In several of my past articles, I have examined ways of automating common or repetitive processes. To that end, I am continuing the discussion this month. One of the great new features of Fireworks MX 2004 is the Styles Panel, shown in Image 1.





This feature allows you to capture the properties of an object for use, later on, in another object. As an example, suppose you were to create an object

with color shading and a bevel effect. Further, let's suppose that this particular job often. We can simply create it once and capture it as a Style. Each of the buttons is referred to as a preview icon.

The wording I just used is important. Notice I didn't say that you create the attributes in the Style, but simply capture them for later use. As you will soon see, we can determine which attributes we want to keep or not keep.

Let's begin by creating a simple Style.

Creating a Style

Let me start by doing something simple in order to show how easy it is to use.

To use the Style in another object, just select the object and then click on the preview icon in the Styles panel. As you can see, applying the Style is extraordinarily easy.

You should be aware that Fireworks saves many effects, such as shadow, glow, bevel, emboss, and stroke size, with absolute values. This means that a Style you apply to a small object may look different with a large object. For that reason, I may have similar styles for different size objects.

Editing a Style

To edit a Style, it is important that you first make certain no objects are selected. What I usually like to do is select **Select>Deselect** from the menu. That way, you know nothing is selected. I then double-click on the preview icon in the

"I am always amazed when I see a software user spend a large amount of time learning a complex feature, but then completely ignore a very simple feature"



I am going to set up a simple object with a fill color and stroke as shown in Image II.

Obviously, it's not very exciting. However, it will give you a good working knowledge of using Styles for more complex scenarios.

Make sure that object is selected before you do the next step. In the Styles panel, click new. This is shown in Image III.

Here you can decide what you want to name your Style, as well as the properties you want to retain for the Style. For instance, in this case, if you were to shut off the Fill type and Fill color, you would just be left with the Stroke properties.

Notice as I stated earlier, we are not creating the Style's properties here. Those come from the object itself. We are only deciding which attributes we want to keep.

Once I say OK, the style is then added to the Styles Panel, with a preview icon, and can be used at any time.

Styles panel. This opens up the dialogue box shown in Image III.

If you needed to change the actual attributes, such as the color of the fill or the stroke size, you would need to create a new Style altogether.

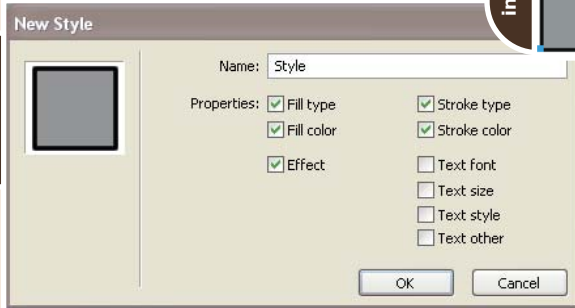
One handy feature is that you can have two styles with the same name. What I sometimes like to do is apply a Style to an object, change the attributes in the object, and capture the new attributes to a Style with the same name. I then delete the original Style.

Once again, you cannot directly change the attributes in the Style.

Importing and Exporting

As a designer, many times I might need to share styles with various other members of the team. Styles can easily be shared with other Fireworks users by using the Import and Export features found in the Options menu.

You first select the Style and then



the Styles that you want. Once they are selected, you then export as you would an individual Style. All the Styles would be saved to the one file name. The user importing the Styles

mean the original settings, but also deleting any custom Styles you might have.

Simply select the Reset Styles menu item from the option menu. You will receive a warning message and, after you select OK, the panel would be reset and any custom Styles would be gone (as well as the subsequent preview icons).

Conclusion

I am always amazed when I see a software user spend a large amount of time learning a complex feature, but then completely ignore a very simple feature.

As you can easily see, Styles is relatively simple to use, but is a powerful time-saving tool. This is in keeping with my previous articles discussing reusability.

Next month, we are going to examine situations where external programs might be useful. ☺

Charles E. Brown is the author of Fireworks MX 2004 Zero to Hero and Beginning Dreamweaver MX 2004. He also contributed to The Macromedia Studio MX Bible. charles@charlesebrown.net

select Export Styles in the panel's option menu. You can decide where you want the file saved and, after naming it, the file will have an extension of .stl.

If you need to load an exported Style, simply select Import Styles from the option menu and browse to the file's location.

As a handy feature, you can export multiple Styles at the same time. You could click on the first Style and then shift-click on the last Style if they are grouped together or, if they are not adjacent, you could ctrl-click on

would then get all the Styles in the file.

One other advantage is that, from time to time, you can find some nice Styles in the Fireworks section of the Macromedia Exchange. It is worth taking a look from time to time at www.macromedia.com and locating the Exchange link in the navigation bar.

Resetting the Styles

There may be a time when you want to reset your Styles panel back to the default settings. This would not only

Once you're in it...



...reprint it!

- Wireless Business & Technology
- Java Developer's Journal
- XML Journal
- ColdFusion Developer's Journal
- PowerBuilder Developer's Journal

Contact Carrie Gebert
201 802-3026
carrieg@sys-con.com



Re Prints



the object panel

by ron rockwell

he Object panel is the key to virtually everything you can do in FreeHand MX. Most of us don't take the time to understand the Object panel, however, and it becomes a source of frustration.





The Object Panel

It seems as though every time you look at the Object panel, its appearance has changed. That's because the panel changes to reflect the attributes and fea-

tures of whatever is selected or "live" in the document. Vector objects will display fill, stroke, and effects information; text will show font name, size, color, and more. If you select an imported graphic, the panel will tell you what kind of graphic it is and will provide a button to allow you to edit the graphic in Fireworks or Flash, or the Links button so you can change the object or gain further information about it.

To compound your initial confusion, effects that are added to any object appear in a hierarchical list that can be rearranged – sometimes. Obviously, an understanding of this powerful panel is in order. By the way, if you don't see the Object panel, go to Window>Object (Cmd/Ctrl+F3) to pop it up – you can move it anywhere on your screen that's convenient.

By default, the Object panel is docked with the Document panel in the Properties "barge." To select one or the other, click its tab. If you want to separate the two panels, click on the Properties Options icon in the top-right corner of the barge and select Group Object With (or Group Document With), then either choose another panel to dock with, or New Panel Group. Other panels may be added to the barge in the same manner, but these two panels get a lot of use and should probably remain in the default arrangement.

Image I shows the Object panel with a simple path selected. There are three buttons at the top left of the panel. The first has a pencil and a plus sign – this is the Add Stroke button. In the middle is the Add Fill button, which shows a paint bucket and a plus sign. The third button has an embossed dot with a drop shadow, and indicates the Add Effects feature. To the right side of the panel head are two icons that have trashcans. The large trashcan on the right deletes any item in the menu that is highlighted. The smaller trashcan with the red menu marks is used to delete an entire branch from the menu. If you had added a fill and a stroke and an effect or two to an object, clicking the Delete Branch button would remove the fill, stroke, and all the effects, leaving the original object untouched. If there is no "branch," this button is grayed out.

The rest of the panel is divided into two areas: the Properties list and the

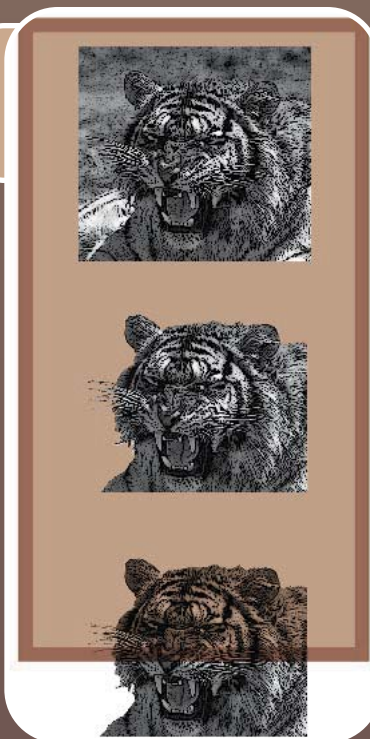
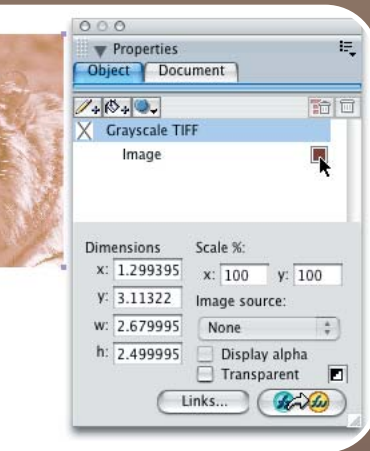
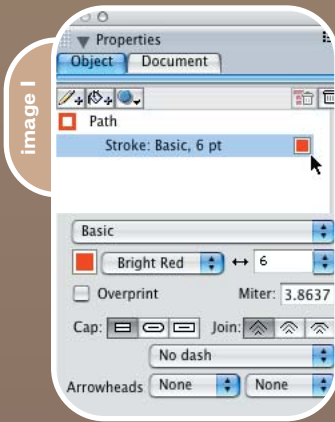
Attributes area. Depending on the object that is selected, and the property in the list, the contents of the Attributes area will display different information.

We could take several pages to describe the various looks and types of information that the Object panel can provide. To alleviate boredom, it might be easier and more informative to see how everything works in action. I've designed a logo for a phony product in order to use as many tricks and tools as possible. It will be obvious to you that many of the tricks wouldn't be used in a decent logo, but you can at least get an idea of what can be done. Just as obviously, some steps have been added just to show a FreeHand feature.

Imported Graphics

I started by importing (File>Import) a grayscale image of a tiger. In order to separate the tiger's head from the background and the rest of his body, I needed to trace an outline with the Pen tool. You can change the color of a grayscale image, so I made it a lighter color in order to make the tracing more visible. To do so, I created a brown color in the Mixer panel and dragged a swatch of the new color onto the Image properties Fill box in the Object panel (see Image II). Now tracing the tiger's outline would be less confusing. The Object panel displays the size and location of the image on the page, along with any scaling changes I may have made. The Image Source dropdown menu would be active if I had color management turned on. The Links button is active only when an imported object is selected. Clicking the button opens the Links panel, where you can see the names, locations, and sizes of the files; extract, change, or embed the files; or select a link from the list and select it (show) in the document.

Image III shows examples of the use of alpha channel and transparency. The top image is a simple grayscale image placed on top of a blue fill. The middle image has an alpha channel and it is displayed via the button in the Object panel. The alpha channel has been knocked out of the image, allowing the background color to show. On the bottom of the figure, the alpha channel has been displayed, and Transparency has been checked. Doing so allows the back-



ground fill color to show through the grayscale image.

For my logo, I wanted to have a posterized look for the tiger's head, so I selected the grayscale image (CMYK or RGB images can't be adjusted like this) and clicked the Image Adjust button in the bottom of the Object panel. Image IV shows the result. I clicked the stairstep icon to get a predefined four-color posterization for the image. If I wanted, I could move the individual bars in the Image Adjust window to create a custom curve as you would adjust the Curves panel in Fireworks...which brings up the next button – the roundtrip Fireworks editor. Clicking this button opens your image in Fireworks for any editing you would like to do.

Interestingly enough, any scale or color modifications you've made to an image in FreeHand will not be applied when the image opens in Fireworks. But be careful when doing the roundtrip. You have a choice of using the original image or a PNG of the image. If you choose the original, any changes you make will be permanent, and there's no going back. However, if you choose the PNG option, you will be asked to find a Fireworks PNG document to work with. If the image you've selected isn't a Fireworks PNG, your only option is to open the original and Save A Copy as a PNG. Then you can click the Done button, which will return you to FreeHand. Once there, place the PNG you just created and click the Fireworks roundtrip button. At that time you can make any adjustments you wish. Clicking the Done button brings you back to FreeHand, where you'll see your updated image. Please keep in mind that when you select a grayscale TIFF in FreeHand and use the Fireworks roundtrip button you will convert the grayscale to RGB, and whether you use the PNG option or work directly on the original, the original grayscale will be modified permanently. Be careful!

A gray tiger in a logo isn't too exciting, so I changed the colors by employing Xtras>Colors>Name All Colors to add four gray tones to the Swatches panel. Then I created new colors for the tiger in the Mixer panel and used the Find & Replace window to change colors. That was a simple matter of selecting a gray

tone in the From column and choosing a color in the To column.

At this point, I had a traced, posterized tiger on the drawing board. I cut (Cmd/Ctrl + x) the posterized tiger, selected the outline path, and chose Edit>Paste Inside to create what other programs call a clipping path. The tiger's head was now separated from its background, and finished for the moment.

Text Handling

As soon as you select the Text tool, the Object panel changes appearance to give you a multitude of text attribute options. Image V shows the initial attribute area setup. There are so many variables that a separate article is needed to do it justice.

The product needed a name, so I typed "TIGER CHOW" in a bold, heavy font (Block Heavy), with a carriage return (Return/Enter key) between words, and centered both lines of text.

I drew a circle with the Ellipse tool by holding down the Shift and Option/Alt key and clicking/dragging from the center of the tiger's face. The circle wasn't quite centered, so I added the Spacebar with my thumb and was able to drag the constrained circle to a better position.

Leaving the circle selected, I Shift-selected the text block and chose Text>Attach to Path (Shift+Cmd/Ctrl+Y). "TIGER" went to the top of the circle, and "CHOW" moved to the bottom (see Image VI: 1). And, boy does it look ugly! FreeHand admittedly has a problem in that it doesn't anti-alias fonts after they've been attached to paths or rotated/skewed. It will print fine, but looks a little scary on the monitor. Have faith.

The text was given a yellow-orange "tiger" color, but looked pretty wimpy, so I added a heavy black outline (8 points) by clicking the Add Stroke button in the Object panel. If you've got typographical sensitivities like I do, you can understand why I was horrified – the bold stroke encroached on the inside of the letterform (see Image VI: 2), and the text was getting uglier by the minute. There are times when this will work – but I can rarely find

them. Try as you might, you cannot rearrange the stroke and fill in the hierarchy of the Object (Text) panel. The stroke will always be on top. So, to get above the problem, so to speak, I cloned the text and deleted the stroke from the Properties list. Bingo! The text was outlined as I had in mind, (see Image VI: 3) and my sensibilities were calmed, even though the orange letterforms were truncated due to the transforming and kerning I had done.

At this point I should tell you that you can eliminate the poor text rendering entirely by choosing Text>Convert to Paths (Shift+Cmd/Ctrl+P). Since the text becomes vector art at that time, rendering is not a problem – but you cannot edit the text as I'll do later in the article.

The text was still a little on the boring side, so I selected the top text block and chose Bevel and Emboss>Inner Bevel from the Add Effects drop-down menu. After adjusting the settings to my pleasure, I pronounced the text done (see Image VI: 4).

Wrapping Up the Cat

Okay, the text is okay, now to add some punch to the cat. I started by selecting it and moving it to a new layer

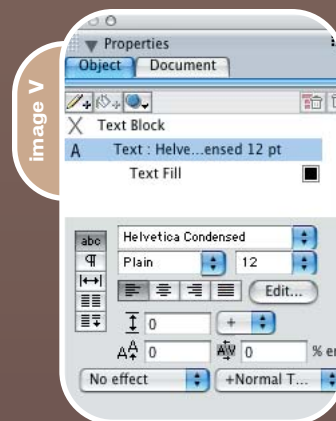
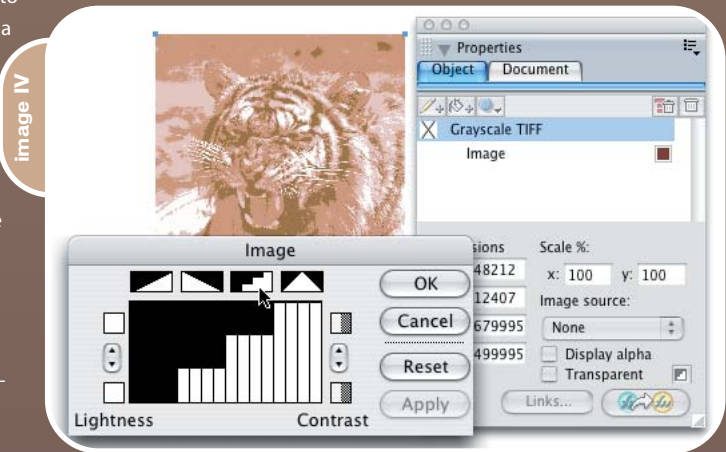




image VI



above the text layer. To accomplish that, I went to the Layers panel, chose New from the Layers options drop-down menu, and named the new layer "Tiger." I switched to the Scale tool (keyboard shortcuts really help in FreeHand, the Scale tool default is F10, but I use Shift+Cmd/Ctrl+S so I don't have to move my hand too far out of typing position). I enlarged the tiger's head so it overlapped the text enough to make it interesting. But that wasn't enough, so in the Object panel I selected Add Effects>Shadow and Glow>Drop Shadow, and increased the distance to really raise the head above the text (see Image VII: 1).

That was okay, but it still lacked something, so back to Add Effects>Bend (see Image VII: 2). Bad idea. Kitty is wet and

mad. The solution was to clone and hide the tiger's head (View>Hide Selection). I enlarged the original tiger head. Then I chose Edit>Cut Contents and deleted the colored image. That left the outline path of the tiger, which I filled with red. Now when I chose the Bend effect, I got the splash I was looking for. I sent the red splash behind the text, and chose View>Show All to bring the head back into view. The splash looked a little stark, so I added a glow (Add Effects>Shadow and Glow>Outer Glow) to soften the effect and give a little added weight to the logo (see Image VII: 3).

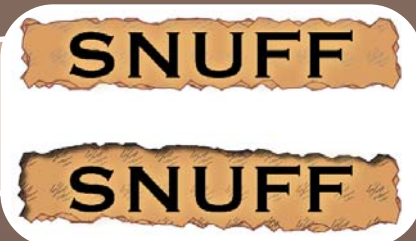
Editing

I felt pretty good about my efforts so far, but then my wife told me that the

image VII



image VIII



added a Ragged effect, with three copies. Then for good measure, I tacked on an inner shadow.

I set the text for the box and thought that the whole arrangement was still a little bland. So I selected corner points on the rectangle and dragged out control handles to give the rectangle a slight S-curve as shown in Image VIII. Notice in the figure that all the ragged lines, the solid fill color, the tiled fill, and the inner shadow adapt immediately to the new path without any work on my part. FreeHand takes care of everything!

I still wasn't satisfied, so I applied an envelope to the rectangle and text. Again, the effects follow my changes perfectly. A few more tweaks of the logo in

product name is "Tiger's Chew," not "Tiger Chow." Boy, I have to pay closer attention to her when she's talking on, and on, and on... If I had converted the text to paths way back at the beginning, I'd have a lot of work ahead of me, and I'd be pretty much like the bent cat in Image VII: 2. Luckily, I chose to go with live text, so it was a matter of choosing Edit>Find & Replace>Text. When that dialog box opened I typed "TIGER CHOW" in the Find field, and "TIGER'S CHEW" in the Replace field. I clicked the Find First button, then Change All, and closed the box. All the effects, strokes and basic placement are still in place. This task could have taken several minutes had the text been converted to paths, but was done in a few seconds by keeping the text live.

Finishing Touches

The product had to be mentioned, so a rectangle was drawn and ungrouped. In the Object panel, I added a light fill color, and for a little texture, I applied another fill. This time it was a tiled fill made of a few lines to indicate patches of hair. Since the tiled fill is above the solid colored fill, both are seen. I added a 2-point brown stroke to the rectangle, but for interest I

A new tool for MX professional developers and designers...



ADVERTISE

Contact: Robyn Forma
robyn@sys-con.com
(201) 802-3022
for details on rates
and programs

SUBSCRIBE

[www.sys-con.com/
mx/subscription.cfm](http://www.sys-con.com/mx/subscription.cfm)
1 (888) 303-5282





general and I had my final logo as seen in Image IX.

Objects with a Grain of Salt

There are several things about the Object panel that can drive you crazy until you figure out what's going on. Number one on most people's list is when you attempt to change attributes on objects within a group. I'll start with text objects. When you have live text, you can change the fill color and/or add a stroke and change its color from any of the usual spots – Swatches, Mixer, Tints, or Object panels, or the color wells in the Tools menu. Nothing new here. But if you convert the text to paths you'll notice that all indicators show a fill and stroke of none, and you can't seem to change the colors at all. That's due to the text characters being in a group, and in some

letters, compound paths (o, e, a, P, Q, and so on). In order to change the color, you have to change the way you think about changing color. First you have to burrow down into the group until you reach the lowest common denominator. The quickest way is to use the Edit>Subselect command. Depending on the situation, one or two applications of this command will fill the panels with your color options. The easiest way to Subselect objects in a group is to create a keyboard shortcut. I use Cmd/Ctrl+Opt/Alt+X; you can make it anything you want – there is no default shortcut.

"Normal" grouped objects work in the same manner as with text. The first time you look in the Object panel, you'll see "Group" as the Property area's first listing, with Contents listed beneath. Double-click

Contents to Subselect the objects within the group, and the number of objects will be listed, with an entry for whatever Stroke and Fill is applied. If your items have the same stroke and fill, you can make a change to them all at this point. However, if you have multiple objects that have different color attributes, the stroke and fill wells will have a dash in them, indicating multiplicity. To make changes, you must isolate individual items. At this point, you could hold down the Shift key and deselect items you don't want to change, but it's almost easier to use the Ungroup command in the beginning and go straight to changing attributes.

If you apply a stroke or fill to the group, all objects will receive the new fill color (or tiled fill, or patterned fill – whatever you choose), and gain the new stroke. All this is in addition to whatever strokes and fills the objects have originally. It's important to keep in mind that you can easily move strokes and fills up and down in the Object panel – with the exception of live text.

Summary

FreeHand's Object panel is a fascinating part of the program's operation. Spend some time on non-critical drawings and text treatments until you become proficient with all the variables within the Properties area and the Attributes area.

Acknowledgments

Many thanks to John Nosal, Delores Highsmith, David Spells, Peter Moody, and other engineers at Macromedia for the technical editing they provide. ☺

Illustrator, designer, author, and Team Macromedia member Ron Rockwell lives and works with his wife, Yvonne, in the Pocono Mountains of Pennsylvania. He is the author of FreeHand 10 f/x & Design and co-authored Studio MX Bible and the Digital Photography Bible, and he is MXDJ's FreeHand editor. He has Web sites at www.nidus-corp.com and www.brainstormer.org. guru@brainstormer.org

image IX



The bible for <CF_Developers>



ADVERTISE

Contact: Robyn Forma
robyn@sys-con.com
(201) 802-3022
for details on rates
and programs

SUBSCRIBE

www.sys-con.com/
cfdj/subscription.cfm
1 (888) 303-5282



COLDFUSION Developer's Journal



IN

Part 1 of this two-part article (MXDJ, Vol. 2, issue 3) I showed how to invoke a Web service for the purpose of validating user input. Part 2 will delve a little deeper into the **ColdFusion MX** language structure specifically designed to handle XML result sets such as those returned from a Web service. The goal is to consume a Web service that will return headline news articles for a user-specified topic. When finished you will be able to add a live user-interactive newsfeed to your Web site.

A WEB SERVICES EXAMPLE PART 2

MAKING HEADLINES

by richard gorremans

LEG





This article will cover:

- **XmlParse():** Returns an XML Document object
- **XmlRootElement:** Root element for an XML Document object
- **XmlName property:** Element name



image I



image II

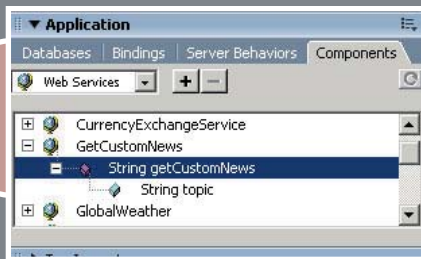


image III

- **XmlChildren[1]:** Referencing the first row in an array containing the child nodes for a specified element
- **XMLSearch():** Search routine that returns an array of nodes that match a provided XPath expression
- **Error trapping with <cftry> and <catch>**

Choosing the Web Service

One resource for locating Web services is located at www.xmethods.com, where you will find the Web service (www.xmlme.com/WSCustNews.asmx?WSDL) that will be used for this article. This particular Web service returns a string that is in an XML format; other Web services may return different values. The Axis engine that is built into ColdFusion MX will deserialize these return values, creating a data type you can use.

Testing the Web Service

Now that the decision has been made as to what we want done and where to find the Web service that will provide the necessary information, the next step is

testing the Web service. Testing will answer two questions.

1. Does the Web service work?
2. What is the Document Tree for the returned result set?

The easiest way to perform these tests is using the Dreamweaver MX Components tab that is located on the Application panel (see Image I). From this tab you can add information on a Web service, including a document tree. Click on the + button to open the "Add using WSDL" window (see Image II), enter the

URL for the WSDL file, and click on the <OK> button.

A new entry will be added to the Component tab of the Application panel. Click on the new entry to view the document tree

for the WSDL file (see Image III).

Using the file administrator, open a new document and drag-and-drop the new Web service into the new document as shown in Image IV.

You have now created the basic coding necessary for consuming the Web service. The code sample created by Dreamweaver MX shows one argument being passed to the Web service (topic), and the result set returned from the Web service will be stored in the

aString variable. Replace "enter_value_here" with the word "Sacramento", which will provide a topic for the Web service to search on for the test.

The next step will be to add the coding to display the results returned from the Web service (see Image V). For displaying the results the XmlParse() function and <CFDUMP> tag are used. XmlParse() is a new function that accepts XML code as a string and returns an XML Document object that represents the various XML elements contained in the document. Image VI shows the results of the <CFDUMP>.

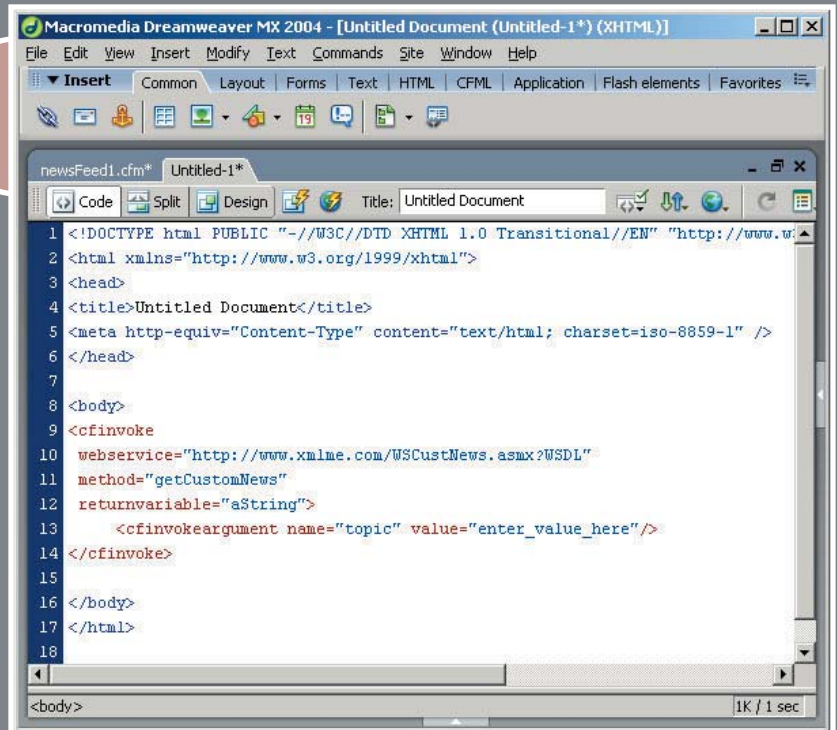
The User Interface

Now that the Web service has been selected and tested, the next step is to create an interface where the user can enter a topic of his or her choice and display the resulting headlines. The following list is a very simplified view of the desired process flow for the user interface.

1. Display entry field for user to enter topic.
2. Invoke Web service, passing user input as an argument and using a default value for topic if one is not provided.
3. Display the result set if there are no errors.
4. Display a message if there is an error.

The single most important variable in the completed form will be the topic

image IV



being passed to the Web service as an argument. A default parameter, sTopic, will be created and a blank value assigned. This variable will control when the <CFINVOKE> tag is executed. The code will be executed only if sTopic has a value.

Another parameter will be the number of articles returned by the Web service, defaulting to zero. If the Web service returns a result set, this variable will be populated with a count of the number of headlines received.

```
<CFPARAM NAME="sTopic" DEFAULT="" />
<CFPARAM NAME="xCount" DEFAULT=0 />
```

With the default values out of the way the next step is to create the GUI. This will consist of a title, one label, an input field for the topic, and a submit button. This will be a rollover form so the CGI.SCRIPT_NAME variable is used to force the page to call itself on post (see Code I). Image VII shows the user input form for topic.

Consuming a Web service, just like accessing a database, should be treated as if there is a good chance the service will be unavailable, the return results are corrupted, or there are no results returned. With a database error you typically receive an error code that can be responded to. When you attempt to consume a Web service, the errors will vary and may be hard to account for. These potential problems are easily dealt with by placing the <CFINVOKE> code inside a <CFTRY>.

The entire code segment is placed inside a conditional statement that prevents the code from being executed unless the form has been submitted and a topic has been provided by the user (see Code II).

In Code II new properties (XmlRoot, XmlName, XmlChildren) and the XMLSearch() function have been introduced. The coding combination ensures that if the provider of the Web service changes the root or child element the program still has a good chance of retrieving the desired information without any modifications.

```
<cfset xRoot = MyXml.XmlIroot>
```

Retrieves information about the root ele-

image V

```
8 </body>
9   <cfinvoke
10    webservice="http://www.xmlme.com/USCustNews.asmx?WSDL"
11    method="getCustomNews"
12    returnvariable="aString">
13     <cfinvokeargument name="topic" value="enter_value_here"/>
14  </cfinvoke>
15  <cfset MyXml = XmlParse(aString) />
16  <cfdump var="#MyXml#">
17 </body>
```

ment.

```
<cfset bNode = xRoot.XmlName>
```

Retrieves the name of the root element. In this case it is "moreovernews" (see Image VI).

```
<cfset cNode =
xRoot.XmlChildren[1].XmlName>
```

Retrieves the name of the first child element. In this case it is "article" (see Image VI).

The combined results of these three <CFSET> calls provide the second parameter needed for the XMLSearch() function. This function will return an array containing the headlines received from the Web service.

The next line (<cfset xCount = arraylen(myNewsFeed)>) returns a count of the number of elements (headlines) the XMLSearch() function found. This count will be used for a looping routine to display the result set and controlling whether or not the results code segment is executed.

The next step is to display the head-

lines to the user. The displaying of this code will be determined by the value of the xCount variable (see Code III). If the Web service returns a result set the value will be greater than zero and can be displayed (see Image VIII). A looping routine will be used to iterate over the result set, displaying the headline_text, source, and harvest_time elements. The url element will be used to create a hyperlink to view the entire story in a separate window (see Image VI).

Typically there are two types of problems with which a Web service will most likely return an error.

1. A connection to the Web service cannot be made.
2. The Web service does not know how to deal with the information provided as the argument (such as a topic where no headlines are found).

To handle these problems a <cfelseif> condition will be added to the code that displays the result set (see Image VIII). Code IV verifies that the form has been submitted and a topic was provided, but the xCount is equal to zero. If either of these problems are encountered a message will be displayed to the user (See Image IX).

image VI

moreovernews	XmlText
article	XmlText
XmlAttribute	
id_123835970	
url	XmlText http://c.moreover.com/click/here.pl?x123835970
headline_text	XmlText Four indicted in BALCO steroids case
source	XmlText The Argus
media_type	XmlText text
cluster	XmlText moreover...
tagline	XmlText
document_url	XmlText http://www.trivalleyherald.com
harvest_time	XmlText Feb 13 2004 12:42PM
access_registration	XmlText
access_status	XmlText

image VII

News Headlines - Coldfusion MX Example

Enter Topic



Image VIII

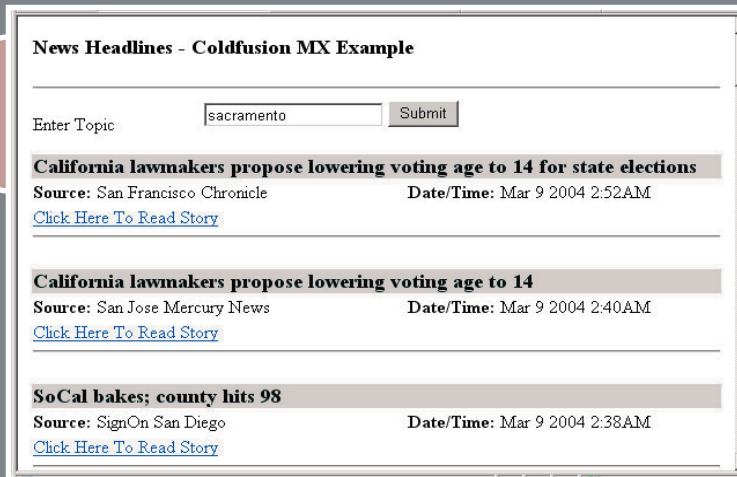


Image IX



Summary

Once you start working with the code presented in this article you will find that there are endless possibilities for creating new types of dynamic user-interactive

Web pages. Companies such as Amazon.com and eBay have already started providing access to their engines via Web services that will greatly shorten development time. For example, with

minimal programming, you can create fully operational e-commerce sites using another company's stock and their Web services.

In my next article you'll learn how to shorten your development time even more by taking this same Web service and applying an XSLT style sheet. Once it's applied, you'll find that your completed Web pages can be more dynamic and, with a few simple conditional statements, display the same information in an unlimited number of formats.

For the past four years Richard Gorremans has been working for EDFUND, the nonprofit side of the Student Aid Commission, located in Rancho Cordova California. A senior software engineer with over 13 years in the business, he has been working as a technical lead producing Web-based products that enable borrowers, lenders, and schools to view and maintain student loan information via the Web.
xbase@volcano.net

```

<table border="0" width="600" align="center">
  <tr>
    <td colspan="2">
      <h3>News Headlines - Coldfusion MX Example</h3>
      <hr />
    </td>
  </tr>
  <tr>
    <td>
      Enter Topic
    </td>
    <td>
      <form name="formTopicSubmit"
        action="#CGI.SCRIPT_NAME#" method="post">
        <input name="sTopic"
          value="#sTopic#"
          width="50"
          maxlength="254">
        <input type="submit" name="tSubmitted"
          value="Submit">
        </form>
      </td>
    </tr>
  </table>

```

```

<cfif isdefined("form.sTopic") and form.sTopic GT "">
  <cfset sTopic = form.sTopic />
  <cftry>

    <cfinvoke
      webservice="http://www.xmlme.com/WSCustNews.asmx?WSDL"
      method="getCustomNews"
      returnvariable="aString">
      <cfinvokeargument name="topic" value="#sTopic#" />
    </cfinvoke>

    <cfset MyXml = XmlParse(aString) />
    <cfset xRoot = MyXml.Xmlroot>
    <cfset bNode = xRoot.XmlName>
    <cfset cNode = xRoot.XmlChildren[1].XmlName>
    <cfset myNewsFeed = XMLSearch(MyXml, "/#bNode#/#cNode#")>
    <cfset xCount = arraylen(myNewsFeed)>

  <cfcatch>
    <cfset xCount = 0>
  </cfcatch>
</cftry>
</cfif>

```

```

<cfif xCount>
  <table border="0" width="600" align="center">
    <cfloop from="1" to="#arrayLen(myNewsFeed)#" index="i">
      <cfoutput>
        <tr>
          <td colspan="2" bgcolor="##66FFFF">
            <font>#myNewsFeed[i].headline_text.xmltext#</font>
          </td>
        </tr>
        <tr>
          <td>
            <font style="font-weight:bold">Source: </font>
            <font>#myNewsFeed[i].source.xmltext#</font>
          </td>
          <td>
            <font style="font-weight:bold">Date/Time: </font>
            <font>#myNewsFeed[i].harvest_time.xmltext#</font>
          </td>
        </tr>
        <tr>
          <td colspan="2">
            <font>
              <a href="#myNewsFeed[i].url.xmltext#"
                target="_blank">
                Click Here To Read Story</a>
            </font>
          </td>
        </tr>
      </cfloop>
    </cfoutput>
  </table>

```

```

<cfelseif isdefined("form.tSubmitted") and trim(sTopic) is not "">
  <table border="0" width="600" align="center">
    <tr>
      <td align="center">
        <font style="font-weight:bold">
          No Information Available For Topic Submitted<br />
          Or unable to connect to Web Service.<br />
          Please try again.
        </font>
      </td>
    </tr>
  </table>
</cfif>

```



MOA Tour

Macromedia Open Architecture quirks, clues, shortcuts, and hints

by tab julius

In this last article (for now) of a series of articles on Xtra development using Macromedia's Open Architecture, or MOA, I thought it'd be nice to take a quick tour through some of the little quirks, clues, shortcuts, and hints I've accumulated over the years. These are in no specific order other than that I've tried to keep related items together.

Director Isn't Loading My Xtra

First, make sure it really isn't loading. Unfortunately, in Director, there is no specific menu item that will show you all the Xtras. In later versions of Director you can type "put the xtralist" (without quotes) in the message window and it will list all Xtras loaded by Director. It's not formatted for easy reading, but it is comprehensive. Other than that, where your Xtra appears depends on what kind of Xtra it is. Scripting, or Lingo Xtras, show up when you type "showxlib" in the message window. In later versions of Director you can find them in the third-party Xtras button on the message window as well.

Sprite/Asset Xtras show up under the Insert menu. Tool Xtras show up on the Xtras menu, and Transition Xtras show up in the transition list.

If you've verified that it's really not there, there are a couple of reasons why it might not show up. First, it has to be in the Xtras folder with a .X32 extension for Windows, or the proper Type/Creator on Mac with the proper resources. It's okay if it's in a subfolder under the Xtras folder, as Director searches all subfolders.

On Windows, the most common reason is failure to include the .DEF file. If you delete all the original source file references from the skeleton template and add in yours instead, it's very easy to forget to add in the .DEF file from the WIN-

PROJ folder. No .DEF file = no loading of Xtras.

If it's not loading and Director is complaining of a "Duplicate Xtra", then you're most likely using a GUID that is already used by another Xtra. Each Xtra must have unique GUIDs, as discussed in an earlier article.

Next, it's possible that something in your registration code is preventing it from registering. Is there a check for a particular resource or version of Director? You should walk through the code and see if there are any exits that might apply.

Finally, is it statically linked to a required external .DLL that it can't find? Because if so, it won't load. A good way to test this is to, on Windows, run Microsoft's DEPENDS.EXE on the file (DEPENDS.EXE comes with Visual Studio, and checks all file dependencies, except for those loaded dynamically at runtime). You might be able to run it on the Xtra in the Xtras folder, or you may have to put the Xtra in the Director folder itself for the test, depending on where you keep your required .DLL and how you have it structured. A similar problem can occur on the Mac with weakly linked libs, although I don't know of any tool, off-hand, that is a Mac equivalent of DEPENDS.EXE.

If you fixed the "problem" but it still doesn't load, it's slightly possible that Director got out of sync with the Xtra. To improve startup time, Director queries each Xtra for its registration information and caches that information; if the cache gets out of sync, it may think your Xtra is unloadable even though it really is. In that case, just delete the cache. The cache file is DIRAPI.MCH and it's safe to delete (but if you're squeamish, just rename it). Director will rebuild the file if it's missing.

Director Crashes at the Splash Screen

Director loads Xtras while the Director splash screen is being displayed. As such, a crash during the splash screen usually is due to an errant Xtra. One obvious place to check is your registration code – use the debugger to walk through it. If it seems clean, and you've got a scripting Xtra, a little-known cause of crashes is an improper message table. At least in earlier versions of Director (have not tried this in MX+), a bad entry in the message table will bring Director to a screeching halt. As an example, I believe a line like the following would do it:

```
"myFunction *, object me, **"
```

or

```
"myFunction * object me, **"
```

I ran into this a number of years ago when I had an Xtra with global commands that I was converting to an object-based model. The original declaration was something like `"* myFunction*"`, meaning that it was global (the first asterisk) and had no restrictions on the number of parameters (the second asterisk). I went through and added "object me" to all the commands but forgot to move the asterisk. Thus it was a wildcard asterisk followed by a specific variable declaration. I do not recall at this point if there was a comma between the asterisk and the "object" keyword. Nevertheless, the net result was that Director crashed big-time; it was not in my C code as far as I could tell, yet it was obviously a result of my Xtra.

It wasn't until after a lot of searching and trying to recall exactly what I had changed that I finally realized it was a change in the message table, and even

then it took a while to track it down. I mention it here because it's easy to make a typo in the message table that results in a not-so-obvious bug that can take forever to find.

The Debugger Isn't Hitting My Breakpoints

Make sure that you're actually debugging your Xtra. I like to compile directly into the Xtras folder in order to minimize any mistakes that might be caused by moving files from the debug or release folder out to somewhere else. If you work on multiple versions of Director, or if your Xtra was originally built for an older version of Director, make sure your debug settings are compiling it into the proper folder (for your current version) and not the old one.

My Xtra Loads, Except in Shockwave

In order for an Xtra to load in Shockwave, there needs to be a flag set during registration that says the Xtra is

Shockwave has its own set of folders where the Shockwave plugin is. An easy way to find it is to simply search for the file "TextXtra.x32" on your system, which will pull up both your Director folder and your Shockwave folder. On Windows, officially there's a registry entry to tell you where it is, but usually you'll find it in `\Windows\System\Macromed\Shockwave 8\Xtras` (note that it says `MACROMED`, not `MACROMEDIA` (8 char limitation). On the Mac, it will usually be `/System Folder/Extensions/Macromedia/Shockwave 8/Xtras`. Also remember that any external .DLLs you need will need to be there too.

Okay, on to some specific programming topics...

Director Goes Through My Function Fine, but Crashes Upon Returning

Did you try to release one of the args from the callPtr? Because that's a no-no. Director owns those args and expects to release them later on. When you use

Then, with the movie, for which you will have a pointer to as a `IMoaDrMovie2` object, you can get at the casts and the score. `GetScoreAccess()` will get you a pointer to the score, and you can use that `IMoaDrScoreAccess` pointer to get your fingers on the sprites, frames, and whatnot. Or, if you want to wander through the casts of a movie, use `IMoaDrMovie2` to `GetCastCount()`, get a cast by name or index, and then with the casts you can get at the cast members.

Ultimately you can traverse the whole tree and get at everything loaded in the movie. You just can't go directly to cast-member `x` of cast `y` of movie `z`, although that might be a nice feature to have.

I'm Getting Crashing Calling My Xtra from Another Thread

Xtras, through MOA, have the ability to call back into Lingo, such as through `IMoaDrPlayer's CallHandler()` function, which will call a handler in the active movie. This works, because Director put every-

“For those writing Xtras, protecting your work is important, if that’s how you make (or supplement) your living”

“Safe for Shockwave”. In other words, you have to explicitly say that the Xtra is safe for Shockwave by default an Xtra is considered not safe for Shockwave.

In Shockwave, you have access to nearly every MOA function that you do when running from a desktop. And you certainly have full access to all of the operating system functions that you normally would. Setting the flag doesn't, in and of itself, set any limitations; it just tells Director that you swear you've taken the necessary precautions to make it safe for Director to load. Such precautions include not retrieving information or files from the user's machine, or doing any tinkering with their system, at least not without providing clear warning to the user and giving him or her the opportunity to opt out.

And remember, to load in Shockwave, the Xtra has to be in the Shockwave Xtras folder, not just the Director folder.

`AccessArgByIndex`, you're not creating an arg, you're simply accessing an existing one. As such, it's not your responsibility to release it.

How Do I Get a Pointer to a Sprite or a Cast Member?

In MOA, to get something you want, you usually have to walk the chain of command, particularly when it comes to casts, scores, and sprites. You can't just instantiate a sprite variable, you have to work your way to it.

Start with `IMoaDrPlayer`, an interface that it's helpful to just acquire in the beginning and keep around. With `IMoaDrPlayer` you can then get a pointer to the active movie (or any of the movies – you can call `GetMovieCount()` and `GetNthMovie()` to walk through the movie list, or just call `GetActiveMovie()` to get the one playing at the time of the call).

thing on hold while it called your Xtra, and now you're calling back into Director, a condition Director allows via MOA. This is considered a synchronous call...Director called your Xtra; you're calling back into Director; it's a straight line (or loop?).

But if you've created a separate thread that needs to call Director, or if you have a window with a callback for an external operating system function that needs to notify Director when something happens, you may run into a crash. This is because Director doesn't have an inkling that you might call it, particularly under Internet Explorer, and your call is wholly unexpected – the equivalent of stopping at a random house at a random time and ringing the doorbell and just going in, rather than having an invitation. Director might be able to accommodate you, but it might not.

If it can't, then you're looking at a potential crash. How do you get around this? By using Director's `Push/Pop Xtra`



context. Basically it works like this – at a “normal” time, when running, when you’ve been called from Director (perhaps when you’re going to set up your thread or your window with a callback), you ask Director for a copy of the movie context via `IMoaDrMovieContext`. Then, later, at the time when you need to call asynchronously into Director, you will need to assert the context, thus making it safe for your call. You do this via `PushXtraContext()`, and when you’re done, you would then call `PopXtraContext()`. Failure to do so, when using threads or your own windows (on Windows), is asking for a crash. It may not happen right away, but you will find that it may die arbitrarily while calling Director if you haven’t asserted the context.

My Xtras Make Windows, but the Screen Doesn’t Refresh When I Move Them

If your Xtra puts up a window, and if you drag the title bar to move the window and Director leaves a trail of windows behind, well, that’s because Director doesn’t know that you opened a window.

Yes, seriously, you have to tell Director that you’ve got a window open. This is because Director has optimized its stage refresh code on the assumption that it is the only window in its process that is open. If you’re going to open a window, you need to tell Director about it. This is done by wrapping it before and after with calls to `IMoaMmWndWin’s WinPrepareDialogBox` and `WinUnprepareDialogBox()`.

On the Mac, there’s a similar issue with dialog boxes, resolved in a similar manner via `IMoaMmWndMac’s MacPrepareModalDialog()` and `MacUnprepareModalDialog()`.

I Want to Sell My Xtra - How Can I Protect It?

This is a much more common question than one might think. For those writing Xtras, protecting your work is important, if that’s how you make (or supplement) your living.

There are commercial tools out there for limiting code use, but typically they’re more for major product manufacturers (like Macromedia) rather than Joe or Jane Developer. Most commonly it’s some kind of a serial number system. The Xtra won’t

work (or will have limited functionality) without a serial number or similar registration code.

What restrictions you put on it depends on you and what your Xtra does. For instance, many developers like to make their Xtra functional in authoring mode, but to run in a projector or Shockwave requires a serial number. This scheme is good because it allows users to try the Xtra in Director’s authoring mode and get used to it and make sure it meets their needs before purchasing it. Usually they don’t buy unless they’re ready and return rates are very low. The drawback is the inevitable last-minute-ohmygosh-I-need-it-now-we’re-shipping-tomorrow emergency order. But it’s a system that works quite well, usually. It won’t work so well for tool-Xtras or authoring-only Xtras.

Another scheme, if appropriate, is to have an authoring version of the Xtra and a runtime version. The runtime one can be shipped and distributed, but it can’t be reused for authoring by someone else. The authoring version has the purchaser’s name in big lights and is not to be redistributed. This is nice because it keeps serial numbers from being distributed in .DIR files, but runs the risk of the authoring version being accidentally distributed and it doesn’t address the issue of try-before-you-buy.

Regardless of which approach you take, you can address the protection by checking for authorization either in your registration code, in your individual functions, or in your handling of the `::Call` function. It is there that you can decide to allow all, some, or no functionality based on whatever criteria you want to set. A call to `IMoaAppInfo::GetInfo()` will get you runtime information such as the `runMode` (whether you’re in authoring or in a projector). Note that some of the info that `GetInfo` supplies, such as a serial number and even a user name and organization name, aren’t available at runtime (in a projector), only at authoring time.

You will then need to come up with a serial number algorithm such that (1) no two serial numbers are alike; and (2) you can verify it’s a valid serial number. You will then need to write a program to generate such numbers, and one to analyze a number and determine if it’s valid.

There is no recommended way to write these. Everyone tries for a tech-

nique that builds in as much protection as possible. You could simply make a unique serial number (by building on date/time and other information), or you could tie it to the user’s name or company by making them enter their name along with the number for verification.

To see if a submitted number, or number and name, is valid you would submit it to a series of tests. These would be the reverse of how you generated the number. Besides just coming up with unique information, you might then want to create a checksum for the number and put it in the number somewhere. Checksums are numbers calculated off of other numbers. Your credit card number has a checksum (usually just a single check digit) that acts as a quick test as to whether or not the rest of the number is correct. For instance, in MasterCard # 5431 2345 6789 0123, the last digit “3” might be calculated from the first 15. If the first 15 ought to come up with a 7, after running through whatever the MasterCard check formula is, but the submitted number is 3, then it’s an immediate indication that the number is invalid for some reason.

You can use the same approach in your serial numbers – but they can be much more complex. Registration codes with letters in them, like “F2aBgADC-33qrUz”, usually decode the letters into numbers, and go from there. Also, parts of the code can be shifted around, scrambled, so ultimately it would be very hard to make a “fake” serial number that would satisfy all the conditions of unscrambling, and check digits, letter conversions, and so on.

Your Xtra would not need to contain a list of all possible serial numbers, it would merely need to have the algorithm to test a provided serial number. If a provided serial number passes all your tests, then you can consider it valid!

Final Question

Will there be more articles on Xtras development?

Maybe, but not immediately. We’re going to move into some advanced Director development for a while. Look forward to some interesting articles on internationalizing your product to run in multiple languages!

Until then, enjoy! ☺☺

Tab Julius has been writing software since the mid-70s, and now works for a software firm developing medical imaging applications, although he still does limited consulting on the side. tab@penworks.com



talking to sprites

Tabs are a basic interface element that Director has tended to ignore. Director MX 2004 introduces a series of new FlashComponent member types to renew and extend the built-in controls, but there is no tab member to be found among them.

by james newton





Tabs combine organization and navigation. They indicate to the user that the current interface provides access to a related set of concepts. For example, your project may require a Preferences window. You could provide one tab for each theme that your preferences need to cover. When the user clicks on a tab, the appropriate screen will appear. Each screen will contain its own input fields, checkboxes, and other buttons.

This article shows you how to create a set of tabs using customized bitmap members and a simple generic behavior. You'll be learning about:

- Matte ink

- Testing commands in the Message window
- Sprite channels and locZ
- Properties and variables
- Variable naming conventions
- Events
- Moving between markers
- Sending messages between sprites
- Behavior instances
- Lists

You can get a preview of the movie that you are going to create at <http://nonlinear.openspark.com/tutorials/tabs.dcr>. You can find the completed movie at:

- Macintosh: <http://nonlinear.openspark.com/tutorials/tabs.sit>
- Windows: <http://nonlinear.openspark.com/tutorials/tabs.zip>

You can download a demo copy of Director MX 2004 from www.macromedia.com/go/try_dmx2004

Proof of Concept

In the first half of this article, you'll create a very simple Tab behavior, just to show that it's possible. In the second half, you'll build a more elaborate behavior which will merit a place in your code library.

Getting Started

Let's start by creating a simple movie with three placeholder screens.

1. In the Score window, add markers named "Marker 1", "Marker 2", and "Marker 3" to frames 2, 32, and 62.
2. Double-click on the Script channel in frame 28, and enter the following script in the window that opens: Create a behavior with the handler...

```
on exitFrame
  go loop
end
```

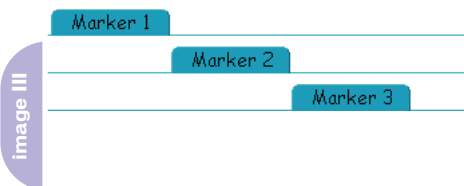
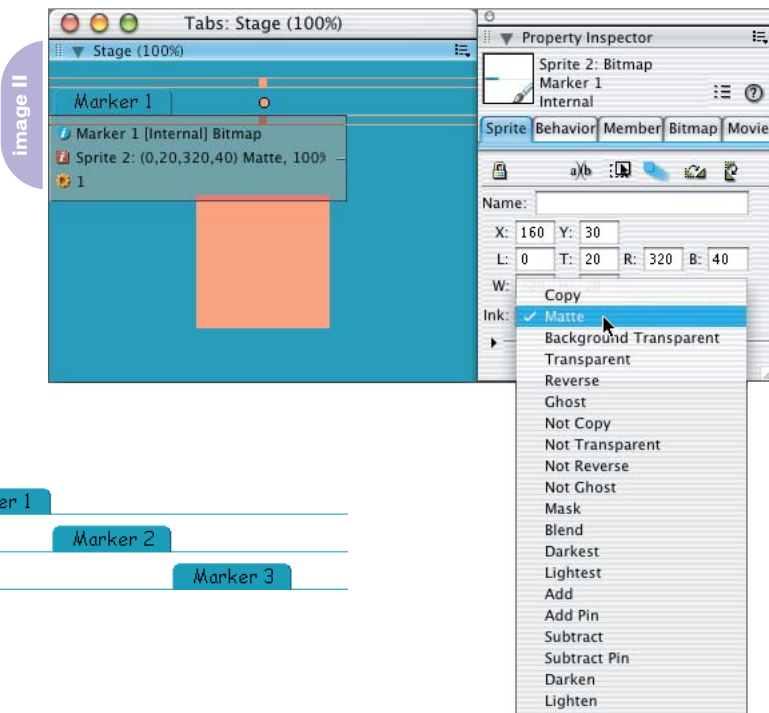
3. Drop the new behavior member on the Frame Script channel of the Score in frames 58 and 88.
4. Using the rectangle tool from the Tool Palette, draw a rectangular Shape sprite on the Stage, in channel 1 and starting in frame 1, and extending to frame 28.
5. Select the new sprite in the Score window, and Alt-drag it (Windows) or Option-drag it (Macintosh) to create new sprites in frames 31 to 58 and again in frames 61 to 88.
6. Using the Property Inspector at the Sprite tab, set the color of these three Shape sprites to red, green and blue. This allows you to tell which section of your movie is playing at any given time.
7. Set the color of the background to something other than white, so that you can create a highlight lighter than the stageColor. I chose color 200 from the Mac System palette - rgb("#006699").

This simple movie is enough for you to test the tabs feature. If you want to make it more like a Preferences window, please feel free to add your own input fields, buttons, and other controls at each of the markers.

Creating the Bitmaps

Let's start by creating a set of bitmap members, one for each of the three tabs that you are going to need. I created my tab bitmap in the Paint Window, in eight steps (see Image I):

1. Create the text.
2. Draw a box around it in a color lighter than the stageColor - rgb("#0099CC") in my case.



3. Draw a darker line down the right-hand side – rgb("#003366").
4. Remove the top corners.
5. Drag the a 3x3 square in the top corners in and down to create a rounded appearance.
6. Fill the box with the same color as the stage.
7. Color the bottom of the box in the same color, including the bottom-right pixel.
8. Add a line to either side of the tab in the same color I used in step 2, to make the bitmap the same width as the stage.

If you prefer, you can use icons instead of text. When you are satisfied, drag the bitmap member onto the Stage, and set its ink to Matte using the Property Inspector (see Image II).

Make two other bitmap members for the other tabs. I simply created a copy of the original and edited that (see Image III).

Name each bitmap member with the name of one of the markers in the Score window. You'll see why we use marker names for the bitmap members in a moment.

Interacting with the Tabs

Place all three bitmaps on the Stage using Matte ink. Notice that the tab in the highest-numbered channel appears to be in front of all the others. I'll assume that you have your "Marker 1" tab in sprite 2, "Marker 2" in sprite 3, and "Marker 3" in sprite 4.

Run your movie, and type the following command in the Message window:

```
sprite(2).locZ = 5
```

The tab in sprite 2 should now appear in front of the others. Try again, this time setting the locZ of sprite 3, and then sprite 4 (see Image IV). What happens when you try using sprite 2 again?

Normally, the order in which sprites are drawn on the Stage depends on the sprite channel they are in. A sprite in channel 2 will appear in front of a sprite in channel 1. When you move a sprite around the stage, you change the value of its locH and locV properties – its horizontal and vertical location. The locZ property affects the layer in which the

sprite appears.

As you have seen, you can force Director to draw the sprites in a different order. When you set the locZ of a sprite to integer number, you tell Director to treat it as if it were in the channel with that number. You can thus give several sprites the same locZ value. However, Director will still distinguish between sprites with the same locZ: it uses their original sprite channel numbers to determine the order. Now that sprites 2, 3, and 4 all have a locZ of 5, sprite 4 will appear in front of the others.

In order to make sprite 2 appear in front, you have to set the locZ of the others back to their original value. By default, the locZ of a sprite is the same as the number of its sprite channel.

How a Behavior Knows Which Sprite It's In

Lingo has a specific keyword for this number: `spriteNum`. In a behavior, you can declare `spriteNum` as a property, and then use `sprite(spriteNum)` to refer to the sprite to which the behavior is attached. The property declaration must occur before any reference to the property. Traditionally, all the properties used by a script are declared at the beginning.

Creating a Simple Behavior

Let's make the tab sprites react to a click. Select all three tab sprites, then right-click (Windows) or Ctrl-click (Mac) and select the Script item in the contextual menu that opens. This will open the Script window. Enter the code that appears in Image V.

Note that this is a quick and dirty piece of code. It uses hard-coded sprite numbers. This is bad practice. If you move your tab sprites to a different set of channels, your behavior will no longer work. We'll see in a moment how to make the code more generic.

When you click on the sprite, Director sends it two messages: `#mouseDown` and `#mouseUp`. In this case, you don't want to do anything when the user first presses the mouse, so you ignore the `#mouseDown` message. The `on mouseUp` handler is the recipe that tells the sprite what to do when the user releases the mouse button over the tab sprite.

The `me` parameter is Director's way of referring to the behavior itself. When you

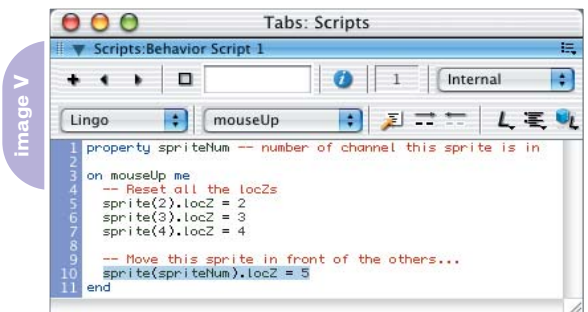
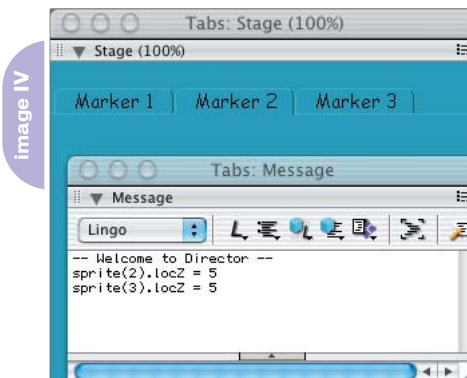
attach a behavior to a sprite, you create a connection between that sprite and a script member. When the Director playback engine first meets the sprite, it creates an instance of the script. Each instance shares the same code, but is stored in a separate place in the computer's Random Access Memory. The `me` parameter stores the address in memory used for that particular instance on that particular sprite.

The word "behavior" is often used ambiguously. Sometimes it means "the script itself"; sometimes it means "an instance of the script in RAM". In this article, I use the words "script" and "instance" explicitly where it is necessary to draw the distinction between the two meanings. We will be taking a closer look at `me` in a moment.

Now run your movie and click on each of the tabs in turn. They pop to the front, just like the real thing. However, the playback head stays at the same marker. You need to add another line of code:

```
tSprite=sprite(spriteNum)
go marker(tSprite.member.name)
```

The name of each tab bitmap member is the same as the marker that you want to display when the user clicks on that tab sprite. You can therefore use the





name of the member to jump to the appropriate marker.

Using Variables

Did you notice that you are now using `sprite(spriteNum)` twice? This forces Director to go back and consult its internal look-up tables a second time. It's much more efficient to save the value for the sprite in a variable, and use the variable instead.

In Lingo, a variable is simply a container; you can put any Lingo value into it. You don't have different variable types for numbers, text strings, or sprite references. In Image VI, I have used two variables: `tSprite` and `tMarker`. The "t" at the beginning of the variable name stands for "temporary". We will create more permanent properties whose names will begin with a "p", shortly. You don't need to use prefixes like this. Director doesn't understand prefixes. Humans do, however, and it can be very helpful when debugging your code to use a variable naming convention.

A temporary variable doesn't need to be declared. It's like a scrap of paper that

Director takes notes on while working inside a particular handler. Once the handler is finished, Director throws the scrap away.

Why Use Matte Ink?

I told you earlier to use Matte ink for your tab sprites. Let's see why. Select all your tab sprites and set them to copy ink. Marker 3 covers the others with a band of white. Try Background Transparent ink.

The sprites now look good, but what happens when you run the movie? Try clicking on the "Marker 2" tab. The playback head jumps to Marker 3. Why? Because although the white background of the sprite is not visible, Director uses the entire bounding box of the bitmap to detect clicks.

With Matte ink when you click on a sprite, Director ignores any white pixels that are not enclosed by non-white pixels. It lets the click pass through to any sprites in lower-numbered channels. Even though the "Marker 3" sprite covers the other sprites entirely, Director lets the mouse events reach the lower-numbered sprites in the area where the "Marker 3" sprite is transparent.

Matte ink is the only ink that affects the way mouse clicks are detected. Other inks affect the way Director draws the sprite.

Time Out

You now have all the code you need to implement a Tab feature in a Director project. If you are Lingo intolerant, you can stop reading this article now. However, the next time you need to use tabs, you'll have to rewrite your behavior

script using the appropriate hard-coded sprite channel numbers. Wouldn't it be better to do a little more work now, so that the next time you need to use tabs, you don't have to open the Script editor? If you think so, read on.

Making the Behavior Generic

To make the behavior generic, you need to ensure that it works regardless of where the tab sprites appear. That means that the behavior has to be able to learn where it is, and where the other sprites with the same behavior are.

Custom Events

A sprite can tell which channel it is in using the `spriteNum` property. But how can one tab sprite know where the other tab sprites are, so that it can move in front? One technique would be to send a message to all the other sprites, telling them to reset their `locZ` values. The `sendAllSprites()` command allows you to do this.

When you click on a sprite, Director automatically sends it a `#mouseUp` event, which is intercepted by the `on mouseUp` handler. You can create your own custom events and match them up with custom handlers with the same name.

Try this in your behavior script using the code shown in Image VII. The `on mouseUp` handler will now send a `#ResetLocZ` event to all sprites, and the `on ResetLocZ` handler knows what to do with it.

First it prints the `spriteNum` of the sprite that received the event in the Message window, along with information on where that sprite's behavior is stored

in the computer's memory (the `me` parameter). Note that the sprites receive the message in order, the lowest-numbered sprite first. The value of `locZ` has no effect on this order. Note also that, although all the sprites share the same behavior script, the value of each instance's `spriteNum` property is different.

Second, the handler resets the `locZ` of the sprite to the same value as its

Image VI

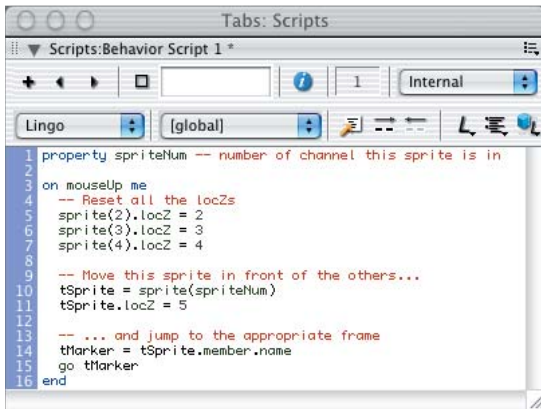
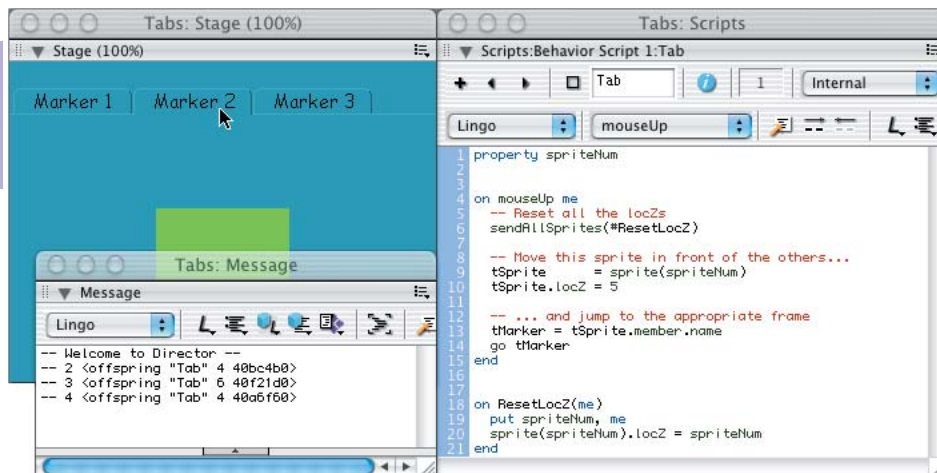
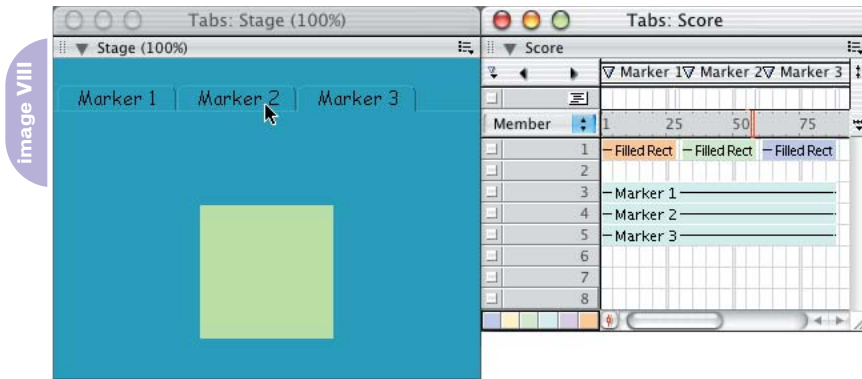


Image VII





spriteNum, which is the bit we are interested in.

You now have only one hard-coded value left in the behavior script: the value 5. What happens if you leave the script as it is and move the sprites up one channel?

Tip: Select the three tab sprites on the Stage or in the Score and press the Ctrl-Up arrow, or the Apple-Up arrow if you are on Macintosh. Now run the movie and click on the “Marker 2” tab. The playback head jumps to Marker 2, but the tab sprite does not appear in front of the sprite in channel 5 (see Image VIII).

Sharing Data with Other Behaviors

One way to avoid this issue would be to set the locZ of the selected tab sprite to the lastChannel + 1. This would move

it in front of all the other sprites.

However, if your application allowed the user to drag and drop items, they might pass behind the tab sprite, and this would look odd. In my experience, it is better to keep the tab sprites in low-numbered channels.

There is another drawback with the current technique: sendAllSprites() can badly slow down your movie. In the current case, this isn't obvious. I have encountered projects where one sprite used sendAllSprites() to broadcast one event, and the sprites that received the event sent out other events, again using sendAllSprites(), and so on. What looked like a single line of code ended up in a flood of messages that almost drowned the machine. If used incorrectly, SendAllSprites could become the Lingo equivalent of the chain letter.

How can the behavior instances

determine which tab sprite is in the highest-numbered channel so that they can set the locZ of the front-most sprite to a higher value? The solution that I propose also reduces the use of

Lingo Lists

In Lingo, a list is like an address book: it contains information about where things are. Just as an address book does not contain the people it refers to, so a

list variable does not contain the data itself. It contains the address where that data can be found in the computer's memory.

If you use two variables to refer to the same list, and then change the contents of one list, the contents of the other variable will change too. Try it in the Message window:

```
gList1 = []
put gList
-- []
gList2 = gList1
-- Change gList2...
gList2.add(#aValue)
-- ... and gList1 changes too:
put gList1
-- [#aValue]
```

Note: I use “g” as a prefix here because variables created in the Message window are global. Any script in any movie can access and alter the value of a global variable... so long as it is properly declared.

You're now ready to use this feature in your behavior. You'll find the final version of the script in Image IX.

There are six new Lingo expressions to learn about:

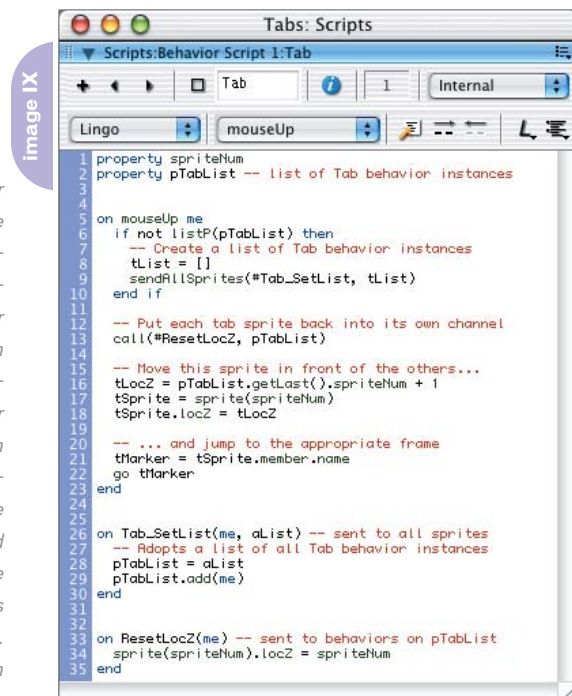
1. if ... then ... end if
2. listP
3. not
4. []
5. getLast()
6. call()

These are all explained in the Lingo dictionary and the online Director help.

“[]” appears at the beginning of the Dictionary, among all the other non-alphabetical characters.

Basically, what happens now is this: when the behavior instances are first created, the property pTabList has no value in any of the instances. The first time you click on a tab sprite, that behavior instance realizes that pTabList is not yet a list. Obliging, it creates a new, temporary list, and sends it out to all sprites on the back of a custom #Tab_SetList event. The custom event's name starts with #Tab_, so the chances are that only the Tab behavior will have a handler of that name. Instances of other behaviors in your movie will ignore it.

The on Tab_SetList handler in each Tab behavior instance receives the list in



James Newton works for OpenSpark Interactive Ltd. The company specializes in designing multi-media applications for improving production processes. His contributions to the Behavior Library, and his articles on Imaging Lingo, 3D mathematics, LDMs, the MultiUser Server and other Director topics have helped Lingo users at all levels.
james.newton@openspark.com

the aList parameter. The “a” prefix stands for “attribute”, meaning that the value was sent from somewhere else, not created in the handler itself, like a temporary variable. When the on Tab_SetList handler is completed, the original tList variable will still be held in the computer’s memory... just as long as it takes the initial on mouseUp handler to complete.

The on Tab_SetList handler in each behavior instance does two things: it sets its own pTabList property so that it refers to the list, and it adds a reference to itself to the list.

Each Tab behavior now knows about all the other Tab behaviors: each behavior has a pTabList property which stores all the behavior’s addresses, in the order of their sprite channels. This means that the last entry in the list refers to the behavior on the sprite in the highest channel. Add 1 to the spriteNum of this last behavior, and you get a locZ in front of all the Tab sprites.

Instead of sending a message to all sprites, you can now simply send a message to the list of Tab behaviors.

Let’s run through that again.

- The behavior instances are all in differ-

ent places in the computer’s memory. The fact that each has a different value of spriteNum proves this.

- All the behavior instances have their own pTabList property, but by sleight of code, all these different properties refer to the same list.
- The contents of this unique list is a set of variables, each of which refers to one of the Tab behavior instances.
- Because of the way Director talks to sprites, these instances are ordered by spriteNum, so the last instance is attached to the sprite in the highest-numbered channel.
- You can use call() to send a message to all the instances in a list.

In other words, the pTabList is like a private club, where everyone knows everyone. The sendAllSprites() command is used just once to create the club.

Conclusion


You need to make one final tweak: reverse the order of the tab sprites so that the “Marker 1” tab appears in the highest-numbered channel. That way, the correct tab will be in front when you start your movie.

You now have a generic behavior that can be used in any sprite channel. The behavior in the download movie contains a couple of changes for reasons of optimization, and is much better commented. You may want to add it to your code library.

It’s taken just 21 lines of code to make a robust behavior. How could this behavior be improved? Here’s a brief shopping list of possible enhancements:

- Create the tab members automatically with a given text label and icon
- Use a single sprite
- Do more than just jumping to a given marker
- Move the appropriate tab sprite to the front if the user navigates to a marker other than by clicking on a tab
- Add support for more than one group of Tab sprites on a given page

Where to Go from Here

MeccaMedialight has made available a widgets library containing open source code for creating Tab and other controls, via their LingoWorkshop site: www.lingoworkshop.com/code/widgets3_lib_tabs.php 



Ending CPU Hogging

Sleep that offers work
by alex zavatone

Apparently since the dawn of time, Director has had quite an affinity for the processor and its cycles. In fact, Director appears to like the processor so much that when running, there is little to no free processor time. In the old days, Director on the Mac had CPUHogticks to determine an amount of time to give back to the processor, but nowadays that command doesn't exist and CPU hogging becomes an issue for the following reasons:

- On laptops, hogging the processor will result in 100% processor allocation, which will result in a hot processor, which will turn the fan and lessen battery life.
- On OS X, your whole system becomes less snappy.
- Director does not play well with other applications if it asks for all the remaining processor power and may take processor power away from other subsystems that the projector may access.
- Hogging the processor makes distributing Director utility applications unrealistic.

So, enough about the problems. Why does it happen and how do we fix it?

From several conversations last century with various Director engineers, I learned several things. Among them, the movie-level Idle handler is only called when Director has nothing else to do or has spare cycles. This means that at times

Director has spare cycles, cycles that could be used by other applications. Also, if Idle is called and completes, and there are spare cycles left, Idle will be called *again* and again until there are no spare cycles present, thereby keeping the spare cycles from being used by other processes. This will happen whether you have put an Idle handler in your movie or not. Following this logic, it would seem that if we were able to trap this Idle event in an Idle han-

command. Amply caffeinated, an aha moment happened and I decided to see what would happen if Director slept for 1 millisecond when Idle was called.

That was it.

This simple handler stopped Director from hogging the processor.

```
On Idle
  sleep 1
end
```

“At times Director has spare cycles, cycles that could be used by other applications”

andler and sleep Director for an appropriately small period, then Director would not be gobbling cycles. Since it wasn't using them anyway this would free up the processor for other tasks *and* still allow Director to operate as fast as it needs to!

Now I know you're saying "Oh, that sounds lovely Sparky, how do you suggest we do this?" And this, gentle reader, is the good part.

From a conversation with Warren Ockrassa about his love of undocumented Xtras, I started playing around with one of those Xtras. In it I found a sleep

If you run or produce Director software for OS X, this has profound effects, all of them good. Director running on Windows also exhibits better processor usage but the benefits are less pronounced.

The trick is that whether running a Director movie on the stage or in a MIAW, there is only a limited amount of idle or free time that Director can use as an opportunity to call Idle. This means that only one of these Idle handlers is needed for any Director session. More than one Idle handler split between the stage and



Advertising Index

any running MIAWs is not needed.

Now, the tricky part is that this aforementioned Xtra, which comes from Macromedia, has different names in different versions and platforms of Director. Great.

It started out as the "UI Helper Xtra", made to support Director's Export to Java functionality in D7. In Director MX, it was renamed the "Watcher Helper Xtra". On Windows, it was originally called "JavaUIHelper.x32". On certain versions of Director for Windows (8, 8.5), this xtra must be installed off of the goodies or xtra partners folder on the Director install CD. As far as I know, it is called "WatcherHelper.x32" on DMX for Windows.

In any case, you can always go to the message window and type the following line to see if the xtra you suspect has the sleep method in it.

```
put interface(xtra "the name of the
xtra I suspect is the one I care
about")
```

Once you have identified the xtra for the version of Director you are using, you must manually include the xtra in the movie xtras list for use within a projector since it is not an asset xtra.

Now make your projector, launch it, and run a processor monitoring app. You'll be pleasantly surprised. Image 1 provides a snapshot of a DMX projector playing nice with other apps on my 1G Ti PB.

A projector using only 11% of the CPU is a joyous site to see.

Think we're done yet?

There are caveats though and, luckily, solutions. Colin Holgate mentioned that this technique is great unless your projector is set to "not animate in the background." Sure enough, when I tested it a no animate projector on OS X started eating up all the processor power when put into the background. Colin's solution to this was to use a regular "animate in background" projector and to trap the activateApplication and deactivateApplication handlers. Within those handlers, set a global used for the sleep value to 1 on activateApplication and to a higher number on deactivateApplication. This technique combined with a regular animate in background projector allows the projector free up

Advertiser	URL	Phone	Page
activePDF	www.activepdf.com		15
ColdFusion Developer's Journal	www.sys-con.com/cfdj/subscription.cfm	888-303-5282	53
CFDynamics	www.cfdynamics.com	866-CFDYNAMICS	9
Cfun '04	www.cfconf.org/cfun-04		75
del Padre Visual Productions	www.delpadre.com		74
Edge Web Hosting	www.edgewebhosting.net/cfdj		21
HostMySite.com	www.hostmysite.com/mxdj	877-248-4678	41
Information Storage + Security Journal	www.issjournal.com	888-303-5282	63
Interakt	www.interaktonline.com		6
IT Solutions Guide	www.sys-con.com/it	201-802-3021	58
Macromedia	www.macromedia.com		2--3, 76
MX Developer's Journal	www.sys-con.com/mx/subscription.cfm	888-303-5282	51
Seapine Software	www.seapine.com	888-683-6456	11
Serverside	www.serverside.net	888-682-2544	27
SYS-CON e-newsletters	www.sys-con.com	888-303-5282	71
SYS-CON Publications	www.sys-con.com/2001/sub.cfm	888-303-5282	69
SYS-CON Reprints	www.sys-con.com	201-802-3026	45
Webcore Technologies	www.webcoretech.com	877-WCT-HOST	33

even more time when in the background.

Requested since before the dawn of time, the end to CPU hogging in authoring and projectors is here.

Enjoy.

• • •

Thanks for contributing to this discovery/technique go to Warren Ockrassa for general troublemaking and research; Joe Siponen (pointing out the xtra name differences between versions); Troy Rollins (MIAW experiments); and Colin Holgate (by default). Members of a number of Director-oriented mailing lists also helped flesh out this technique. ☺

Alex (Zav) Zavatore previously worked on Macromedia's Director/Shockwave Engineering team performing QA for Lingo sometime last century. Between then and now, he has done stuff and even performed things (product design, architecture, and implementation). Marooned in San Francisco and when not being sarcastic, he has been spotted being formal, stuffy, and consulting for clients whenever able. zavpublic-at-mac.com



Dream Out Loud

d el Padre Visual Productions has recently completed the design of a Macromedia Flash MX Professional 2004-based Digital Business Card for aerospace and defense contractor BAE Systems. DVP designers set out to design the presentation entirely in Flash MX Professional 2004 and Alias Maya 5.0. An interactive interface console allows the user to choose from three forms of content: a three-minute video overview of the SSE group, a Flash slideshow-type presentation including additional information about the parent company, and a very dynamic high-tech contact screen. Each link initiates a "fly in" animation from the interface console position to the content area, which is three large video monitors on the far wall of the Mission Control environment.

www.delpadre.com



CFUN4



301.424.3903

info@teratech.com

Two whole days of...
Networking! Learning! Fun!

Sixth Annual ColdFusion Conference

June 26th & 27th, 2004

Charlie Arehart	Simon Horwith
Jo Belyea-Doerrman	Larry Hull
Raymond Camden	Chafic Kazoun
Christian Cantrell	Matt Liotta
Sandra Clark	Tom Muck
Sean Corfield	Rey Muradaz
Robert Diamond	Nate Nelson
Michael Dinowitz	Samuel Neff
Steve Drucker	Jeff Peters
David Epler	John Quarto
April Fleming	Neil Ross
Ben Forta *tentative	Stephen Shapiro
Shlomy Gantz	Michael Smith
Critter Gewlas	Geoff Snowman
Mark Gorkin	Jeff Tapper
Hal Helms	Dave Watts

\$199

Per Person
Special Price until 3/31/04

5 TRACKS!

Bootcamp - Basic ColdFusion and Flash topics

Advanced - Advanced ColdFusion topics

Empowered - Fusebox & Project management topics

Accessibility - making sites that disabled people can use, section 508

Integration - Flash, Flex and other technologies integrated with CF topics

www.cfconf.org/cfun-04/


"I learned numerous techniques last year that have helped myself and our development team to better deliver quality products to our customers. CFUN is also a great venue to meet some of the names you see in CFDJ and DevNet and talk with them one on one."

-Phillip D





Consumer
Products



Music



Fighting AIDS



Water

INTO

WHAT ARE YOU INTO?

At Macromedia, we're continually inspired by the passion of our customers. Visit "Into" to see their stories, or share your own. www.macromedia.com/into



Announcing Macromedia MX 2004:
New versions of Macromedia® Flash™, Dreamweaver®,
Fireworks® and Studio. [Run with it.](#)

Copyright © 2003 Macromedia, Inc. and its licensors. All rights reserved. Macromedia, the Macromedia logo, Dreamweaver, Flash, Fireworks, and Macromedia Flash are trademarks or registered trademarks of Macromedia, Inc. in the United States and/or other countries. Other marks are the properties of their respective owners.